CSRI SUMMER PROCEEDINGS 2024

CSRI Summer Program
The Center for Computing Research at Sandia National Laboratories

Editors:
M. Adams, T. Casey, B.W. Reuter
Sandia National Laboratories

December 17, 2024

SAND#: SAND2024-1668O

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

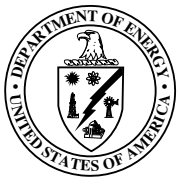
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.doe.gov/bridge

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/ordering.htm



Preface

The **Computer Science Research Institute (CSRI)** brings university faculty and students to Sandia National Laboratories for focused collaborative research on Department of Energy (DOE) computer and computational science problems. The institute provides an opportunity for university researchers to learn about problems in computer and computational science at DOE laboratories, and help transfer results of their research to programs at the labs. Some specific CSRI research interest areas are: scalable solvers, optimization, algebraic preconditioners, graph-based, discrete, and combinatorial algorithms, uncertainty estimation, validation and verification methods, mesh generation, dynamic load-balancing, virus and other malicious-code defense, visualization, scalable cluster computers, beyond Moore's Law computing, exascale computing tools and application design, reduced order and multiscale modeling, parallel input/output, and theoretical computer science. The CSRI Summer Program is organized by CSRI and includes a weekly seminar series and the publication of a summer proceedings.

1. CSRI Summer Program 2024. In 2024 the CSRI summer program was again executed in a hybrid fashion, with a large cohort of student interns working on-site at both Albuquerque and Livermore campuses joined by their remote colleagues in program activities virtually. Most remote students were able to at least visit the Albuquerque or Livermore campuses during the internship. The summer program included students from 1400 – the Center for Computing Research (CCR) and 8700 – the Center for Computation & Analysis for National Security (CANS). This year's program included the traditional *Summer Seminar Series* and *Summer Proceedings*, continued the annual *Lightning Talks* session (previously *Virtual Poster Blitz*), and included a special seminar on 6/26/24 on software development best-practices with the "Introduction to Collaborative Git Workshop for Interns", presented by Sylvain Bernard and Miranda Mundt from Org. 1424. Our participation in the Computational Science and Analysis Institute (CSA) continued again this year with our partners in the Engineering Sciences Summer Institute (ESSI). Together, we provided four sessions on professional development with Dr. Jacquilyn Weeks from Word Tree Consulting, supported in-person site visits for some virtual interns, and ran an escape room and various facilities tours at Sandia, including the CA Data Center and Combustion Research Facility lab, and in the surrounding areas.

2. Seminar Series. The CSRI Summer Seminar Series is a quintessential part of the CSRI Summer Intern Program Experience. Students are exposed to a broad showcase of research from across Sandia, enriching their knowledge of the labs while providing introductions to novel subject areas, as well as different career paths available in the national labs. We extend our deepest thanks to the staff who spoke at the 2024 Seminar Series. These speakers and their talk titles are listed in Table 2.1.

Table 2.1: List of talks and speakers in the 2024 Seminar Series

Date	Name	Org	Title
6/6	Chris Siefert	1465	From PDEs to Linear Algebra to High Performance Computing
6/13	Amanda Dodd	8700	An Introduction to Sandia, its Missions and Computational Research at the lab
6/17	Jeff Woolstrum	1684	FLEXO and Fusion
6/27	Meg McCarthy & Ember Sikorski	1444	How to Train your Atoms: the LAMMPS and FitSNAP Codes at Sandia
7/1	Bill Rider	1544	Verification and Validation with Uncertainty Quantification is the Scientific Method for Computational Science
7/9	Dan Ibanez	1443	Evolving HPC Hardware and Shock Physics Software
7/15	Brian Granzow	1443	Mesh Adaptivity in Computational Physics
7/29	Kara Peterson	1442	Climate Modeling & Analysis at Sandia: An Applied Mathematician's Perspective
8/5	Ben Feinberg	1422	Everything Old is New Again: The Past, Present, and Future of Analog Computing
8/12	Pieterjan Robbe	8351	Between Phases: Understanding Uncertainty in Nuclear Fission Gas Release via Phase Field Models
8/19	Christophe Bonneville	8734	Accelerating Phase Field Simulations Through a Hybrid Adaptive Fourier Neural Operator with U-Net Backbone

3. Lightning Talks. In preparation for the proceedings, *lightning talks* were held on 7/31/2024 where all interns hired under the CSRI intern posting were invited to participate. These students submitted a summary slide of their current results or intended research for their summer project, and gave a two minute elevator-pitch style presentation to their peers. Other interns outside of CSRI were also invited to participate with their mentor's approval. This event provided an opportunity to formalize work directions, socialize their ideas, as well as offering a chance to network and interact with other interns and staff across the labs. The slides from the lightning talk event are published under SAND2024-10948PE.

4. Proceedings. All students and their mentors were strongly encouraged to contribute a technical article to the CSRI Proceedings. For many students, these proceedings are the first opportunity to write a research article. These proceedings serve both as documentation of summer research and also as research training, providing students the first draft of an article that could be submitted to a peer-reviewed journal. Each of these articles has been reviewed by a Sandia staff member knowledgeable in the technical area, with feedback provided to the authors. Contributions to the 2024 CSRI Proceedings have been organized into three categories: *Computational & Applied Mathematics, High Performance & Post-Moore Computing* and *Machine Learning*.

All participants and their mentors who have contributed their technical accomplishments to the proceedings should be proud of their work and we congratulate and thank them for participating. Additionally, we would like to thank those who reviewed articles for

the proceedings. Their feedback is an extremely important part of the research training process and has significantly improved the proceedings quality. Our many thanks are extended to these reviewers: Jonas Actor ✧ Andrew Baczewski ✧ Sneha Banerjee ✧ Jon Berry ✧ Patrick Blonigan ✧ Tiernan Casey ✧ Jeff Connors ✧ Mary Alice Cusentino ✧ Aditya Dhumuntarao ✧ Alejandro Diaz ✧ Matthew Dosangh ✧ Danny Dunlavy ✧ Christian Gilbertson ✧ James Goff ✧ Brian Granzow ✧ Joey Hart ✧ Dan Ibanez ✧ John Jakeman ✧ Anders Johansson ✧ John Kallaugher ✧ Hemanth Kolla ✧ Paul Kuberry ✧ Richard Lehoucq ✧ Justin Li ✧ Jay Lofstead ✧ Denis Mamaluy ✧ Kathryn Maupin ✧ Jerry McNeish ✧ Meg McCarthy ✧ Shane McQuarrie ✧ Stan Moore ✧ Erin Mussoni ✧ Justin Owen ✧ Steve Owen ✧ Ravi Patel ✧ Tim Proctor ✧ Siva Rajamanickam ✧ Jaideep Ray ✧ Denis Ridzal ✧ Antonio Russo ✧ Mohan Sarovar ✧ Whit Schonbein ✧ Chris Siefert ✧ Andrew Steyer ✧ Carlos Michelon Strofer ✧ Corinne Teeter ✧ Josh Teves ✧ Irina Tezaur ✧ Aidan Thompson ✧ Anh Tranh ✧ Craig Ulmer ✧ Ryan Viertal ✧ Ruben Villarreal ✧ Rado Vuchkov ✧ Chris Wentland ✧ Nick Winovich ✧ Mitch Wood ✧ Patrick Xiao ✧ Ichi Yamazaki ✧ Tian Yu Yen ✧ Tianyi Zhang ✧.

Deepest Thanks. We would like to thank all students and mentors for their extraordinary patience and dedication, especially given the hybrid nature of the intern program. We would also like to thank the program managers for the CSRI Summer Intern Program, Michael Wolf (1465) and Jerry McNeish (8734); as always, their support has been critical throughout the organization of the seminars and the editing of these proceedings. Furthermore, the CSRI Summer Intern Program would not be possible without the administrative support of Lisa Mahkee, Sandra Portlock, Hailey Poole, Sabrina Ahumada, and many others. We would also like to extend a very special thank you to Erin Stelter of the CSA program, Gaby Bran Anleu of the ESSI program, Sasha Safonov and Steven Barker of the CCD program, and Jacob Davis for his role in setting up tours. Their assistance greatly enriched the experiences of our interns this year, and we are deeply appreciative of their help.

M. Adams
T. Casey
B.W. Reuter

December 17, 2024

Table of Contents

Articles

I. Computational & Applied Mathematics

Computational & Applied Mathematics are concerned with the design, analysis, and implementation of algorithms to solve mathematical, scientific, or engineering problems. Articles in this section describe methods to design or analyze new algorithms, discretize and solve partial differential equations, couple multiphysics systems of equations, and analyze sensitivity & quantify uncertainty in complex systems.

1. *G.H. Brown, E.T. Phipps, H. Jolla, D.L. Bull,* and *T.S. Ehrmann* Extracting Climate Phenomena: Beyond PCA
2. *A. de Castro* and *P. Kuberry* Comparing Stability of Partitioned Heterogeneous Time-integration Methods Involving Index-2 DAEs Resulting from High-order Adams-Moulton and Backward Difference Formula Time Integration Schemes
3. *A. Chin* and *T. Catanach* Backwards Sequential Monte Carlo for Efficient Bayesian Optimal Experimental Design
4. *E. Crislip, M. Khalil,* and *K. Neal* Data Assimilation: Addressing Spurious Correlations and Scalability Issues
5. *M. Drayton, R. Bandy,* and *T. Portone* Uncertainty in Reduced Finite-rate Ablation Models for Reentry Vehicles
6. *R. de Farias, M.B.P. Adams,* and *W.J. Rider* Implications of the Two Interacting Blast Wave Verification Problem for Computational Shock Hydrodynamics
7. *D. Hughes, C. Eldred* and *E.C. Cyr* Discrete Exterior Calculus for Hodge-Helmholtz Problem
8. *E. Huynh, P. Bochev,* and *E & P. Kuberry* A DMD-based Partitioned Scheme for Time-Dependent Coupled Parametric PDEs
9. *L.F. Maia, R. Baraldi,* and *D.P Kouri* An Inexact Weighted Proximal Trust-Region Method
10. *I. Moore, C.R. Wentland, A. Gruber,* and *I. Tezaur* Domain Decomposition-based Coupling of Operator Inference Reduced Order Models Via the Schwarz Alternating Method
11. *R. Pawar* and *P. Bochev* Operator Inference Based Flux Surrogate Algorithm for Coupled Transmission Problems
12. *B. Shawcroft, K.M. Shepherd* and *S.A. Mitchell* TUSQH: Topological Control of Volume-fraction Meshes Near Small Features and Ugly Geometry
13. *M. Tunnell* and *E.G. Boman* Parallel Incomplete LU Factorizations Based on Alternating Triangular Solves
14. *A. Vijaywargiya, S.A. McQuarrie* and *A. Gruber* Tensor Parametric Operator Inference with Hamiltonian Structure
15. *E. Whitesides, J.P. Mendez, J. Ivie, X. Gao,* and *S. Misra* Simulating Atomic Precision Advanced Manufacturing (APAM) Enhanced BJT

M. Adams
T. Casey
B.W. Reuter

December 17, 2024

EXTRACTING CLIMATE PHENOMENA: BEYOND PCA

GABRIEL H. BROWN*, ERIC. T PHIPPS†, HEMANTH KOLLA‡, DIANA L. BULL§, AND THOMAS S. EHRMANN§

Abstract. Two new methods for the calculation of climate indices, one based on tensors and another on affine subspaces, are proposed and experimentally studied by application to the Madden-Julian oscillation. The tensor method is found to perform well when the data is highly preprocessed, but struggles as the amount of preprocessing decreases. A theoretical mechanism for this behavior is conjectured. Unlike the tensor method, which requires only a reference binary timeseries for the phenomena of interest (present/not present), the affine method requires a continuous reference timeseries. However, it offers distinct advantages over more common methods like PCA including: the ability to map a climate index from one level of data preprocessing to another, the obviation of a combinatorial search. Preliminary experiments show that the drop in performance is lesser for the affine approach when the amount of preprocessing is reduced. We believe the affine method merits further study as a technique to approximately invert possibly non-invertible, nonlinear preprocessing steps. Such an approach could enable indices derived from preprocessed data to be viewed in the full, unprocessed climate space, providing direct insight into connections between the phenomenon of interest and more dominant phenomena.

1. Background. Climate indices are scalar values, usually computed and reported daily or monthly, which indicate the strength or presence of some particular phenomena. For example, there are dozens of standard climate indices tracking phenomena like precipitation, modes of variability like El Niño Southern Oscillation (ENSO), and heatwaves. Such indices can be used by meteorologists and forecasters to quickly extract information on phenomena relevant to global and local weather in the near future.

Such climate indices are usually defined and computed using a two stage offline-online approach. The offline stage uses historical climate data to define an approximation for the phenomena, usually the linear subspace implied by columns of a truncated principal component decomposition. In the online stage, climate agencies use current climate data and the phenomenon's approximate representation to compute a daily or monthly index value, often via projection. A full pipeline for the computation of a projection-based index is detailed in the Methods section below.

The historical climate data used in the offline stage is most naturally thought of as a dense tensor of shape $(n_{\text{space}}, n_{\text{var}}, n_{\text{time}})$; in particular, we will assume $n_{\text{space}} \in \{n_{\text{lon}}, n_{\text{lat}} n_{\text{lon}}\}$, and n_{var} is the number of variables (e.g. sea surface temperature, windspeed, precipitation). As a consequence of the data we use, this work includes altitude dependence implicitly using a variable measured on two isobaric surfaces, but altitude may be included directly in n_{space}. When longitude and latitude are used together, we combine them into a single tensor mode since this results in approximate representations which are more interpretable by climate scientists.

While climate data is most naturally arranged as a tensor, PCA (called empirical orthogonal functions in the climate literature) is the most widely used technique to approximate and intepret climate phenomena from data [10]. To apply PCA the climate tensor is first unfolded, combining all non-temporal modes into the columns, and having each column represent a different time (or vice versa). In such cases, it is often valuable to try applying tensor methods, which respect the natural layout of the data, among other possible benefits like uniqueness and ability to capture higher order connections.

*Oden Institute, University of Texas at Austin, ghbrown@utexas.edu
†Sandia National Laboratory, Scalable Algorithms (01465), etphipp@sandia.gov
‡Sandia National Laboratory, Scalable Modeling & Analysis (08753), hnkolla@sandia.gov
§Sandia National Laboratory, Climate Systems (08931), dlbull@sandia.gov, tsehrma@sandia.gov

As a first attempt to apply tensor methods to climate science, we develop a tensor decomposition based approach for computing a climate index and compare to indices generated using PCA. Inspired by some of the theoretical shortcomings of this tensor method, we develop a new approach based on affine subspaces which has unique advantages over traditional climate index methods, though operates in a slightly altered setting. A simple and prototype implementation of this method is also compared to indices generated via PCA, and shows promising behavior.

We believe the affine method can provide a route to overcoming difficulties arising from the ubiquitous procedure of data preprocessing. If a climate index for the phenomenon of interest is based on the decomposition of preprocessed data, then "raw" climate data (living the unprocessed climate space) cannot be directly compared with the decomposition but instead must first be preprocessed. The cost of processing new data daily or monthly is not prohibitive, but forces users of the index to interpret and view the decomposition and the processed daily data in the preprocessed climate space, which does not necessarily reflect reality. We propose the affine method as a way to approximately invert the effects of the processing on the index, enabling an effective decomposition in the full, unprocessed climate space.

2. Methods.

2.1. Projection-based climate index. We will focus on the computation of a climate index computed using a projection onto a linear space. If $\mathbf{d} \in \mathbb{R}^{n_{\text{space}} n_{\text{var}}}$ is daily climate data, and $\mathbf{B} \in \mathbb{R}^{n_{\text{space}} n_{\text{var}} \times k}$ is some basis for the phenomena of interest, then the climate index for that day would be given by $\|\mathbf{P_B d}\|_2$, where $\mathbf{P_B}$ is the projector on to the linear subspace $\text{col}(\mathbf{B})$, the column space of $\mathbf{B}$. The motivation for such a formulation is clear: if $\mathbf{d}$ lies near $\text{col}(\mathbf{B})$ (intuitively the weather is similar) then not much will be lost in the projection, alternatively if $\mathbf{d}$ is nearly orthogonal to $\text{col}(\mathbf{B})$ (the weather is intuitively dissimilar) the norm of of the projection will be small.

Note the importance of $\mathbf{B}$ and $\mathbf{d}$ being generated by the same preprocessing. Suppose $\mathbf{B}$ is a basis for a linear space extracted from data from which the seasonal cycles (for example) had been removed. One cannot fairly project daily climate data $\mathbf{d}$ that has not had its seasonal component removed, $\mathbf{B}$ is simply not a meaningful basis for such vectors. This restriction sometimes complicates the construction of processes to compute daily indices. First, steps like annual cycle removal must rely on partial information about the yearly climate to date, rather than a complete dataset of yearly climate. Perhaps more importantly, intuition or meaning attached to a basis $\mathbf{B}$ (or really the linear space it represents) is strongly tied to the level of preprocessing used in its creation. Preprocessing steps are often non-invertible or non-trivial to apply to matrices, meaning that it can be hard to translate the influence of the phenomenon captured by $\mathbf{B}$ onto the unprocessed global weather state.

2.2. Similarity measures for timeseries. Throughout this work it will be necessary to estimate the similarity between two timeseries $\mathbf{x}$ and $\mathbf{y}$.

If the timeseries are continuous we will use the Pearson correlation coefficient

$$r(\mathbf{x}, \mathbf{y}) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \sqrt{\sum_i (y_i - \bar{y})^2}}. \tag{2.1}$$

If the continuous timeseries have an associated threshold (which can be used to cast to a binary timeseries) or are binary, then one can use the Peirce skill score

$$\text{PSS}(\mathbf{x}, \mathbf{y}) = \frac{TP \cdot FP - TN \cdot FN}{(TP + FN)(FP + TN)}, \tag{2.2}$$

where TP is the number of true positives, FN is the number of false negatives, and so on [6]. For Peirce skill score the second argument $\mathbf{y}$ is treated as the reference timeseries for the purpose of determining "true" and "false". Throughout we use the Peirce skill score to compare the performance of a thresholded timeseries with a reference binary forecast, as well as to select the threshold for a timeseries (discussed in more detail below).

2.3. CP tensor decomposition. The canonical polyadic (CP) decomposition of a tensor is a representation of the tensor as a sum of outer products. If $\boldsymbol{\mathcal{X}}$ is rank-r, then it may be written as the sum of r vector outer products and no fewer, $\boldsymbol{\mathcal{X}} = \mathbf{a}^{(1)} \circ \mathbf{b}^{(1)} \circ \mathbf{c}^{(1)} + ... + \mathbf{a}^{(r)} \circ \mathbf{b}^{(r)} \circ \mathbf{c}^{(r)}$. It is often convenient to organize these vectors into so called factor matrices, for example $\mathbf{A} = \begin{bmatrix}\mathbf{a}^{(1)} & ... & \mathbf{a}^{(r)}\end{bmatrix}$, and use the following shorthand to denote the reconstruction of the tensor from these factor matrices

$$\boldsymbol{\mathcal{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \coloneqq \sum_{i=1}^{r} \mathbf{a}^{(i)} \circ \mathbf{b}^{(i)} \circ \mathbf{c}^{(i)}. \tag{2.3}$$

However, computing an exact decomposition is rarely useful or practical in application settings. More frequently, low-rank CP decompositions are applied as a tool to approximate tensors. To obtain a *simultaneous* rank-k CP approximation to $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{m \times n \times p}$ one would solve

$$\min_{\mathbf{A} \in \mathbb{R}^{m \times k}, \mathbf{B} \in \mathbb{R}^{n \times k}, \mathbf{C} \in \mathbb{R}^{p \times k}} \|\boldsymbol{\mathcal{X}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2. \tag{2.4}$$

The problem in Equation 2.4 may be solved in a variety of ways, including gradient descent, but the most popular algorithm is an alternating least squares algorithm [5]. A single "sweep" of this algorithm solves a linear least squares problem for a locally optimal value of $\mathbf{A}$ given fixed $\mathbf{B}$ and $\mathbf{C}$, a locally optimal value of $\mathbf{B}$ given fixed $\mathbf{A}$ and $\mathbf{C}$, and similarly for $\mathbf{C}$. Sweeps continue until some convergence criterion in satisfied.

Alternatively, one could solve a sequence of problems like 2.4, subtracting off reconstructions of existing components. At the jth step of this approach, one would solve

$$\min_{\mathbf{A}^{(j)} \in \mathbb{R}^{m \times k}, \mathbf{B}^{(j)} \in \mathbb{R}^{n \times k}, \mathbf{C}^{(j)} \in \mathbb{R}^{p \times k}} \left\| \boldsymbol{\mathcal{X}} - \sum_{i=1}^{j-1} [\![\mathbf{A}^{(i)}, \mathbf{B}^{(i)}, \mathbf{C}^{(i)}]\!] - [\![\mathbf{A}^{(j)}, \mathbf{B}^{(j)}, \mathbf{C}^{(j)}]\!] \right\|_F^2. \tag{2.5}$$

This is the so called *sequential* or greedy approach to CP decomposition, and is not equivalent to the simultaneous approach. After p steps of this greedy process one would have a CP approximation to $\boldsymbol{\mathcal{X}}$ rank-pk. Strictly speaking pk is the number of components in the decomposition, and the decomposition could have a lower rank; the same caveat applies to the simultaneous approach above, but we will nevertheless use "rank" colloquially to describe the number of components in a CP decomposition.

2.4. Choosing between simultaneous and sequential CP approximations. The primary goal in this work is to effectively isolate the linear subspace associated with a phenomenon of interest, which in turn defines a timeseries climate index. This subsection is dedicated to explaining the strengths and weaknesses of simultaneous and sequential CP approximations for this task.

We begin by providing a basic overview of the properties and behavior of each method.

	strength	weakness
simultaneous	• exact selection of rank can result in unique solution • optimal error for a given rank (assuming global minima is found)	• finding exact rank is NP-hard [4] • no relationship between rank-k and $k+1$ decomposition, components may be entirely different • may converge to a local minimum
sequential	• components nest as number of components increase (by construction, assuming fixed initialization)	• very few theoretical guarantees or studies • suboptimal error for given rank • no clear way to select k • may converge to a local minimum (at each step)

We note there are two primary ways to use a CP decomposition: compression and interpretation. In the former one only uses the decomposition as an approximation to the original tensor, while in the latter case the components are interpreted or otherwise used directly. Of the two, the latter is more difficult; the different in reconstruction error between a rank r and $r+1$ approximant is likely small, but it is possible for the factors to change drastically. Furthermore, well-posedness and existence of solutions is only an issue when interpretation of components is necessary; the reconstructed tensor will converge, but the components may not [2]. Our direct use of the components to form a linear subspace defining the climate index means we must be more wary of such issues.

In other contexts, such as blind sources separation, simultaneous CP approximation is often applied to a tensor that is **assumed or known to have some low-rank outer product structure** [1]. Such knowledge about the tensor is related to physical and mathematical knowledge of the data generation process. Even if the data tensor is noisy, it is often possible to tell (after iterating over different ranks) when the appropriate rank has been chosen. In these cases the recovered components are often similar to the "true" factors which generated the data. A sequential CP decomposition of the data enjoys no such properties, often requiring a much higher rank to achieve a comparable residual while providing components different than those which generated the data.

We emphasize that we are not in the aforementioned setting, as **climate tensors are almost certainly not created by some outer product-like data generation process**. Instead, climate data is the sampling of the solution of a nonlinear partial differential equation on a grid of space, time, and variables that is often tensorial. Such a tensor will have *some* rank, but we believe it will be neither low nor meaningful. As such, we cannot assume that the climate phenomenon of interest is comprised of a subset of the computed components, regardless of whether the CP approximation is computed simultaneously or sequentially. This will be a fundamental problem for the application of CP to the task of extracting climate phenomenon from data which is not dominated by said phenomena. In fairness, common linear algebra methods like singular value decomposition or principal components are expected to behave similarly, as the returned vectors are guaranteed only to be optimal approximations rather than interpretable components of one phenomenon or another. The most important conclusion of this discussion is that **we can expect to enjoy none/few of the positive properties usually associated with (simultaneous) CP decompositions**.

2.5. Projection onto a CP decomposition. Anticipating a projection-based climate index based on an approximate CP decomposition, we define the projection of tensor data onto a CP decomposition. Letting $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{n_{\text{space}} \times n_{\text{var}} \times n_{\text{time}}}$ be a climate tensor to be projected onto $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of rank-k and with appropriate dimensions, we ultimately want a projection-based index timeseries $v \in \mathbb{R}^{n_{\text{time}}}$. As such, we choose the basis for projection to be the Khatri-Rao product $\mathbf{B} = \mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{n_{\text{space}} n_{\text{var}} \times k}$, and reshape the tensor into a mode-3 unfolding $\mathbf{D}_{(3)} \in \mathbb{R}^{n_{\text{time}} \times n_{\text{space}} n_{\text{var}}}$ so $\mathbf{v} = \left\| \mathbf{P}_{\mathbf{A} \odot \mathbf{B}} \mathbf{D}_{(3)}^T \right\|_{\text{col}}$. We refer to the function $\|\cdot\|_{\text{col}} : \mathbb{R}^{m \times n} \to \mathbb{R}^n$ as the "column norm", which operates by taking the 2-norm of every column in a matrix.

Projections onto a matrix may be computed in a variety of ways, including QR decomposition or pseudo-inverse. Here the pseudo-inverse definition of a projector allows for convenient manipulation of the expression into a form that naturally leverages existing tensor tools. The definition of projection using the pseudo-inverse would in this case give

$$\mathbf{P}_{\mathbf{A} \odot \mathbf{B}} \mathbf{D}_{(3)}^T = (\mathbf{A} \odot \mathbf{B})(\mathbf{A} \odot \mathbf{B})^\dagger \mathbf{D}_{(3)}^T. \tag{2.6}$$

Note that the quantity $(\mathbf{A} \odot \mathbf{B})^\dagger \mathbf{D}_{(3)}^T$ is the solution to one of the three linear least squares subproblems used in the popular alternating least squares algorithm for the solution of 2.4. Specifically, it is the transpose of the solution to the linear least squares subproblem for the third factor matrix. Recognizing that part of the above projection is a component of the CP solve process allows us to use well-tested and optimized routines for the most costly and complicated part of this projection.

2.6. Pipeline for a projection-based climate index. In Algorithm 1 we provide an overview of the basic steps of a pipeline from raw data to climate index. This pipeline uses a projection onto a linear space (hypothesized to represent the phenomena of interest) as a critical step in determining the strength of said phenomena on a particular day.

Many of the steps in Algorithm 1 have multiple options for their implementation, each with their own nuances, but the basic inputs and outputs have largely the same meaning. Therefore, we dedicate a small discussion to each of the four steps below, as well as typical choices for our specific investigation.

Algorithm 1 Climate index pipeline

// — Offline —
$D_{\text{pre}} \leftarrow PREPROCESS(D_{\text{raw}})$
$D_{\text{approx}} \leftarrow APPROX(D_{\text{pre}})$
// — Online —
$D_{\text{pre,daily}} \leftarrow PREPROCESS(D_{\text{raw,daily}})$
$D_{\text{proj,daily}} \leftarrow PROJECT(D_{\text{approx}}, D_{\text{pre,daily}})$ // project daily data onto approximation
$v_{\text{daily}} \leftarrow NORM(D_{\text{proj,daily}})$

***PREPROCESS*()**

This function preprocesses raw climate data. Example steps include: removal of leading (sub)annual harmonics, subtraction of the previous N-day average; averaging over latitudes. Note that since this function works on a day-by-day basis it can just as well be applied to a large concatenation of data (in the offline stage) and a single daily data point (in the online stage).

***APPROX*()**

This function computes an approximation of the input whose general purpose is to capture dominant features. The present work uses truncated PCA, CP decomposition, or

a new affine subspace algorithm detailed in Section 5. Often these methods are applied to (temporal) mean-centered data, and the removal of the mean could just as well belong to $PREPROCESS()$. Another process that may be applied is a subselection of the features captured by the approximation. For example, a rank-15 PCA might be computed, but only 2 features (not necessarily the first/dominant 2) might be returned by $APPROX()$ after some combinatorial search.

***PROJECT*()**

This function projects daily data onto a non-temporal linear subspace implied by the approximation of the data from the previous step. By non-temporal we mean that all modes of the decomposition contribute, except for time. For example, if $APPROX()$ is computed using a rank-k PCA of the data the output would include two factors of shapes $(n_{\text{space}}n_{\text{var}}, r)$ and (n_{time}, k). The column space of the first is then the linear subspace onto which daily data will be projected. Alternatively, if $APPROX()$ employs a rank-k CP approximation of the input data tensor then then the factor matrices for space and variables would together form the column space by a process described later.

***NORM*()**

This function computes the norm of the input, typically the column norm, 2-norm, or Frobenius norm.

2.7. Aligning climate indices. Often two climate indices for the same phenomenon may not be directly comparable. For example, they may have different scales, thresholds, or offsets from 0. Therefore, we propose a deterministic scheme to align any timeseries with another timeseries which has a natural threshold. Let $\mathbf{v}_{\text{ref}} \in \mathbb{R}^{n_{\text{time}}}$ be a reference timeseries with threshold $b_{\text{ref}} \in \mathbb{R}$, and suppose we wish to compute the appropriate threshold b for some other timeseries $\mathbf{v}$. Leveraging the threshold of the reference timeseries we find b by grid searching over thresholds to find the one which maximizes the Peirce skill score between the reference binary classification implied by b_{ref} and the trial classification implied by using b as the threshold for $\mathbf{v}$. Although in practice we often have a continuous reference timeseries $\mathbf{v}_{\text{ref}}$ and its threshold, this approach only needs a binary reference. Further operations like shifting and scaling the timeseries to provide a good visual match are trivial, since any affine transformation applied to the entire timeseries may also be applied to the computed threshold without changing its classification.

3. Case study: Madden-Julian Oscillation. To analyze the performance of different approaches to compute climate indices, we examine the Madden-Julian oscillation (MJO) as a case study. The MJO is an eastward propagating atmospheric phenomenon located near to the equator, and can influence extreme events like monsoons and cyclones, as well as local weather like precipitation [9]. Because the MJO is also characterized by convective, circulatory patterns and precipitation, some or all of the following variables are often employed to investigate MJO: east-west windspeed at an 850 hPa isobar (lower troposphere), east-west windspeed at a 200 hPa (upper troposphere), outgoing longwave radiation. Here, these variables are represented on a rectangular longitude-latitude grid capturing the equatorial region of Earth, $[-15°, 15°] \times [0°, 360°]$. The measurement of windspeeds at different altitudes implicitly introduces some altitudinal dependence, even though the two quantities are treated as variables. Specifically, this data is an assimilation product of measurements to simulations from the Modern-Era Retrospective analysis for Research and Applications 2 (MERRA-2), during years 1980-2022 [3].

The state of the art method for real-time MJO index generation is that of Wheeler and Hendon [8], who suggest using a rank-2 PCA to represent a linear subspace which including all three physical variables. Their approach first preprocesses the data by removing annual and seasonal harmonics as well as lower frequency components like ENSO, all phenomena

assumed to be "more dominant" than MJO, then latitudinally averaging. The first step is intended to remove dominant signals relating to seasonal variability, the second is intended to remove ENSO, while the third reduces computational cost by a reduction in dimension. After extracting this subspace, the MJO index is generated by projection of similarly preprocessed data onto the spatial PCA components. Due to unique properties of PCA, it is possible to construct an index with a natural threshold of 1. For the purposes of generating $\mathbf{v}_{\text{ref}}$ we latitudinally average the data, but we otherwise forgo the average across latitudes since it is both interesting and computationally feasible with current hardware. Absent a "true" MJO index we will compare to this $\mathbf{v}_{\text{ref}}$, as the Wheeler and Hendon approach has been standard and successful in the climate and forecasting community for over a decade. MJO indices computed using their methodology are reported by various agencies including the National Oceanic and Atmospheric Administration.

4. Tensor method. In Figure 4.1 we show the resulting indices of the CP based approach for multiple levels of data preprocessing, using all three variables like Wheeler and Hendon. For comparison, we also plot two indices generated using the Wheeler and Hendon rank-2 PCA approach (one with and without latitudinal averaging). The original Wheeler and Hendon approach of removing harmonics, high-pass filtering the data, and latitudinally averaging is used to obtain our reference index $\mathbf{v}_{\text{ref}}$, which is in turn used to threshold and align the other timeseries. For each non-reference index, the Peirce skill score is computed with $\mathbf{v}_{\text{ref}}$ as a rough measure of performance.

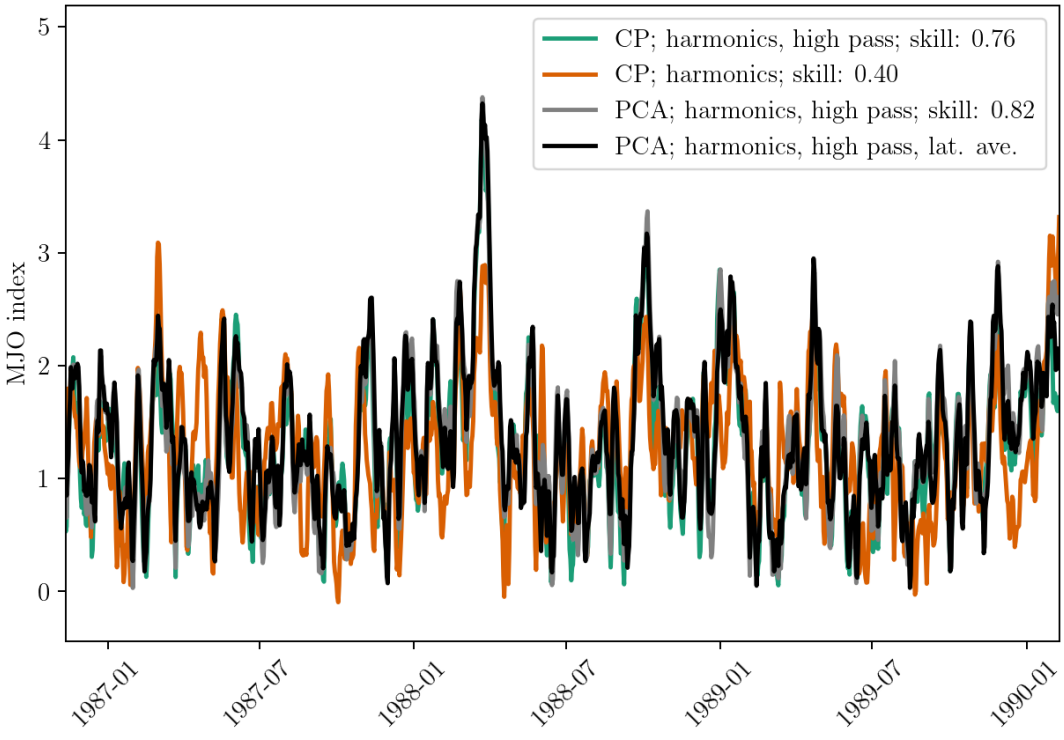


FIG. 4.1. *MJO indices obtained by a greedy CP based approach compared the Wheeler and Hendon PCA approach, each for a variety of different preprocessing amounts. The black curve is computed essentially according to the original scheme of Wheeler and Hendon, and is the reference used to threshold and align the other timeseries. The reported score is the Peirce skill score as compared to the reference data.*

A first conclusion unrelated to the CP results is that removing the latitudinal averaging step does not appear to significantly degrade the performance of the Wheeler and Hendon approach. That is, the grey and black index in 4.1 agree quite well to the eye, and have high Peirce score. This would suggest that future efforts studying MJO should not latitudinally average the data unless absolutely necessary, since this can remove information like North-South shift that climatologists often try to add back in other ways.

The teal curve represents a rank-2 greedy CP approach applied to data which was preprocessed by removing harmonics and then high pass filtering. In particular, it is the best (in terms of Peirce score) subset of 4 or less components from a rank-15 greedy decomposition; which in this case is the first and second components. Since the point of preprocessing the data this way is to make it dominated by MJO, it is not entirely surprising that these 2 components form the best subset. Perhaps more surprising is the fact that the Kronecker structured linear subspace formed by the spatial and variable CP factor matrices captures the arbitrary linear subspace formed by the PCA. This suggests empirically that although the original data may not be outer product structured as a whole, the leading subspace of this processed data is approximately Kronecker structured. Such a study is mathematically interesting in its own right, and could lead to non-trivial upper bounds on the error incurred by using CP instead of SVD.

Likewise, the orange curve is a rank-2 greedy CP approach applied to data which was preprocessed by removing harmonics. The same subselection procedure as above is applied, and again the best subset is the first and second components. However, because the data is no longer dominated by the MJO, it is unclear how much these first two components capture the MJO as compared to whatever phenomena dominate the data, perhaps ENSO. The relatively small amount of an MJO signal in the first two components could explain the large drop in performance on the less processed data.

Finally, we attempt to explain why the CP approach seems to perform adequately for the highly preprocessed data, but much worse for less preprocessed data. Some degradation is expected since the data is fundamentally different, but we suspect there are important theoretical concerns at play here too. As emphasized above, the lack of outer product structure in climate data tensors means that we cannot assume that the phenomenon of interest (here MJO) will be cleanly represented by a small number of components in a low-rank greedy or simultaneous CP decomposition. Simply stated, the components of the low-rank decomposition are those which approximate the data tensor best, and there is no known theory for this setting suggesting that different phenomena in the data tensor are traceable to (sums of) different components. Instead, every component may contribute a little to the reconstruction of the phenomenon of interest, making it impossible to select a small number of representative components for the phenomenon. Why then does the CP approach seem to work well when the data is highly preprocessed? When the data tensor is dominated by MJO the first few components of a low-rank decomposition are likely contribute almost entirely to MJO.

5. A new approach for subspace extraction. The approach to finding an optimal decomposition (or really linear subspace) thus far has been to compute a decomposition of the data minimizing the residual/reconstruction error, then choose the subset of components achieving the optimal classification accuracy with some reference classification. From the second step, there is some level of implicit "trust" in the reference classification. If one trusts this classification, then another approach is viable: find a subspace which best recovers the reference classification when data is projected onto it. In particular we will search for an affine subspace $A = \{\mathbf{v} \in \mathbb{R}^m : \exists \mathbf{c} : \mathbf{v} = \mathbf{B}\mathbf{c} + \mathbf{b}\}$, where $\mathbf{B} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are understood to be fixed quantities.

This approach has a few benefits:

- gives a direct and simple representation of a subspace which best captures the phenomenon;
- by nature of the previous point, it provides an approximate way to map an index generated by a large amount of preprocessing to less preprocessed data;
- no need for expensive combinatorial searches for components of a CP or PCA decomposition;
- the pipeline from 2.6 still applies, $APPROX()$ will find an optimal affine subspace and $PROJECT()$ will project onto it.

In addition, it has a few limitations:

- phenomena may reside on manifolds more complicated than affine; for example, even if the phenomena is perfectly affine for preprocessed data, the preprocessing steps are not necessarily linear nor invertible (e.g. latitudinal averaging);
- requires the solution of a nonlinear manifold-constrained optimization problem; this is feasible but often more expensive than PCA or even CP via alternating least squares;
- more difficult to leverage a discrete $\mathbf{v}_{\text{ref}}$ due to the nature of the optimization problem (we attempt to mitigate this);
- while it is possible to obtain an orthogonal basis for the "linear part" of the affine subspace, such a basis is not unique.

Note that the first of these limitations is shared by PCA and CP, since they will essentially approximate the spatial part of the data with a linear subspace.

5.1. Problem formulation and solution. The basic problem to be solved in $APPROX()$ is finding the best affine subspace as measured by some similarity measure on the index, that is

$$\min_{A} \quad d(\|\mathbf{P}_A \mathbf{D}\|_{\text{col}}, \mathbf{v}_{\text{ref}}), \tag{5.1}$$

where A an affine subspace of dimension k, $\mathbf{P}_A$ is the projector onto A, and $\mathbf{D} \in \mathbb{R}^{n_{\text{space}} n_{\text{var}} \times n_{\text{time}}}$ is the flattened data tensor, and d is some similarity measure between two continuous time-series (not necessarily a proper metric). Choices for d include norms like the 2 norm or the Pearson correlation coefficient. It is generally preferable to to choose similarity measures that are "nearly binary", meaning that they rely less on the continuous values of $\mathbf{v}_{\text{ref}}$ and as much as possible on the binary classification it implies. However, to have a smooth objective function, some level of continuity is necessary for d.

The formulation above is relatively abstract, as optimization over the space of affine subspaces is a somewhat nonstandard task. One approach to turning this into a standard optimization problem is to parameterize the affine subspace as a linear subspace and a shift $A = \{\mathbf{v} \in \mathbb{R}^m : \exists \mathbf{c} : \mathbf{v} = \mathbf{U}\mathbf{c} + \mathbf{b}\}$. Here $\mathbf{U} \in V_k(\mathbb{R}^m)$ is an orthogonal matrix (V_k is the Stiefel manifold, rank-k orthonormal matrices) parameterizing the linear part and $\mathbf{b} \in \mathbb{R}^m$ is a vector parameterizing the shift. Writing $\mathbf{P}_A$ in terms of $\mathbf{U}$ and $\mathbf{b}$ we obtain the equivalent problem

$$\min_{\mathbf{U} \in V_k(\mathbb{R}^m), \mathbf{b} \in \mathbb{R}^m} \quad d(\left\|\mathbf{U}\mathbf{U}^T(\mathbf{D} - \mathbf{b})\right\|_{\text{col}}, \mathbf{v}_{\text{ref}}), \tag{5.2}$$

where we have explicitly written out the projection onto the affine space A using $\mathbf{U}$ and $\mathbf{b}$. Note that the solution to any problem of the form 5.2 is non-unique in $\mathbf{U}$ since $\mathbf{U}\mathbf{M}$ for some square orthonormal $\mathbf{M}$ achieves the same residual. The proposed approach works well, but a more careful implementation would instead optimize over the Grassmannian (space of linear subspaces) rather than $V_k(\mathbb{R}^m)$.

Before giving the full form of the optimization problem used for our experiments, we will comment on the general approach to solve optimization problems with the orthogonal matrix constraint. For the above problem, the iterates and solution must lie on the product manifold $V_k(\mathbb{R}^m) \times \mathbb{R}^m$, which we achieve by using a manifold optimization package, Pymanopt. Pymanopt handles the computation of appropriate differential geometric gradients and offers multiple manifold constrained optimizers [7]. For simplicity, we use gradient descent with derivatives calculated by automatic differentiation.

Finally, the actual objective function used for the rest of the experiments is

$$\min_{\mathbf{U}\in V_k(\mathbb{R}^m),\mathbf{b}\in\mathbb{R}^m,l\in\mathbb{R}} \quad -r\left(\sigma(3\mathbf{p}(\left\|\mathbf{U}\mathbf{U}^T(\mathbf{D}-\mathbf{b})+\mathbf{b}\right\|_{\text{col}}-l),\sigma(3\mathbf{p}(\mathbf{v}_{\text{ref}}-l_{\text{ref}}))\right), \tag{5.3}$$

where r is the Pearson correlation coefficient and the function $\mathbf{p}:\mathbb{R}^n\to\mathbb{R}^n$ is defined as $\mathbf{p}(\mathbf{v}) = \frac{\mathbf{v}}{\max(\mathbf{v})-\min(\mathbf{v})}$, essentially scaling the input to have peak-to-peak spread of 1. A timeseries is first shifted by its respective threshold (after which threshold is 0), then scaled by its peak-to-peak value (after which peak-to-peak is 1), then a scaled sigmoid function is applied in order to make the index "more binary". After processing both the candidate and reference timeseries in this way, their similarity is measured using the Pearson correlation coefficient, negated to promote a large correlation value.

5.2. Results. To analyze the performance of the affine approach, we also consider the MJO and use the same climate data. However, we restrict ourselves to the latitudinally averaged data for the sake of computational efficiency, discussed more later. Figure 5.1 shows the resulting indices from the affine approach for multiple levels of data preprocessing. For comparison we also plot the index generated using the Wheeler and Hendon rank-2 PCA approach, including latitudinal averaging. For each non-reference index, the Peirce skill score is computed with $\mathbf{v}_{\text{ref}}$ as a rough measure of performance.

From Figure 5.1, the resulting timeseries outperform the CP approach in terms of Peirce score, and provide a closer visual match. In addition to many theoretical and computational aspects of this problem discussed in the conclusion, more experimentation is needed to determine if this improved performance remains when the data includes latitudes. Such experiments were too costly given the prototype nature of the current implementation.

Separately, it is interesting to note that because the Wheeler and Hendon approach uses PCA to effectively compute a dimension 2 affine subspace (the linear PCA subspace shifted by the data's mean), the affine approach could in theory exactly recover the black curve when the preprocessing is the same (teal). Possibly due to the sigmoidal scaling and or a local minima we do not quite achieve this, although the performance is still quite good.

6. Conclusion and future work. This work focused on projection-based climate indices, and developed and tested two new approaches for computing such indices.

The first approach is based on a low-rank tensor decomposition of the data, and requires only binary reference data for the purpose of index thresholding and alignment. This approach works well on highly preprocessed data, but struggles on less processed data. We attempt to explain this behavior, and relate it to the lack of theoretical results guaranteeing that separate phenomena will be represented by separate components in the decomposition when the data is not appropriately structured.

The second approach seeks an affine subspace which best recovers a reference continuous timeseries, treating a measure of timeseries similarity as the objective function. As such, this approach appears to outperform the tensor based approach for the task of recovering the index, but can also be used for other tasks, like mapping an approximation of a phenomena from one level of preprocessing to another. This method may find use in the generation of climate data decompositions which lie in the raw or unprocessed space,

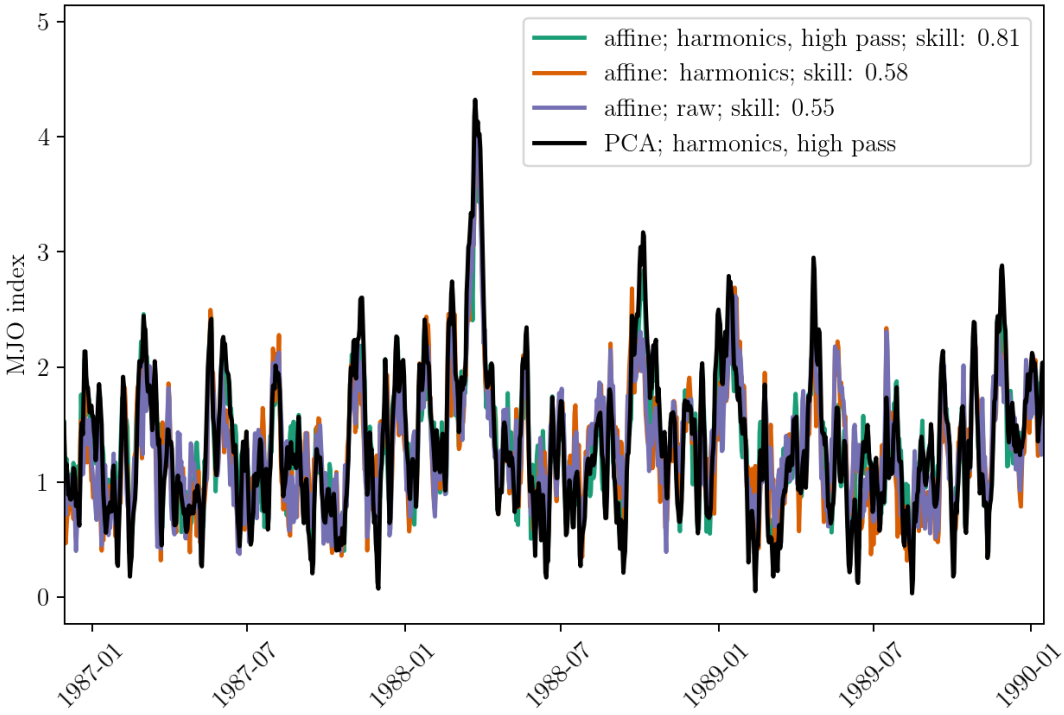


FIG. 5.1. *MJO indices obtained using a rank/dimension 2 affine approach compared the Wheeler and Hendon PCA approach, each for a variety of different preprocessing amounts. The black curve is computed essentially according to the original scheme of Wheeler and Hendon, and is the reference used to threshold and align the other timeseries. The reported score is the Peirce skill score as compared to the reference data.*

enabling direct insight into how the phenomenon of interest couples with the full climate system. The current mathematical formulation of the problem results in a non-unique solution (for which a remedy is provided), but a more thorough study on existence, uniqueness, and special problem structure is of interest. Additionally, new or improved algorithms and implementations for this problem may also be a topic of future work. For $\mathbf{D}$ with $n_{\text{lon}} = 576$, $n_{\text{time}} = 15706$, $n_{\text{var}} = 3$ each iteration of gradient descent takes about 30 seconds. This puts modest data sets, like the inclusion of latitudinal variation, outside the realm of possibility for the current implementation (purely in terms of runtime).

Finally, we reiterate an important climatological observation from our case study on the Madden-Julian oscillation. When using the Wheeler and Hendon-like PCA approach, the removal of latitudinal averaging had a minimal effect on the resulting index. Since this step destroys useful latitudinal information like North-South shifts, we suggest the climate community review the viability of using fully resolved data regardless of their preferred mathematical method.

REFERENCES

[1] P. COMON AND M. RAJIH, *Blind Identification of Under-Determined Mixtures based on the Characteristic Function*, (2005).

[2] V. DE SILVA AND L.-H. LIM, *Tensor Rank and the Ill-Posedness of the Best Low-Rank Approximation Problem*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 1084–1127.

[3] R. GELARO, W. MCCARTY, M. J. SUÁREZ, R. TODLING, A. MOLOD, L. TAKACS, C. A. RANDLES,

A. Darmenov, M. G. Bosilovich, R. Reichle, K. Wargan, L. Coy, R. Cullather, C. Draper, S. Akella, V. Buchard, A. Conaty, A. M. Da Silva, W. Gu, G.-K. Kim, R. Koster, R. Lucchesi, D. Merkova, J. E. Nielsen, G. Partyka, S. Pawson, W. Putman, M. Rienecker, S. D. Schubert, M. Sienkiewicz, and B. Zhao, *The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2)*, Journal of Climate, 30 (2017), pp. 5419–5454.

[4] C. J. Hillar and L.-H. Lim, *Most Tensor Problems Are NP-Hard*, J. ACM, 60 (2013), pp. 45:1–45:39.

[5] T. G. Kolda and B. W. Bader, *Tensor Decompositions and Applications*, SIAM Review, 51 (2009), pp. 455–500.

[6] A. Manzato, *A Note On the Maximum Peirce Skill Score*, Weather and Forecasting, 22 (2007), pp. 1148–1154.

[7] J. Townsend, N. Koep, and S. Weichwald, *Pymanopt: a python toolbox for optimization on manifolds using automatic differentiation*, J. Mach. Learn. Res., 17 (2016), pp. 4755–4759.

[8] M. C. Wheeler and H. H. Hendon, *An All-Season Real-Time Multivariate MJO Index: Development of an Index for Monitoring and Prediction*, Monthly Weather Review, 132 (2004), pp. 1917–1932.

[9] C. Zhang, *Madden-Julian Oscillation*, Reviews of Geophysics, 43 (2005).

[10] Z. Zhang and J. C. Moore, *Empirical Orthogonal Functions*, in Mathematical and Physical Fundamentals of Climate Change, Elsevier, 2015, pp. 161–197.

COMPARING STABILITY OF PARTITIONED HETEROGENEOUS TIME-INTEGRATION METHODS INVOLVING INDEX-2 DAES RESULTING FROM HIGH-ORDER ADAMS-MOULTON AND BACKWARD DIFFERENCE FORMULA TIME INTEGRATION SCHEMES

AMY DE CASTRO* AND PAUL KUBERRY†

Abstract.
We consider an extension of partitioned heterogeneous time integration (PHTI) methods developed in the context of a Hessenberg index-1 DAE to that of a Hessenberg index-2 DAE. The PHTI framework solves for an interface flux term through a dual Schur complement and allows for the coupling of arbitrary multistep or multistage time integrators on each individual subdomain. Although this framework proved successful for the Hessenberg index-1 DAE in [8], the higher index DAE considered here does not enjoy the same stability or accuracy over a broad range of heterogeneous time integrators. We investigate, in particular, the families of Adams-Moulton (AM) and backward difference formula (BDF) schemes and offer an alternative definition of the Lagrange multiplier that improves the performance of the AM methods.

1. Introduction. In our previous work, we have considered the solution of coupled problems sharing a non-overlapping interface through the Implicit Value Recovery (IVR) method [7], as well as the extension of that scheme to the application of reduced order models built from composite reduced bases [5]. The IVR method provides a framework to evaluate well-posedness and the non-singularity of the Schur complement system arising in the scheme. IVR decouples the subdomain equations and equates the temporal stability of the formulation with the stability of the time integrator chosen without requiring additional stability considerations; however, the resulting system is only an index-1 DAE when used with explicit time integrators.

Our goal in this work is to formulate a coupling scheme following the partitioned heterogeneous time-integration (PHTI) methods presented in [8] and then extend the scheme to the use of a reduced order model on at least one subdomain. PHTI methods allow for the coupling of arbitrary implicit or explicit multistage or multistep methods, making them more flexible and practical than IVR as they can utilize optimized discretizations specific to the subdomain physics. Specific advantages of PHTI over other heterogeneous methods include its balancing of the stability and accuracy of couplings based on monolithic formulations with the greater flexibility provided by coupling schemes which lag interface coupling terms. The PHTI approach is centered on two main cores: approximations of the interface flux obtained from an auxiliary monolithic Schur complement system, and a polynomial-in-time approximation of the interface flux over the time coupling window.

In [8], PHTI was applied to an air-sea system coupled by a Robin condition on the interface. The inclusion of the algebraic variable within the Robin constraint results in a differential algebraic equation (DAE) of Hessenberg index-1, which we will also refer to as a semi-explicit index 1 DAE. Much work has been done to extend multistep methods for ODEs to the case of these index-1 DAEs; however, as the index of the DAE increases, one cannot expect the same method to work without adjustment. For example, multistep methods applied directly to a semi-explicit index-1 DAE are stable and convergent to the same order of accuracy for the DAE as for the standard nonstiff ODE, see [2], Section 3.2. The only restrictions to this are that the constraint equation, if solved separately, must be solved with sufficient accuracy, and there may be additional complications if the underlying ODE is stiff. However, for general index-1 DAEs and for Hessenberg index-2 DAEs to attain

*Sandia National Laboratories, agdecas@sandia.gov
†Sandia National Laboratories, pakuber@sandia.gov

order of accuracy greater than two, the coefficients of multistep methods for DAEs must satisfy the order conditions for ODEs as well as an additional set of constraints. Theorem 3.1.1 of [2] states, "The k-step constant stepsize BDF method ($k < 7$) applied to constant coefficient linear DAE systems of index ν is convergent of order $O(h^k)$ after $(\nu - 1)k + 1$ steps." This shows that the coefficients of BDF methods satisfy these additional constraints and achieve higher order accuracy [1, 2].

Other traditional ODE methods, such as Adams-Moulton or Adams-Bashforth, may not necessarily satisfy these additional order conditions; the literature grew out of BDF and Runge-Kutta (RK) schemes for particular classes of DAEs and remains primarily focused on these types of methods. Multistep or RK methods applied to general DAEs do not enjoy universal stability or convergence properties; one can only show results that are dependent on the form of the DAE.

Results for Runge-Kutta methods discuss fully implicit or half-explicit approaches, i.e. Chapter 4 of [2] and Chapter 10 of [1]. Again, there are additional order conditions the coefficients must satisfy, and these methods still often suffer from severe order reduction for higher index DAEs. For Hessenberg index-2 DAEs, suggestions include adjusting the method to include a projection step onto the constraint manifold, which makes the piecewise polynomials approximating both the differential and algebraic variables possibly discontinuous on the time grid points. Using Radau collocation instead of Gauss collocation may work for lower index forms, but even here the algebraic variable is approximated by a piecewise polynomial that may be discontinuous at the mesh points t^k [1].

The problem we have focused on in [5, 7] involves traditional continuity of states and continuity of flux conditions on the interface. This results in a Hessenberg index-2 DAE, and the increased index poses difficulties for the extension of numerical methods. Previously for the IVR method, we have overcome this difficulty by reducing the DAE to Hessenberg index-1 form via differentiation of the continuity of state condition. The original implementation of the PHTI method in [8] was also for an index-1 DAE. Thus, this paper represents our initial work for the solution of the index-2 DAE. The above observations demonstrate that while the PHTI method seems to enjoy convergence and accuracy for a wider range of methods such as Adams-Moulton and explicit Runge-Kutta schemes in [8], its application to a higher index DAE is not straightforward. The increased index requires more care as to which methods may be used for discretization. Analysis for how these methods would perform when combined is not available, and it is unclear whether the foundational assumption that the algebraic variable may be approximated by a continuous-in-time polynomial is valid.

In the rest of this paper, we present our model problem and show its formulation as a Hessenberg index-2 DAE in Section 2. Different multistep time discretizations are developed and compared in Section 3, and the auxiliary monolithic Schur complement formulation is derived. Next, we examine the extension to a multistage method and the corresponding Lagrange interpolating polynomial in Section 4, showing numerical results for different couplings in Section 5 and offering conclusions and suggestions in Section 6.

2. Model problem. We consider the following transmission problem, defined on a bounded domain $\Omega \in \mathbb{R}^d$ for $d = 2, 3$ divided into two non-overlapping subdomains Ω_1, Ω_2 with interface γ:

$$\begin{aligned} \dot{u}_i - \nabla \cdot F_i(u_i) &= f_i && \Omega_i \times (0, T] \\ u_i &= g_i && \text{on } \Gamma_i \times (0, T], \text{ for } i = 1, 2 \\ u_i &= u_{i,0} && \text{on } \Omega_i, \quad t = 0. \end{aligned} \tag{2.1}$$

In the above, for $i = 1, 2$, $\Gamma_i = \partial\Omega_i/\gamma$, $u_i := u_i(\boldsymbol{x}, t)$ is the unknown scalar field, $F_i(u_i) := \kappa_i \nabla u_i - \boldsymbol{a} u_i$ is the total flux function, $\boldsymbol{f_i}, \boldsymbol{g_i}$ are given source and boundary terms,

$u_{i,0}$ is the initial condition, and $\boldsymbol{a} := \boldsymbol{a}(\boldsymbol{x},t), \kappa_i := \kappa_i(\boldsymbol{x},t) > 0$ are the advection field and diffusion coefficients. In contrast to our previous work, which solves these equations by the partitioned Implicit Value Recovery (IVR) scheme [5, 7], we enforce continuity of the states u_i and continuity of the total flux along the interface:

$$u_1(\boldsymbol{x},t) - u_2(\boldsymbol{x},t) = 0 \text{ and } F_1(\boldsymbol{x},t)\cdot\boldsymbol{n}_\gamma - F_2(\boldsymbol{x},t)\cdot\boldsymbol{n}_\gamma = 0 \text{ on } \gamma\times[0,T]. \tag{2.2}$$

Without loss of generality, we may take the interface normal vector $\boldsymbol{n}_\gamma$ to point in the direction of Ω_1; i.e. on the interface, $\boldsymbol{n}_\gamma = \boldsymbol{n}_2$.

For the IVR method, continuity of the velocities is enforced in place of the state continuity condition, as this gives rise to a Hessenberg Index-1 DAE, which is central to the scheme. As we are going to pursue an extension of the PHTI framework [8], we retain the more common interface condition, which results in a Hessenberg Index-2 DAE form. However, we note that in the original PHTI work, a Hessenberg Index-1 DAE was obtained by the enforcement of the Robin condition $F_\gamma := F_1\cdot\boldsymbol{n}_\gamma = F_2\cdot\boldsymbol{n}_\gamma$ and $F_\gamma = \alpha(u_1-u_2)$ on γ.

Let $H^1(\Omega_i)$ denote the Sobolev space of all square integrable functions on Ω_i whose first derivatives are also square integrable, with $H^1_\Gamma(\Omega_i)$ the subspace of $H^1(\Omega_i)$ containing functions that vanish on Γ_i. For ease of notation, we omit the subscript i on Γ, and Γ_i, denoted Γ, should be understood to correspond to the i from Ω_i. The notation $(\cdot,\cdot)_\omega$ denotes the L^2 inner product on a subdomain or interface ω. The trace of $H^1(\Omega_i)$ is represented by $H^{1/2}(\gamma)$, with dual space $H^{-1/2}(\gamma)$ and dual product $\langle\cdot,\cdot\rangle_\gamma$.

The weak form of (2.1) may be derived, where we enforce the continuity of states condition in (2.2) by a Lagrange multiplier (LM):
Find $\{u_1,u_2\}\in H^1_\Gamma(\Omega_1)\times H^1_\Gamma(\Omega_2)$ *such that*

$$\begin{aligned}
(\dot{u}_1,v_1)_{\Omega_1} + (F_1(u_1),\nabla v_1)_{\Omega_1} &= (f_1,v_1)_{\Omega_1} - (F_1(u_1)\cdot\boldsymbol{n}_\gamma,v_1)_\gamma \qquad \forall v_1\in H^1_\Gamma(\Omega_1)\\
(\dot{u}_2,v_2)_{\Omega_2} + (F_2(u_2),\nabla v_2)_{\Omega_2} &= (f_2,v_2)_{\Omega_2} + (F_2(u_2)\cdot\boldsymbol{n}_\gamma,v_2)_\gamma \qquad \forall v_2\in H^1_\Gamma(\Omega_2)\\
\langle u_1-u_2,\mu\rangle_\gamma &= 0 \qquad \forall\mu\in H^{-1/2}(\gamma)
\end{aligned} \tag{2.3}$$

Discretization of (2.3) will yield a monolithic formulation where the subdomains are coupled through the interface flux (2.2) and through the constraint equation, and our goal is to derive a Schur complement equation from this auxiliary monolithic formulation. Before proceeding, we briefly recall the formulas of the families of multistep Backwards Differentiation Formulas (BDF) and Adams-Moulton (AM) methods for an equation $\dot{y} = f(t,y)$:

- BDF(s), order s, with weights $\{w_j\}_{j=0}^s$:

$$y^{n+1} = w_0\Delta t f(t^{n+1},y^{n+1}) + \sum_{j=1}^{s} w_j y^{n+1-j},$$

- AM(s), order $s+1$, with weights $\{w_j\}_{j=0}^s$:

$$y^{n+1} = y^n + \Delta t\sum_{j=0}^{s} w_j f(t^{n+1-j},y^{n+1-j}).$$

We want to use a LM to represent the interface flux terms $(F_i(u_i)\cdot\boldsymbol{n}_\gamma,v_i)_\gamma$ in (2.3), as these quantities are equivalent for $i=1,2$ from (2.2). At this juncture, we must decide whether to consider the LM as a variable representing a single flux value at each time step or as an integral of the flux over a time step with coefficients suggested by the multistep method. The first approach suggests that the LM may be thought of as representing traction forces at

a particular instance in time, and PHTI uses this assumption to construct an approximation of the LM using a Lagrange interpolating polynomial. However, we are without assurance that the LM is indeed continuous in time, and the second option of considering it as purely algebraic may be advantageous, as we will discuss in the following section.

At each time step of a BDF(s) scheme, only the current unknown LM appears in the formulation; as no LMs appear on the right-hand side of the system, these two representations of the LM may differ only in semantics. However, for AM methods, the difference between these two options impacts results. With this in mind, we define two different LMs and formulate the Schur complement system for the auxiliary monolithic system with both methods.

1. **Method 1: LM as flux at a single time step.**
   Define $\widehat{\lambda}^{n+1} \in H^{-1/2}(\gamma)$ as $\widehat{\lambda}^{n+1} := F_2(u_2(t^{n+1})) \cdot \boldsymbol{n}_\gamma$
2. **Method 2: LM as an integral of flux over time.**
   Define $\lambda^{*,n+1} \in H^{-1/2}(\gamma)$ as $\lambda^{*,n+1} := \int_{t^n}^{t^{n+1}} F_2(u_2(t)) \cdot \boldsymbol{n}_\gamma dt$
   $\approx \Delta t \sum_{j=0}^{s} w_j F_2(u_2(t^{n+1-j})) \cdot \boldsymbol{n}_\gamma$, where w_j are the weights for the AM(s) method.

3. Comparison of auxiliary monolithic systems. We discretize (2.3) in time first, enforcing the constraint equation at time t^{n+1}. We will refer to a s-step BDF method as BDF(s) and to a s-step AM method as AM(s)-M1 or AM(s)-M2, where M1 and M2 refer to method 1 and method 2 of defining the LM, given at the end of Section 2.

The subdomain equations for $i = 1, 2$ and for all $v_i \in H^1_\Gamma(\Omega_i)$ are given by the following for BDF(s), AM(s)-M1, AM(s)-M2:

$$
\begin{aligned}
(u_i^{n+1}, v_i)_{\Omega_i} &= \sum_{j=1}^{s} w_j (u_i^{n+1-j}, v_i)_{\Omega_i} + w_0 \Delta t \Big((f_i^{n+1}, v_i)_{\Omega_i} - (F_i(u_i^{n+1}), \nabla v_i)_{\Omega_i} + (-1)^i \langle \widehat{\lambda}^{n+1}, v_i \rangle_\gamma \Big), \\
(u_i^{n+1}, v_i)_{\Omega_i} &= (u_i^n, v_i)_{\Omega_i} + \Delta t \sum_{j=0}^{s} w_j \Big((f_i^{n+1-j}, v_i)_{\Omega_i} - (F_i(u_i^{n+1-j}), \nabla v_i)_{\Omega_i} + (-1)^i \langle \widehat{\lambda}^{n+1-j}, v_i \rangle_\gamma \Big), \\
(u_i^{n+1}, v_i)_{\Omega_i} &= (u_i^n, v_i)_{\Omega_i} + \Delta t \sum_{j=0}^{s} w_j \Big((f_i^{n+1-j}, v_i)_{\Omega_i} - (F_i(u_i^{n+1-j}), \nabla v_i)_{\Omega_i} \Big) + (-1)^i \langle \lambda^{*,n+1}, v_i \rangle_\gamma.
\end{aligned}
\tag{3.1}
$$

Discretizing in space, let M_i, K_i represent the mass and flux matrices for each subdomain, and let G_i represent the algebraic form of the continuity constraint in (2.3) for $i = 1, 2$. Define the matrices

$$
A_0 = \begin{bmatrix} M_1 + \Delta t w_0 K_1 & 0 & c_0 G_1^T \\ 0 & M_2 + \Delta t w_0 K_2 & -c_0 G_2^T \\ G_1 & -G_2 & 0 \end{bmatrix} \qquad A_1 = \begin{bmatrix} M_1 - \Delta t w_1 K_1 & 0 & -B_1^1 \\ 0 & M_2 - \Delta t w_1 K_2 & B_2^1 \\ 0 & 0 & 0 \end{bmatrix}
$$

$$
A_j = \begin{bmatrix} -\Delta t w_j K_1 & 0 & -B_1^j \\ 0 & -\Delta t w_j K_2 & B_2^j \\ 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{f}^k = \begin{bmatrix} \Delta t f_1^k \\ \Delta t f_2^k \\ 0 \end{bmatrix},
$$

where

$$
c_0 = \begin{cases} \Delta t w_0 & \text{BDF, AM(s)-M1} \\ 1 & \text{AM(s)-M2} \end{cases} \qquad \text{and} \qquad B_i^j = \begin{cases} \Delta t w_j G_i^T & \text{AM(s)-M1} \\ 0 & \text{AM(s)-M2.} \end{cases}
$$

α	AM(2)-M1	AM(2)-M2
0	1.716	0.999
1	0.999	0.999
1/200	0.999	0.999

TABLE 3.1

Magnitude of largest eigenvalue for AM(2) discretizations: $\alpha = 0$ corresponds to the transmission problem in (2.1)-(2.2), while $\alpha \neq 0$ represents the Robin condition used in [8].

For BDF(s), the algebraic system becomes:

$$A_0 \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \widehat{\lambda}^{n+1} \end{bmatrix} = \sum_{j=1}^{s} \begin{bmatrix} w_j M_1 & 0 & 0 \\ 0 & w_j M_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1^{n+1-j} \\ u_2^{n+1-j} \\ \widehat{\lambda}^{n+1-j} \end{bmatrix} + w_0 \boldsymbol{f}^{n+1}. \tag{3.2}$$

For AM(s)-M1, the matrix system is:

$$A_0 \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \widehat{\lambda}^{n+1} \end{bmatrix} = A_1 \begin{bmatrix} u_1^{n} \\ u_2^{n} \\ \widehat{\lambda}^{n} \end{bmatrix} + \sum_{j=2}^{s} A_j \begin{bmatrix} u_1^{n+1-j} \\ u_2^{n+1-j} \\ \widehat{\lambda}^{n+1-j} \end{bmatrix} + \sum_{j=0}^{s} w_j \boldsymbol{f}^{n+1-j}, \tag{3.3}$$

whereas for AM(s)-M2 we have:

$$A_0 \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \lambda^{*,n+1} \end{bmatrix} = A_1 \begin{bmatrix} u_1^{n} \\ u_2^{n} \\ \lambda^{*,n} \end{bmatrix} + \sum_{j=2}^{s} A_j \begin{bmatrix} u_1^{n+1-j} \\ u_2^{n+1-j} \\ \lambda^{*,n+1-j} \end{bmatrix} + \sum_{j=0}^{s} w_j \boldsymbol{f}^{n+1-j}. \tag{3.4}$$

A prominent similarity between the BDF and the AM(s)-M2 discretizations is the absence of any previous LM terms on the right-hand sides of the system. Notably, the AM(s)-M1 method differs in this way, and we will show that this affects the stability of the system. Although our notation for the BDF(s) system refers to a LM at each time step, note that no previous $\widehat{\lambda}^{n+1-j}$ appear on the RHS of (3.2). Thus, we can think of the LM for BDF as purely algebraic also, behaving similarly to λ^* in (3.4).

For AM(2)-M1 and AM(2)-M2, we conduct eigenvalue analysis to examine the stability of each method. Discretizing the unit square with $\Delta x = \Delta y = 1/64$, with γ at $x = 0.5$, and setting $\Delta t = 3.37 \times 10^{-3}$, the magnitude, truncated to three decimals, of the largest eigenvalue of the matrix $\begin{bmatrix} A_0 & 0 \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} A_1 & A_2 \\ I & 0 \end{bmatrix}$ is shown in the first row of Table 3.1.

As a further examination, we calculate the eigenvalues of the AM(2) methods when applied to a model problem utilizing the Robin interface condition, which corresponds to the case studied in [8]. For this adjustment, the lower right block of the matrix A_0 contains a scaled interface mass matrix, $\alpha^{-1} M_\gamma$, where α is the parameter from the Robin condition. Including this term improves the stability of the system, as seen in last two rows of Table 3.1.

From this analysis, we observe that a Hessenberg Index-1 DAE, represented by the Robin condition, allows for more leniency in the treatment of the LM. When the index of the system is increased to a Hessenberg Index-2, it seems that Adams-Moulton methods require the LM to represent an integral over time of the interface flux rather than a particular flux value at time t^n. BDF methods avoid this difficulty entirely as the discretization only results in a single LM term.

3.1. Schur complement formulation. We now derive the auxiliary monolithic Schur complement equation used to update the interface flux at each time step. Once the LM is solved for, the two subdomain systems may be updated separately using a different time scheme than the one used to derive the Schur complement, although there may be limitations on what schemes are allowable.

Whether BDF(s), AM(s)-M1, or AM(s)-M2 is used to discretize the monolithic system, notice that each of (3.2)-(3.4) results in a similar algebraic structure. In order to discuss the derivation of the Schur complement for the auxiliary monolithic system in a way that encapsulates the BDF(s) and both AM(s) methods, define the variable

$$g^{n+1} = \begin{cases} \widehat{\lambda}^{n+1} & \text{BDF(s), AM(s)-M1} \\ \lambda^{*,n+1} & \text{AM(s)-M2.} \end{cases}$$

Then (3.2)-(3.4) all result in the structure:

$$A_0 \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ g^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_1}^{n+1} \\ \boldsymbol{s_2}^{n+1} \\ \boldsymbol{s_3}^{n+1} \end{bmatrix}, \tag{3.5}$$

with the vector $[\boldsymbol{s_1}^{n+1}, \boldsymbol{s_2}^{n+1}, \boldsymbol{s_3}^{n+1}]^T$ representing the right-hand side of (3.2), (3.3), or (3.4) accordingly. We partition each subdomain into components corresponding to interior and interface variables, represented by a subscript of 0 and γ. With $W_i = M_i + \Delta t w_0 K_i$ for $i = 1, 2$, and with a slight abuse of notation so that the matrix G_i is taken to act only on the interface variables, the expanded system is:

$$\begin{bmatrix} W_{1,00} & W_{1,0\gamma} & 0 & 0 & 0 \\ W_{1,\gamma 0} & W_{1,\gamma\gamma} & 0 & 0 & c_0 G_1^T \\ 0 & 0 & W_{2,00} & W_{2,0\gamma} & 0 \\ 0 & 0 & W_{2,\gamma 0} & W_{2,\gamma\gamma} & -c_0 G_2^T \\ 0 & G_1 & 0 & -G_2 & 0 \end{bmatrix} \begin{bmatrix} u_{1,0}^{n+1} \\ u_{1,\gamma}^{n+1} \\ u_{2,0}^{n+1} \\ u_{2,\gamma}^{n+1} \\ g^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_{1,0}}^{n+1} \\ \boldsymbol{s_{1,\gamma}}^{n+1} \\ \boldsymbol{s_{2,0}}^{n+1} \\ \boldsymbol{s_{2,\gamma}}^{n+1} \\ \boldsymbol{s_3}^{n+1} \end{bmatrix}. \tag{3.6}$$

Eliminating interior degrees of freedom, define the matrices $P_i = W_{i,\gamma\gamma} - W_{i,\gamma 0}(W_{i,00})^{-1} W_{i,0\gamma}$ and vectors $\widehat{\boldsymbol{s_i}}^{n+1} = \boldsymbol{s_{i,\gamma}}^{n+1} - W_{i,\gamma 0}(W_{i,00})^{-1} \boldsymbol{s_{i,0}}^{n+1}$ for $i = 1, 2$. Then (3.6) becomes

$$\begin{bmatrix} P_1 & 0 & c_0 G_1^T \\ 0 & P_2 & -c_0 G_2^T \\ G_1 & -G_2 & 0 \end{bmatrix} \begin{bmatrix} u_{1,\gamma}^{n+1} \\ u_{2,\gamma}^{n+1} \\ g^{n+1} \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{s}}_{\boldsymbol{1}}^{n+1} \\ \widehat{\boldsymbol{s}}_{\boldsymbol{2}}^{n+1} \\ \boldsymbol{s_3}^{n+1} \end{bmatrix}, \tag{3.7}$$

and we may eliminate the upper 2×2 block for the interface DoFs as well to obtain the auxiliary monolithic Schur complement system:

$$S g^{n+1} = G_1 P_1^{-1} \widehat{\boldsymbol{s}}_{\boldsymbol{1}}^{n+1} - G_2 P_2^{-1} \widehat{\boldsymbol{s}}_{\boldsymbol{2}}^{n+1} - \boldsymbol{s_3}^{n+1}. \tag{3.8}$$

Above, S is the Schur complement matrix $S := c_0(G_1 P_1^{-1} G_1^T + G_2 P_2^{-1} G_2^T)$. Once g^{n+1} is obtained by (3.8), the subdomain problems are effectively decoupled, and may be updated with different time integrators from the scheme used to derive this auxiliary monolithic Schur complement.

4. Multistage methods. Ideally, PHTI would allow us to combine an implicit multistep AM or BDF method for the auxiliary monolithic Schur complement with a different time integrator that is appropriate to update Ω_i, $i = 1, 2$. As a multistage solver such as the

q-stage Runge Kutta (RK) methods are commonly used explicit time integration schemes, we describe the application of the q-stage RK scheme, RK(q), to our subdomain problem, defined by weights $\{a_{jm}, b_j, c_j\}$ for $j = 1, \ldots, s$, $m \leq j$ from a Butcher tableau. For RK schemes, it makes most sense to consider the LM as continuous in time as in method 1 of Section 2, so that we may reasonably query it in the formulation at intermediate stages. Thus, we begin from a subdomain equation of the form $M_i \dot{u}_i = f_i - K_i u_i + (-1)^i G_i^T \widehat{\lambda}$. Discretization with an explicit RK(q) method results in:

$$\begin{aligned} M_i u_i^{n+1} &= M_i u_i^n + \Delta t \sum_{j=1}^{q} b_j k_j(u_i^n, t^n), \quad \text{where} \\ k_j(u_i^n, t^n) &:= f_i(t^n + c_j \Delta t) - K_i \left(u_i^n + \Delta t \sum_{m=1}^{j-1} a_{jm} k_m \right) + (-1)^i G_i^T \widehat{\lambda}(t^n + c_j \Delta t). \end{aligned} \tag{4.1}$$

In order to compute the term $\widehat{\lambda}(t^n + c_j \Delta t)$ at intermediate time steps, we approximate the LM $\widehat{\lambda}$ by its Lagrange interpolating polynomial of degree $s_{aux} - 1$, where s_{aux} is the order of the BDF or AM scheme used to derive the auxiliary monolithic Schur complement. Given solutions $\widehat{\lambda}^{n-k}$ for $k = 1, \ldots, s_{aux} - 1$, we construct the interpolant $L(t)$ as

$$L(t) = \sum_{k=0}^{s_{aux}-1} \ell_k(t) \widehat{\lambda}^{n-k}, \tag{4.2}$$

where $\ell_k(t)$ is the k^{th} Lagrange basis function. We refer the reader to Sections 3 and 4 of [8] for more details. We also note that creating a Lagrange interpolant of the solutions $\lambda^{*,n+1}$ to the AM(s)-M2 scheme does not seem meaningful to combine with RK methods, as one is creating an interpolation of values that are already time integrated. In summary, the PHTI method allows us to solve for an interface flux term by forming an auxiliary monolithic system with an implicit multistep method. Once the flux term is solved for by the Schur complement equation (3.8), each subdomain may be updated independently using a multistep or multistage method of choice. We describe the solution process below for a heterogeneous coupling, and WLOG assume that a multistep solver is used on Ω_1 and a multistage solver on Ω_2; the formulation for multistep or multistage on both subdomains would follow.

1. Choose a multistep method of order s, either BDF(s), AM(s)-M1, or AM(s)-M2, to discretize the monolithic system, following the steps outlined in Section 3.1.
2. Form and solve the auxiliary monolithic Schur complement (3.8). The solution of this equation is $\widehat{\lambda}^{n+1}$ if using BDF(s) or AM(s)-M1, or $\lambda^{*,n+1}$ if using AM(s)-M2.
3. If using a multistage method for a subdomain solver, construct the Lagrange interpolating polynomial $L(t)$ of degree s-1 (4.2).
4. Update Ω_1 using a multistep method of order p:
   - BDF(p): $W_1 u_1^{n+1} = \Delta t w_0 f_1^{n+1} - \Delta t w_0 G_1^T \widehat{\lambda}^{n+1} + \sum_{j=1}^{p} w_j M_1 u_1^{n+1-j}$
   - AM(p): Define $r^{n+1} := M_1 u_1^n + \Delta t w_0 f_1^{n+1} + \sum_{j=1}^{p} \Delta t w_j \left(f_1^{n+1-j} - K_1 u_1^{n+1-j} \right)$
     - For AM(p)-M1, $W_1 u_1^{n+1} = r^{n+1} - \sum_{j=0}^{p} \Delta t w_j G_1^T \widehat{\lambda}^{n+1-j}$
     - For AM(p)-M2, $W_1 u_1^{n+1} = r^{n+1} - G_1^T \lambda^{*,n+1}$
5. Update Ω_2 using a multistage method of order q:

$$k_j(u_2^n, t^n) := f_2(t^n + c_j\Delta t) - K_2\left(u_2^n + \Delta t \sum_{m=1}^{j-1} a_{jm}k_m\right) + G_2^T L(t^n + c_j\Delta t)$$

$$M_2 u_2^{n+1} = M_2 u_2^n + \Delta t \sum_{j=1}^{q} b_j k_j(u_2^n, t^n).$$

Lastly, note that we have assumed that the same time step is used on each subdomain to ease the notation and formulation. If the time step on Ω_i differs from that used to derive the auxiliary monolithic system, the state variable u_i on Ω_i must be evolved until the end of the coupling window by using a Lagrange interpolating polynomial (4.2) to derive the needed values of the LM within the coupling window.

5. Numerical Results. We consider two manufactured solutions which are linear in space and either linear or quadratic in time, referred to as problems 1 and 2, respectively, in Table 5.1. The diffusion coefficients in each subdomain are set to $\kappa_i = 10^{-5}$, and the

Problem (1)	Problem (2)
$u_{\text{exact}}(x, y, t) = t(x + 2y + 3)$	$u_{\text{exact}}(x, y, t) = t^2(x + 2y + 3)$

TABLE 5.1
Manufactured solutions on Ω

advection field is chosen as $\boldsymbol{a} = (0.5 - y, x - 0.5)$. The domains under consideration are $\Omega_1 = [0, 0.5] \times [0, 1]$ and $\Omega_2 = [0.5, 1] \times [0, 1]$, so that the interface γ is the line $x = 0.5$. We impose homogeneous Dirichlet boundaries on all non-interface boundaries Γ_i. Using a spatial mesh of $\Delta x = 1/32$ and $\Delta y = 1/64$ on each subdomain yields 2145 nodes in Ω_i for $i = 1, 2$, and we set the final time to be $T = 3$, with $\Delta t = 3.37 \times 10^{-3}$. In each of the following tables, relative L^2 errors are shown between the computed and exact solutions, where

$$L^2 \text{ error } = \frac{(\sum_{i=1}^{2} ||u_i - u_{\text{exact}}||_{0,\Omega_i}^2)^{1/2}}{||u_{\text{exact}}||_{0,\Omega}}.$$

Note that AM(0) and BDF(1) are both Backward-Euler and AM(1) is the trapezoidal method. We expect AM(s) to exactly capture linear and quadratic solutions for $s \geq 0$ and $s \geq 1$, respectively. Likewise, BDF(s) should be able to represent linear and quadratic solutions exactly for $s \geq 1$ and $s \geq 2$, respectively. We begin by using homogeneous couplings, implementing the same time discretization for the auxiliary monolithic system as we implement on each subdomain Ω_i. In Table 5.2, we observe that each AM(s)-M2 and BDF(s) method is able to accurately capture the linear problem, as expected. The AM(s)-M1 methods, however, only solve the system accurately for $s \leq 1$. Based on the eigenvalue analysis in Section 3, we anticipated that AM(2)-M1 would not be stable, and this behavior is also exhibited by higher order AM(s)-M1 methods.

Table 5.3 shows similar results for the quadratic in time problem. The first order methods do not reach machine precision, but higher order schemes for BDF(s) and AM(s)-M2 accurately capture the quadratic solution. Again, AM(s)-M1 is unstable for $s \geq 2$.

Next, we experiment with coupling AM(s)-M2 methods of different orders for the quadratic in time problem. Table 5.4 shows that these couplings are accurate to machine precision for $s \geq 2$. The use of AM(1)-M2 with any higher order method experiences a loss in accuracy, but combinations of AM(2)-M2, AM(3)-M2, and AM(4)-M2 for the auxiliary

Scheme for aux. monolithic	Scheme on Ω_1	Scheme on Ω_2	L^2 Error
AM(0)-M2	AM(0)-M2	AM(0)-M2	8.4424e-15
AM(1)-M2	AM(1)-M2	AM(1)-M2	9.1996e-15
AM(2)-M2	AM(2)-M2	AM(2)-M2	9.0012e-15
AM(3)-M2	AM(3)-M2	AM(3)-M2	8.8577e-15
AM(4)-M2	AM(4)-M2	AM(4)-M2	9.0032e-15
AM(0)-M1	AM(0)-M1	AM(0)-M1	8.4434e-15
AM(1)-M1	AM(1)-M1	AM(1)-M1	9.0293e-15
AM(2)-M1	AM(2)-M1	AM(2)-M1	Inf
AM(3)-M1	AM(3)-M1	AM(3)-M1	NaN
AM(4)-M1	AM(4)-M1	AM(4)-M1	NaN
BDF(1)	BDF(1)	BDF(1)	8.4105e-15
BDF(2)	BDF(2)	BDF(2)	2.8666e-14
BDF(3)	BDF(3)	BDF(3)	4.1021e-14
BDF(4)	BDF(4)	BDF(4)	1.2004e-14
BDF(5)	BDF(5)	BDF(5)	1.2194e-14

TABLE 5.2
Homogeneous couplings for the linear in time problem (1)

Scheme for aux. monolithic	Scheme on Ω_1	Scheme on Ω_2	L^2 Error
AM(0)-M2	AM(0)-M2	AM(0)-M2	9.8838e-04
AM(1)-M2	AM(1)-M2	AM(1)-M2	7.9203e-15
AM(2)-M2	AM(2)-M2	AM(2)-M2	7.8532e-15
AM(3)-M2	AM(3)-M2	AM(3)-M2	7.7915e-15
AM(4)-M2	AM(4)-M2	AM(4)-M2	7.9485e-15
AM(0)-M1	AM(0)-M1	AM(0)-M1	9.8838e-04
AM(1)-M1	AM(1)-M1	AM(1)-M1	7.9072e-15
AM(2)-M1	AM(2)-M1	AM(2)-M1	Inf
AM(3)-M1	AM(3)-M1	AM(3)-M1	NaN
AM(4)-M1	AM(4)-M1	AM(4)-M1	NaN
BDF(1)	BDF(1)	BDF(1)	9.8838e-04
BDF(2)	BDF(2)	BDF(2)	1.9410e-14
BDF(3)	BDF(3)	BDF(3)	2.8341e-14
BDF(4)	BDF(4)	BDF(4)	1.0399e-14
BDF(5)	BDF(5)	BDF(5)	1.0253e-14

TABLE 5.3
Homogeneous couplings for the quadratic in time problem (2)

monolithic equation and each subdomain seem to work well. We also implement AM(s)-M2 for the auxiliary monolithic with BDF on one or both subdomains, resulting in $O(1)$ errors. Implementing BDF for the auxiliary monolithic with AM-M2 on the subdomains is highly unstable. One example of each of these scenarios is included in Table 5.4.

We now combine multistep and multistage solvers. Using BDF for the auxiliary monolithic and RK for the subdomains, Table 5.5 shows that BDF(1) and BDF(2) work with almost all RK(q) schemes for the linear in time problem, with the exception of BDF(2) and RK(1). BDF(3) and BDF(4) are unstable with any order RK scheme; we restrict ourselves to BDF(3) in the table. For the quadratic in time problem, BDF-RK couplings behave similarly; however, Table 5.6 show that none of the schemes reach machine precision when

Scheme for aux. monolithic	Scheme on Ω_1	Scheme on Ω_2	L^2 Error
AM(2)-M2	AM(1)-M2	AM(1)-M2	6.4204e-09
AM(3)-M2	AM(2)-M2	AM(2)-M2	7.8707e-15
AM(4)-M2	AM(3)-M2	AM(3)-M2	7.7797e-15
AM(1)-M2	AM(2)-M2	AM(2)-M2	6.2517e-09
AM(2)-M2	AM(3)-M2	AM(3)-M2	7.7908e-15
AM(3)-M2	AM(4)-M2	AM(4)-M2	7.8578e-15
AM(4)-M2	AM(2)-M2	AM(2)-M2	7.7380e-15
AM(2)-M2	AM(4)-M2	AM(4)-M2	7.9283e-15
AM(2)-M2	AM(4)-M2	AM(2)-M2	7.8894e-15
AM(2)-M2	BDF(3)	BDF(3)	1.7826
BDF(3)	AM(2)-M2	AM(2)-M2	NaN

TABLE 5.4

Combinations of implicit solvers for the quadratic in time problem (2)

used for the quadratic problem.

Scheme for aux. monolithic	Scheme on Ω_1	Scheme on Ω_2	L^2 Error
BDF(1)	RK(1)	RK(1)	7.9622e-15
BDF(1)	RK(2)	RK(2)	8.2862e-15
BDF(1)	RK(3)	RK(3)	1.1895e-14
BDF(1)	RK(4)	RK(4)	8.4319e-15
BDF(2)	RK(1)	RK(1)	1.2433e+104
BDF(2)	RK(2)	RK(2)	8.2766e-15
BDF(2)	RK(3)	RK(3)	1.1691e-14
BDF(2)	RK(4)	RK(4)	8.3731e-15
BDF(3)	RK(1)	RK(1)	Inf
BDF(3)	RK(2)	RK(2)	1.5764e+98
BDF(3)	RK(3)	RK(3)	1.4245e+98
BDF(3)	RK(4)	RK(4)	1.7185e+98

TABLE 5.5

Discretizations using RK(q) for each subdomain and BDF(s) for the auxiliary monolithic system for the linear in time problem (1)

Lastly, we notice that the use of BDF(s) for the auxiliary monolithic system, RK(q) for Ω_1, and BDF(s) for Ω_2 as seen in Table 5.7 loses accuracy when compared to similar schemes using RK(q) on both subdomains in Table 5.5. Although they show the same stability behavior, schemes that use RK(q) for both subdomains achieve much better accuracy than their counterparts that use BDF(s) on Ω_2. Multistage RK(q) methods for discretizing the subdomains combine better with BDF(s) for the auxiliary monolithic system than with AM(s)-M2, as observed in Table 5.8.

6. Conclusions. Our previous work with the IVR method, as well as the PHTI implementation in [8], have both shown promising results in terms of accuracy and stability for index-1 DAEs. However, our investigations in this paper demonstrate that the extension of the PHTI framework to an index-2 DAE is not straightforward. Care needs to be taken as the index of the DAE is increased, and combinations of multistage and multistep time integrators on subdomains should not be coupled arbitrarily. Additionally, the results for the AM(s)-M1 method suggest that considering the LM as continuous at the points t^n may

Scheme for aux. monolithic	Scheme on Ω_1	Scheme on Ω_2	L^2 Error
BDF(1)	RK(1)	RK(1)	9.8843e-04
BDF(1)	RK(2)	RK(2)	2.0482e-07
BDF(1)	RK(3)	RK(3)	2.0482e-07
BDF(1)	RK(4)	RK(4)	2.0482e-07
BDF(2)	RK(1)	RK(1)	2.3808e+115
BDF(2)	RK(2)	RK(2)	1.9868e-07
BDF(2)	RK(3)	RK(3)	1.9868e-07
BDF(2)	RK(4)	RK(4)	1.9868e-07
BDF(3)	RK(1)	RK(1)	Inf
BDF(3)	RK(2)	RK(2)	6.1426e+105
BDF(3)	RK(3)	RK(3)	6.1426e+105
BDF(3)	RK(4)	RK(4)	Inf

TABLE 5.6

Discretizations using RK(q) for each subdomain and BDF(s) for the auxiliary monolithic system for the quadratic in time problem (2)

Scheme for aux. monolithic	Scheme on Ω_1	Scheme on Ω_2	L^2 Error
BDF(1)	RK(1)	BDF(1)	8.3222e-07
BDF(1)	RK(2)	BDF(1)	6.8516e-07
BDF(1)	RK(3)	BDF(1)	6.8516e-07
BDF(1)	RK(4)	BDF(1)	6.8516e-07
BDF(2)	RK(1)	BDF(2)	1.1841e+96
BDF(2)	RK(2)	BDF(2)	7.8704e-07
BDF(2)	RK(3)	BDF(2)	7.8704e-07
BDF(2)	RK(4)	BDF(2)	7.8704e-07
BDF(3)	RK(1)	BDF(3)	Inf
BDF(3)	RK(2)	BDF(3)	3.4667e+77
BDF(3)	RK(3)	BDF(3)	3.4671e+77
BDF(3)	RK(4)	BDF(3)	3.4657e+77

TABLE 5.7

Discretizations using RK(q) for Ω_1 and BDF(s) for Ω_2 and the auxiliary monolithic system for the linear in time problem (1)

not be a valid assumption.

Further progress might be found via the recent coupling framework that uses discontinuous Galerkin methods in time (DGiT), [4, 3]. The subdomain integrators must first be derived using a representation as piecewise polynomial in time (examples may be found in, *e.g.* [6, 9]). The Lagrange multiplier is given its own polynomial structure, which might not be continuous. In [3], for example, it was shown that one may appeal to the stability analysis to inform the construction of discrete coupling conditions that close the system and determine the flux in a way that enforces the stability. Although this was done using an unconditionally stable integrator with an energy analysis, the same principles could be investigated in the current setting of PHTI methods.

After this proceeding was compiled detailing the work performed as part of the summer intern program, we noted that Method 2 yields a LM that represents a time integrated quantity. We are investigating projecting this time integrated quantity to a flux time-density before passing it to other time integrators, which appears to be a promising direction.

Scheme for aux. monolithic	Scheme on Ω_1	Scheme on Ω_2	L^2 Error
AM(2)-M2	RK(1)	RK(1)	2.5201
AM(2)-M2	RK(2)	RK(2)	2.5227
AM(2)-M2	RK(3)	RK(3)	2.5227
AM(2)-M2	RK(4)	RK(4)	2.5227
AM(3)-M2	RK(1)	RK(1)	2.5276
AM(3)-M2	RK(2)	RK(2)	2.5302
AM(3)-M2	RK(3)	RK(3)	2.5302
AM(3)-M2	RK(4)	RK(4)	2.5302

TABLE 5.8

Discretizations using RK(q) for each subdomain and AM(s)-M2 for the auxiliary monolithic system for the linear in time problem (1)

7. Acknowledgements. The authors would like to thank Jeffrey Connors and Andrew Steyer for their constructive feedback that improved the quality of this manuscript.

REFERENCES

[1] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, Society for Industrial and Applied Mathematics, 1998.

[2] K. Brenan, S. Campbell, and L. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier Science Publishing Co., Inc., 1989.

[3] J. M. Connors, J. Owen, P. Kuberry, and P. Bochev, *A conservative discontinuous-galerkin-in-time (dgit) multirate time integration framework for interface-coupled problems with applications to solid–solid interaction and air–sea models*, Computer Methods in Applied Mechanics and Engineering, 426 (2024), p. 116975.

[4] J. M. Connors and K. C. Sockwell, *A multirate discontinuous-Galerkin-in-time framework for interface-coupled problems*, SIAM Journal on Numerical Analysis, 60 (2022), pp. 2373–2404.

[5] A. de Castro, P. Bochev, P. Kuberry, and I. Tezaur, *Explicit synchronous partitioned scheme for coupled reduced order models based on composite reduced bases*, Computer Methods in Applied Mechanics and Engineering, 417 (2023), p. 116398.

[6] M. Delfour and F. Dubeau, *Discontinuous polynomial approximations in the theory of one-step, hybrid and multistep methods for nonlinear ordinary differential equations*, Mathematics of Computation, 47 (1986), pp. 169–189.

[7] K. Peterson, P. Bochev, and P. Kuberry, *Explicit synchronous partitioned algorithms for interface problems based on Lagrange multipliers*, Computers & Mathematics with Applications, 78 (2019), pp. 459–482.

[8] K. C. Sockwell, P. Bochev, K. Peterson, and P. Kuberry, *Interface flux recovery framework for constructing partitioned heterogeneous time-integration methods*, Numerical Methods for Partial Differential Equations, 39 (2023), pp. 3572–3593.

[9] S. Zhao and G. W. Wei, *A unified discontinuous Galerkin framework for time integration*, Mathematical Methods in the Applied Sciences, 37 (2014), pp. 1042–1071.

BACKWARDS SEQUENTIAL MONTE CARLO FOR EFFICIENT BAYESIAN OPTIMAL EXPERIMENTAL DESIGN

ANDREW CHIN* AND TOMMIE CATANACH†

Abstract. The expected information gain (EIG) is a crucial quantity in Bayesian optimal experimental design (OED), quantifying how useful an experiment is by the amount we expect the posterior to differ from the prior. However, evaluating the EIG can be computationally expensive since it requires the posterior normalizing constant for which analytical solutions are only possible for basic models. A rich literature exists for estimation of this normalizing constant, with sequential Monte Carlo (SMC) approaches being one of the gold standards. In this work, we leverage two idiosyncrasies of OED to improve efficieny of EIG estimation via SMC. The first is that in OED, we simulate the data and thus know the true underlying parameters. Second, we ultimately care about the EIG, not the individual normalizing constants. This lets us create an EIG-specific SMC estimator that starts with a sample from the posterior and tempers *backwards* towards the prior. The method arises from the observation that, in many cases, the Monte Carlo variance of standard SMC estimators for the normalizing constant of a single dataset are significantly lower than the variance of the normalizing constants across datasets; the latter thus contributes the majority of the variance for EIG estimates. This suggests the potential to slightly increase variance while drastically decreasing computation time by reducing the SMC population size and, taking this idea to the extreme, opens the door to unique estimators. We demonstrate our method on a coupled spring-mass system where we observe significant performance improvements.

1. Introduction. Optimal experimental design (OED) is a powerful method for selecting design parameters for experiments that update model uncertainty using observational data. By quantifying the utility U of a certain design d, one can then maximize this utility over all the designs to find the best experiment d^*:

$$d^* = \arg\max_d U(d).$$

In Bayesian settings, we are often interested in the information gain (IG) from an experiment for an unknown parameter θ given the dataset y. This is quantified by the Kullback-Leibler (KL) divergence from the prior $p(\theta)$, which we assume does not depend on d, to the posterior $p(\theta \mid y, d)$ [15]:

$$\mathrm{IG}(y \mid d) = D_{KL}(p(\theta \mid y, d) \parallel p(\theta)) = \int_\theta p(\theta \mid y, d) \log \frac{p(\theta \mid y, d)}{p(\theta)} \, \mathrm{d}\theta.$$

As we do not have access to y before an experiment is run, our utility function is the expected information gain (EIG):

$$\begin{aligned} \mathrm{EIG}(d) &= E_{y|d}[D_{KL}(p(\theta \mid y, d) \parallel p(\theta))] \\ &= \int_y p(y \mid d) \int_\theta p(\theta \mid y, d) \log \frac{p(\theta \mid y, d)}{p(\theta)} \, \mathrm{d}\theta \, \mathrm{d}y \\ &= \int_y \int_\theta p(\theta, y \mid d) \log \frac{p(y \mid \theta, d)}{p(y \mid d)} \, \mathrm{d}\theta \, \mathrm{d}y \end{aligned} \tag{1.1}$$

Henceforth, we omit d from the notation for brevity.

This expectation has no analytical solution outside of the most basic examples, and thus we resort to Monte Carlo integration. This proceeds by drawing a dataset, estimating the information gain, and repeating this many times to obtain an average information gain.

*Johns Hopkins University, achin23@jhu.edu
†Sandia National Laboratories, tacatan@sandia.gov

However, the information gain itself is also generally intractable due to the presence of the posterior normalizing constant $p(y)$, also known as the evidence or marginal likelihood. Thus considerable effort is often placed on estimating $p(y)$ or its log [16], and not all methods are guaranteed to give accurate results. In low dimensions and simple problems, simple nested Monte Carlo based on importance sampling is a common choice [15]. In more complex cases, more stable but expensive Monte Carlo estimators are required. A wide variety of these estimators exist, as $p(y)$ arises in other applications such as Bayesian model selection and free energy estimation [10, 7]. An alternative approach relies on approximate variational methods [6, 13], but these lack the same asymptotic guarantees as Monte Carlo.

Principal among these is a class of estimators based on sequential Monte Carlo (SMC) [4]. SMC starts with a large number n of samples from a known distribution. This group of samples is often referred to as a population of particles. It then evolves this population via importance resampling and Markov Chain Monte Carlo (MCMC) to sample a sequence of intermediate tempered distributions that converge to the posterior. One common sequence is the following power posterior sequence [8], which we index by the increasing sequence of temperatures $t_i, i = 0, \ldots, N$. By convention, $t_0 = 0$ and $t_N = 1$, so at the lowest temperature we recover the prior, and at the highest temperature we recover the posterior.

$$p_{t_i}(\theta \mid y) = \frac{p(y \mid \theta)^{t_i} p(\theta)}{z_t(y)}, \quad z_{t_i}(y) = \int_\theta p(y \mid \theta)^{t_i} p(\theta) \, \mathrm{d}\theta \tag{1.2}$$

At each level, we can compute

$$\begin{aligned} E[p(y \mid \theta)^{\Delta t_i}] &= \int_\theta p(y \mid \theta)^{\Delta t_i} \frac{p(y \mid \theta)^{t_i} p(\theta)}{z_{t_i}} \, \mathrm{d}\theta \\ &= \frac{\int_\theta p(y \mid \theta)^{t_i + \Delta t_i} p(\theta) \, \mathrm{d}\theta}{z_{t_i}} \\ &= \frac{\int_\theta p(y \mid \theta)^{t_{i+1}} p(\theta) \, \mathrm{d}\theta}{z_{t_i}} \\ &= \frac{z_{t_{i+1}}}{z_{t_i}} \end{aligned}$$

where $\Delta t_i = t_{i+1} - t_i$. Taking the product up to the $N-1$ level yields $p(y)$:

$$\begin{aligned} \prod_{i=0}^{N-1} \frac{z_{t_{i+1}}}{z_{t_i}} &= \frac{z_{t_1}}{z_{t_0}} \frac{z_{t_2}}{z_{t_1}} \cdots \frac{z_{t_N}}{z_{t_{N-1}}} \\ &= \frac{z_{t_N}}{z_{t_0}} \\ &= \frac{\int_\theta p(y \mid \theta) p(\theta) \, \mathrm{d}\theta}{\int_\theta p(\theta) \, \mathrm{d}\theta} \\ &= \frac{p(y)}{1}. \end{aligned}$$

This is known as the stepping stone algorithm [18], and in SMC the number of levels and their temperatures can be chosen adaptively [3]. The tempering gradually guides the samples to areas of high posterior density, and leads to some of the most accurate estimates of $p(y)$[7] via the following estimator:

$$\hat{p}(y) = \prod_{i=0}^{N-1} \frac{1}{n} \sum_{j=1}^{n} p(y \mid \theta_{i,j})^{\Delta t_i}$$

where $\theta_{i,j}$ is the jth particle of the ith temperature.

This accuracy, however, comes at the expense of computational efficiency. Often, hundreds, if not thousands, of particles are used, each requiring numerous MCMC steps for each tempering level. This can lead to hundreds of millions of likelihood evaluations to compute the EIG for a single design, quickly becoming infeasible for some models.

We propose a new estimator inspired by SMC but tailored to the OED setting, which hinges on two key observations. The first is that in OED, we require $E[\log p(y)]$, and the variance in $\log p(y)$ across different y often dominates the variance in estimating $\log p(y)$ for a single y when using SMC. We find that we can significantly reduce the number of particles to achieve similar Monte Carlo error but with far less computational cost. The second observation is that because we generate the data y when computing the EIG, we know the true θ that led to the data and can utilize this for sampling. Generating y requires drawing $\theta \sim p(\theta)$ and then $y \sim p(y \mid \theta)$, resulting in the joint sample (θ, y). Instead of discarding θ, we can treat this joint sample as if we drew $y \sim p(y)$ and $\theta \sim p(\theta \mid y)$, giving us a free posterior sample. Our estimator starts with the posterior sample and draws from a sequence of distributions tempered backwards towards the prior. In other words, for the power posterior in Equation 1.2, we start at $t_N = 1$ and decrease towards $t_0 = 0$, opposite of what is traditionally done in SMC. This lead to an algorithm we call *backwards sequential Monte Carlo* that is also robust to overestimating the EIG under poor MCMC mixing. A number of modifications are possible which we highlight for future exploration.

We demonstrate our estimator on a coupled spring-mass system with multimodal posteriors. Not only do we show that traditional SMC estimators can be used with an order of magnitude fewer particles, but also that our backwards estimator provides a further fourfold improvement in computational cost. An additional simulation based on fitting a Johnson-Cook model is included in Appendix A.

2. Backwards sequential Monte Carlo.

2.1. Motivation: balancing variances. Unlike areas like Bayesian model selection, in OED we are not interested with solely estimating $p(y)$; we are interested in $\text{EIG}(d)$, for which individual $\text{IG}(y)$ will depend on their own $p(y)$. Hence for an estimate of the EIG there are two sources of variance. The first is variance in $\text{IG}(y)$ across the datasets y, and the second is variance from its estimate $\hat{\text{IG}}(y)$ for a given y, which we denote $Var(\hat{\text{IG}}(y) \mid y)$. Even if $\text{IG}(y)$ is known in closed form, in which case $Var(\hat{\text{IG}}(y) \mid y) = 0$, an estimate of the EIG can still have high variance if $Var(\text{IG}(y))$ is high. If we assume $Var(\hat{\text{IG}}(y) \mid y)$ is constant across y and $\hat{\text{IG}}$ is unbiased, we can formalize this through the law of total variance:

$$\begin{aligned} Var(\hat{\text{IG}}(y)) &= E[Var(\hat{\text{IG}}(y) \mid y)] + Var(E[\hat{\text{IG}}(y) \mid y]) \\ &= Var(\hat{\text{IG}}(y) \mid y) + Var(\text{IG}(y)) \end{aligned}$$

In our experiments, we find that standard SMC can be overly precise in the sense that $Var(\hat{\text{IG}}(y) \mid y) \ll Var(\text{IG}(y))$. Significant cost is spent on achieving excellent information gain estimates, but this precision is unwarranted in light of the overall variance across datasets. As an example, we demonstrate this on a coupled spring-mass system model that we elaborate further on in Section 3.2. For a fixed design, we draw 100 different y using different θ, and for each we compute 30 estimates $\hat{\text{IG}}(y)$ using an SMC algorithm [3] with 250 particles. The results for 5 such datasets are shown in Table 2.1. We find that the variance of $\hat{\text{IG}}(y)$ are over two orders of magnitude less than the variance of $\text{IG}(y)$ across the 100 datasets, which was 22.4.

In standard nested Monte Carlo, the total variance is a simple function of the inner and outer loops [12], and the variance tradeoff can be balanced appropriately. We apply this idea

Dataset	$\hat{\text{IG}}(y_i)$	$Var(\hat{\text{IG}}(y_i) \mid y_i)$
y_1	20.210	0.081
y_2	12.218	0.079
y_3	21.272	0.067
y_4	13.462	0.044
y_5	16.098	0.056

TABLE 2.1
Comparison of variances of 30 SMC estimates of IG using 250 particles within datasets and values of IG across datasets. The variance of the information gains across 100 datasets was 22.4, though only 5 datasets are shown here.

to SMC by reducing the number of particles. Shown in Table 2.2, we find that on a single y with 150 particles, the variance is almost identical to 250 particles. At 50 and 20 particles, we still have an order of magnitude less variance than $Var(\text{IG}(y))$. Only when we get to 10 particles does the $Var(\hat{\text{IG}}(y) \mid y)$ approximately equal $Var(\text{IG}(y))$. While this trend may not hold for all types of problems, it should be relatively cheap to determine this empirically for a given problem; the cost of ballparking $Var(\hat{\text{IG}}(y) \mid y)$ using a handful of SMC runs is small relative to the entire OED process, which would require hundreds, if not thousands, of SMC runs. It is important to recognize that these few-particle SMC estimators are no longer useful for estimating individual information gains due to their variance and thus could not be applied in other contexts; their utility is unique to calculating EIGs.

# Particles	Estimate of $\text{IG}(y_1)$	$Var(\hat{\text{IG}}(y_1) \mid y_1)$
250	20.210	0.081
150	20.377	0.133
50	20.513	0.570
20	20.399	1.384
10	24.802	25.123

TABLE 2.2
Comparison of $Var(\hat{IG}(y_1) \mid y_1)$ for different numbers of SMC particles.

We can try pushing this further by considering the extreme case of using only a single particle. This seems ill-advised due to the inability to do importance resampling with a single particle, and indeed in practice many attempts are unsuccessful. However, this is mostly a consequence of the algorithm failing to mix, not a failure of estimation using few samples. We consider the slightly different question of whether we can use a single sample, assuming hypothetically we could directly draw from each temperature. To emulate this, we run SMC using 1000 particles to ensure good mixing, but do the information gain estimation using only a single random particle from each temperature. This results in a variance of 14.903. In other words, while using one particle is not feasible in this case, using one sample is. We may need more total datasets to achieve the same Monte Carlo standard error, but this would be a small cost relative to the overall speedup. In the next section, we propose a method to draw these single samples.

2.2. Algorithm. Now we make use of a second unique feature of OED: datasets are simulated when computing the EIG. This means we have access to the true underlying parameter θ^*. Crucially, we can treat joint sample (y, θ^*) as being generated first by drawing y from its marginal, and then θ^* from the posterior. This gives us one free draw from the posterior, and so we can start our sampling from θ^* and consider sampling a sequence

of tempered distributions beginning from the posterior and going to the prior, taking a single sample at each level. This makes sampling easier since we start with a sample from the most difficult distribution and decrease temperature over time, preventing degeneracy issues where our samples end up stuck in low likelihood regions. This also means we are less concerned with multimodality since we are not trying to explore each power posterior thoroughly, only enough to get a single effective sample. We call this scheme backwards SMC.

Since we will be increasing the variance in our estimates, we switch to using the thermodynamic integral to help reduce bias by directly targeting $\log p(y)$ at the expense of needing more tempering levels:

$$\log p(y) = \int_0^1 \int_\theta \log p(y \mid \theta) \frac{p(y \mid \theta)^t p(\theta)}{z(y \mid \theta)} \, \mathrm{d}\theta \, \mathrm{d}t.$$

The sequence of temperatures provide a discretization of t on $[0, 1]$, and we then use Simpson's rule to approximate the outer integral [1]. By interchanging the order of integration, we can also view our estimator as first doing numerical integration with Simpson's rule using a single draw at each temperature, and then averaging over all the integrals. This enables us to estimate Monte Carlo standard errors by computing the variance across the integrals. The thermodynamic integral approach is also valid for SMC, but the number of tempering levels needed for accurate integration tends to be higher than the number of tempering levels needed for good SMC estimates, and the bias is less of an issue when the estimate of $p(y)$ is precise.

The full algorithm is presented in Algorithm 1, and despite only using a sample at each iteration, we will see in Section 3 that it is still effective. The tempering sequence and MCMC kernel are user specified, with further discussion on their selection in Section 3.1.

Algorithm 1 Backwards sequential Monte Carlo

Require: Dataset (θ^*, y), Temperatures $t = \{t_0, \ldots, t_N\}$
Ensure: Output $\hat{\mathrm{IG}}(y)$
1: Initialize $\theta \leftarrow \theta^*$, $i \leftarrow N - 1$, $l \leftarrow Array[\log p(y \mid \theta^*)]$
2: **while** $i \geq 0$ **do**
3: $\quad \theta \leftarrow MCMC(\theta, p_{t_i})$ // Run MCMC on p_{t_i} starting at θ
4: $\quad i \leftarrow i - 1$
5: $\quad l$.append$(\log p(y \mid \theta))$
6: **end while**
7: **return** $\log p(y \mid \theta^*) - Simpson(t, l)$ // Use Simpson's rule for thermodynamic integral.

One downside to not having a large population of samples is that we lose some ability to estimate the covariance of the past temperature levels for use in adapting the MCMC kernel for the current temperature, especially when using random walk Metropolis algorithms. However, because we start at a point of relatively high likelihood and then temper in the backwards direction, poor mixing generally results in expected likelihoods at each temperature to be slightly higher than the true values; lower temperatures normally place more mass in areas where the likelihood is extremely small by nature of being more spread out, but the MCMC may not have time visit these tail regions. Thus our estimates of $\log p(y)$ will be biased upwards, and so IG estimates are biased downwards. This means that even in this suboptimal regime for MCMC, our estimator is not overly optimistic about any experiment. As we will see in Section 3, OED can still be conducted with these biased estimates.

2.3. Extensions. While simply tempering backwards is a novel scheme, we can further augment this by the fact that, as in traditional SMC, we are able to draw directly from the prior distribution. This leads to various extensions of backwards SMC.

The first is to set the MCMC transition kernel to converge to the prior distribution; we know that the optimal proposal distribution at $t = 0$ is the prior itself, so we can use a proposal density which tempers to the prior, for example

$$f(\theta' \mid \theta) = p(\theta')^{1-t} g(\theta' \mid \theta)^t$$

where $g(\theta' \mid \theta)$ is the standard random walk proposal of a Gaussian density centered at the current value θ with some chosen standard deviation. Assuming the prior is Gaussian, f is still a Gaussian density. An alternative that we leave for future work is to use the preconditioned Crank-Nicolson method [5], tempering the step size parameter to converge to the prior.

The second extension is to inform the initialization at each level based on the idea that the next temperature level should place more mass towards where the prior has more mass. Instead of starting θ at the last position of the previous level, we can nudge the starting point of the MCMC towards the prior mean or mode, either by taking a number of steps in that direction, or by searching in that direction until the likelihood ratio of the new point and the existing point reaches some threshold. We call these methods "magnetic" backwards SMC. A primitive version of this method, as well as the tempered proposals, are used for the primary simulation in Section 3.

Another possibility is to run SMC both forward and backwards, since the sample on either end can be drawn directly. For example, for N tempering levels, one run forward SMC from the prior for first $N/2$ levels, and use backwards SMC from the posterior for the last $N/2$ levels. If the MCMC accrues a minor bias at each level, this process could conceivably reduce that bias by reducing the distance of each level from a level where we can draw directly. This bidirectional property can be used for diagnosing the performance of the sampler as well. By comparing the distribution of log likelihoods after tempering to the distribution from direct draws, we can get a sense of how well the samplers have mixed to get to that point. We can also compared the final values output by the forward and backwards directions to see if they are relatively similar. We leave further exploration of this to future work.

3. Simulations.

3.1. Setup. We compare our algorithm to the SMC algorithm of [3], which adaptively chooses the tempering levels and MCMC iterations based on effective sample size calculations. Different numbers of particles are also considered to illustrate the discussion in Section 2.1, with a 250 particle run being considered the gold standard we compare against.

For our backwards estimator, we use a tempering sequence based on previous work [1], with $N = 100$ levels and temperatures $t_i = (i/N)^5$. For the MCMC, a simple random walk Metropolis is used, where the proposal standard deviation is adapted based on the previous temperature's acceptance rate using the feedback controller of [2] to target an ideal acceptance rate of 0.234 [9]. There is a balance between the number of temperature levels and number of MCMC iterations because more temperatures means that the difference between subsequent distributions is smaller; fewer MCMC iterations should then be required to achieve convergence. Assuming the number of levels is sufficiently large to calculate the thermodynamic integral, it is unclear whether, for example, 100 levels of 100 iterations is generally better than 200 levels of 50 iterations. Figure 3.1 demonstrates how the absolute error changes for the model in Section 3.2 as we vary these parameters when compared to a

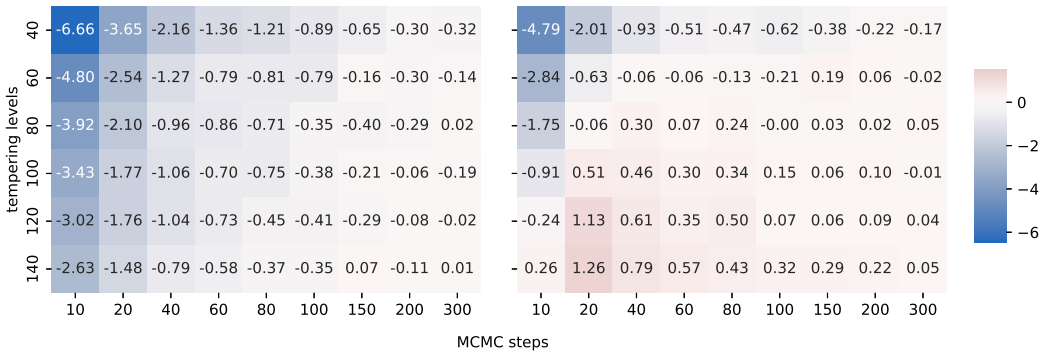


FIG. 3.1. *Left: Absolute error of the standard backwards SMC, with downward bias trend clearly visible when levels or steps are too low. Right: Error of magnetized backwards SMC. Values are in comparison to a long 250 particle SMC estimate, with Monte Carlo standard errors around 0.3.*

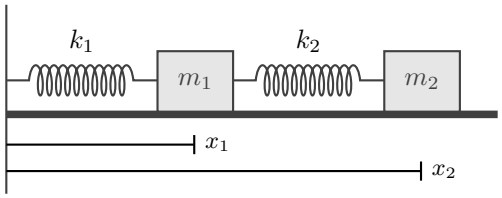


FIG. 3.2. *Coupled spring-mass system. Each of the masses also has an associated friction coefficient* b_1 *and* b_2.

gold standard SMC run, and it appears that there is not a significant difference past a certain threshold. Baed on this, we fix the number of levels at 100 and only adjust the number of MCMC iterations per temperature for simplicity. We implement an early stopping criteria for MCMC at each level once the Spearman correlation between the starting log likelihoods and the current log likelihoods drops below 0.1. We also halve the number of steps taken once the proposal standard deviation equals the prior standard deviation, indicating the power posterior is spread out enough such that more iterations are not critical. These are heuristic adaptations for now, though further investigation in this area would help the robustness of the sampler.

For initialization, we use the magnetized estimator where, between levels, we take up to a tenth of the planned MCMC iterations toward the prior mean using the proposal standard deviation as the step size, stopping if the log likelihood at any new step is less than a tenth of the starting log likelihood. We observe better performance with fewer MCMC iterations across almost all configurations, though overestimation is now more likely.

For tuning, we fix a design and run multiple MCMC iterations, stopping roughly when our estimates stabilize while accounting for Monte Carlo standard error, which ended up being 60 iterations. Any initial SMC runs used to determine whether this sampler is viable as mentioned in Section 2.1 can also be used to inform tuning.

3.2. Coupled spring-mass system. Our model will be a coupled spring-mass system where two masses, m_1 and m_2, are on a surface with respective friction coefficients b_1 and b_2. They are joined together by a spring with spring constant k_2, with the first mass then joined to a fixed point by a second spring with spring constant k_1. The springs are assumed to have length 0 when no forces are applied to them, and the starting positions and velocities

of the masses are set to 0. This setup is outlined in Figure 3.2. Each of the two masses, their friction coefficients, and the two spring constants are considered unknown parameters, so we will have a six dimensional posterior for $(m_1, m_2, b_1, b_2, k_1, k_2)$. All parameters are given log-normal priors with mean 0 and standard deviation 1.

The experiment involves imparting a damped oscillating force on the system representing a vibration, described by the following differential equations:

$$\begin{aligned} x_1' &= v_1 \\ v_1' &= \frac{-b_1 v_1 - (k_1 + k_2)x_1 + k2x_2}{m_1} \\ x_2' &= v_2 \\ v_2' &= \frac{-b_2 v_2 + k_2(x_1 - x_2) + f(\gamma, t)}{m_2} \end{aligned}$$

where $f(\gamma, t) = 5\sin(\gamma t)\exp(-t/5)$ is the forcing function, x_i is the position of the ith mass, and v_i is the velocity of the ith mass.

We observe a noisy version of the position x_1 at 100 equally spaced time points. The noise is Gaussian with mean 0 and standard deviation 0.025, resulting in the observed data $y = \{y_1, \ldots, y_{100}\}$ with

$$y_t = x_{1t} + \epsilon_t, \quad \epsilon_t \overset{iid}{\sim} N(0, 0.025^2), \quad t = 1, \ldots, 100$$

where x_{1t} represents the position of the first mass at time t. Figure 3.3 shows examples of three randomly generated datasets.

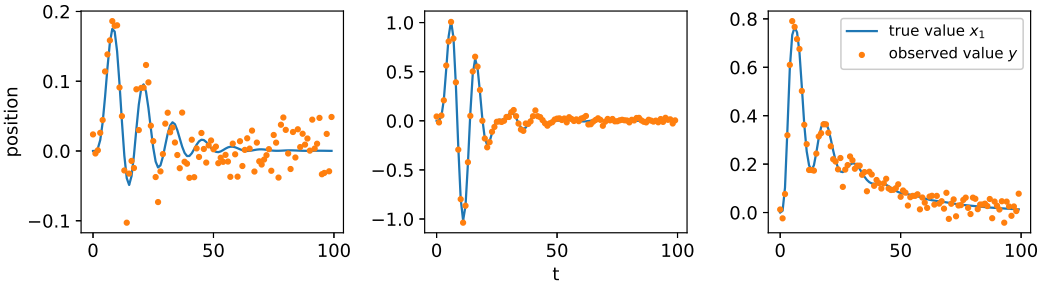


FIG. 3.3. *Three examples of observed data y in orange, along with the true signal x in blue. The true signal corresponds to the position of the first mass m_1 over the 100 time points.*

We consider γ to be the design variable and evaluate ten equally spaced designs between 0 and 2 of the form $0.2j, j = 1, \ldots, 10$. By observing only one mass, we induce multimodality and curvature in the posterior distribution. Figure 3.4 shows four dimensions of an example posterior.

1000 datasets are drawn for each method, with performance measured by the number of likelihood evaluations required and how well the final EIG values correspond to those of an expensive SMC run. The likelihood evaluations are the dominant operation from a computational cost perspective and are therefore used as a hardware agnostic metric for computational efficiency.

3.3. Results. Our first result highlights the discussion from Section 2.1. Shown in Figure 3.5, by decreasing the number of SMC particles by an order of magnitude to 20,

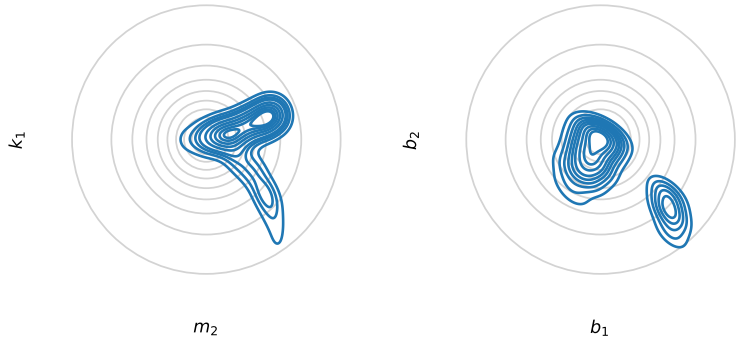


FIG. 3.4. *Example posterior for four of the parameters based on kernel density estimation using samples from SMC, with the standard normal priors shown in grey. The posterior demonstrates multimodality as well as curvature.*

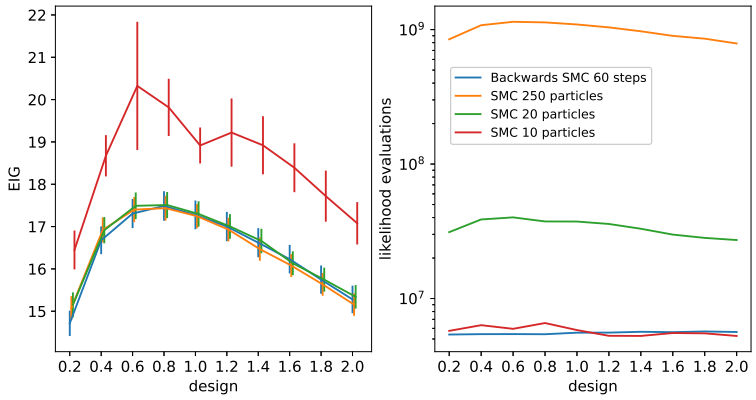


FIG. 3.5. *Left: EIG estimates across 10 designs, with vertical bars representing two Monte Carlo standard errors above and below the estimate. Right: Number of likelihood evaluations used to generate each estimate. The adaptive nature of SMC results in a varying number of evaluations per design.*

we gain an order of magnitude of efficiency with little loss of performance compared to the gold standard 250 particle runs. At 10 particles, significantly more upward bias and noise occurs. Notably, the failure seems to be catastrophic as opposed to gradual; below some threshold, mixing fails and the estimates are poor, while above that threshold the estimates are still stable. Reducing the SMC particles below 10 led to even worse estimates with occasional numerical instability and are not shown. Our backwards estimator performs well but with another fourfold decrease in likelihood evaluations compared to the 20 particle SMC. Standard nested Monte Carlo was ineffective, with Monte Carlo standard errors over 90 for the same computation cost as the 20 particle SMC.

For demonstration, we also run a basic version of our estimator with only 10 MCMC iterations for each temperature and no further adaptivity, shown in Figure 3.6. As expected, this yields underestimates of the EIG. However, the resulting EIG curve can still be used for OED, as the bias is sufficiently similar across designs. The curve is also still relatively smooth due to the lack of degenerate samples increasing the variance. Even if accurate EIG

estimates are required, this could be used to enable a coarse first pass to narrow the search space of designs.

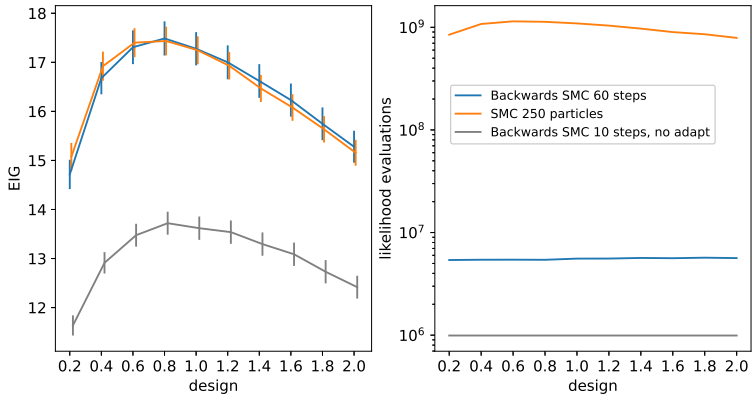


FIG. 3.6. *Comparison of results using 10 iterations of MCMC per temperature for backwards SMC. The EIGs are clearly underestimated but still form a usable curve for OED.*

One caveat is that the backwards SMC has around a 15% higher Monte Carlo standard error, being about 0.167 compared to about 0.145 for SMC. This is not surprising since we are increasing $Var(\hat{\text{IG}}(y) \mid y)$, and so it is reasonable to say that cost of obtaining estimates with the same precision as SMC in this case would require about 33% more iterations than what was ran in our simulations. However, our main takeaways do not change, as an additional 33% cost still represents significant savings over alternatives.

4. Discussion. In this work, we introduced our backwards SMC estimator and demonstrated its computational benefit over traditional SMC estimators. We now present a number of future directions to be explored.

One main downside of backwards SMC is the difficulty in tuning or adapting the number of iterations per level, so additional work here could significantly ease implementation. Since we only have a single chain for each distribution, convergence criteria are hard to establish. One possibility is running multiple chains, though this comes at the expense of computation time and is not clearly better than running the standard SMC. Changing the MCMC transitions kernel to use an algorithm like Hamiltonian Monte Carlo [14] could also be an improvement, both in terms of mixing in high dimensions as well as having the ability to adaptively set trajectory lengths with an algorithm like the No-U-Turn sampler [11].

Another key question is understanding when this estimator can be used. Currently, one would have to rely on a handful of SMC runs to determine whether this estimator might lead to efficiency gains. A more robust method or diagnostic that does not require running other SMC algorithms would be valuable.

Lastly, the use of the free posterior sample is not restricted to SMC. Other techniques for normalizing constant estimation, like parallel tempering [17], could utilize this to speed up convergence of the chains and potentially overcome some of the shortcomings of SMC.

REFERENCES

[1] B. CALDERHEAD AND M. GIROLAMI, *Estimating Bayes factors via thermodynamic integration and population MCMC*, Computational Statistics & Data Analysis, 53 (2009), pp. 4028–4045.

[2] T. A. Catanach, *Computational methods for Bayesian inference in complex systems*, California Institute of Technology, 2017.
[3] T. A. Catanach and J. L. Beck, *Bayesian updating and uncertainty quantification using sequential tempered MCMC with the rank-one modified Metropolis algorithm*, arXiv preprint arXiv:1804.08738, (2018).
[4] N. Chopin, O. Papaspiliopoulos, et al., *An introduction to sequential Monte Carlo*, vol. 4, Springer, 2020.
[5] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White, *MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster*, Statistical Science, 28 (2013), pp. 424 – 446.
[6] A. Foster, M. Jankowiak, E. Bingham, P. Horsfall, Y. W. Teh, T. Rainforth, and N. Goodman, *Variational bayesian optimal experimental design*, Advances in Neural Information Processing Systems, 32 (2019).
[7] M. Fourment, A. F. Magee, C. Whidden, A. Bilge, F. A. Matsen IV, and V. N. Minin, *19 dubious ways to compute the marginal likelihood of a phylogenetic tree topology*, Systematic biology, 69 (2020), pp. 209–220.
[8] N. Friel and A. N. Pettitt, *Marginal likelihood estimation via power posteriors*, Journal of the Royal Statistical Society Series B: Statistical Methodology, 70 (2008), pp. 589–607.
[9] A. Gelman, W. R. Gilks, and G. O. Roberts, *Weak convergence and optimal scaling of random walk metropolis algorithms*, The annals of applied probability, 7 (1997), pp. 110–120.
[10] A. Gelman and X.-L. Meng, *Simulating normalizing constants: From importance sampling to bridge sampling to path sampling*, Statistical science, (1998), pp. 163–185.
[11] M. D. Hoffman, A. Gelman, et al., *The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.*, J. Mach. Learn. Res., 15 (2014), pp. 1593–1623.
[12] X. Huan and Y. M. Marzouk, *Simulation-based optimal Bayesian experimental design for nonlinear systems*, Journal of Computational Physics, 232 (2013), pp. 288–317.
[13] Q. Long, M. Scavino, R. Tempone, and S. Wang, *Fast estimation of expected information gains for Bayesian experimental designs based on Laplace approximations*, Computer Methods in Applied Mechanics and Engineering, 259 (2013), pp. 24–39.
[14] R. M. Neal, *MCMC using Hamiltonian dynamics*, arXiv preprint arXiv:1206.1901, (2012).
[15] T. Rainforth, A. Foster, D. R. Ivanova, and F. Bickford Smith, *Modern Bayesian experimental design*, Statistical Science, 39 (2024), pp. 100–114.
[16] K. J. Ryan, *Estimating expected information gains for experimental designs with application to the random fatigue-limit model*, Journal of Computational and Graphical Statistics, 12 (2003), pp. 585–603.
[17] S. Syed, A. Bouchard-Côté, G. Deligiannidis, and A. Doucet, *Non-reversible parallel tempering: a scalable highly parallel mcmc scheme*, Journal of the Royal Statistical Society Series B: Statistical Methodology, 84 (2022), pp. 321–350.
[18] W. Xie, P. O. Lewis, Y. Fan, L. Kuo, and M.-H. Chen, *Improving marginal likelihood estimation for Bayesian phylogenetic model selection*, Systematic biology, 60 (2011), pp. 150–160.

Appendix A. Johnson-Cook model. As an additional example, we run a Johnson-Cook model of a stress-strain curve for a hypothetical material. The data follows the following model, where the experiment involves observing the stress at 100 equally spaced strain percentages ϵ from 0 to 0.5, with varying measurement noise dependent on whether the material is in the elastic or plastic phase:

$$y = \begin{cases} E\epsilon + \delta_e & \text{if } \epsilon E < A \\ \left(A + B\left(\epsilon - \frac{A}{E}\right)^n\right)\left(1 + C\log\left(\dot{\epsilon}\right)\right)\left(1 - \left(\frac{T-293}{775-293}\right)^m\right) + \delta_p & \text{otherwise} \end{cases}$$

where $\delta_e \overset{iid}{\sim} N(0,1)$ and $\delta_p \overset{iid}{\sim} N(0,10)$. We consider the strain rate $\dot{\epsilon}$ and temperature T as experimental variables. The other parameters represent unknown material constants for which we assign the following priors:

$$\begin{aligned} &E \sim N(73000, 10000^2), \quad A \sim N(350, 100^2) \\ &B \sim N(650, 200^2), \quad n \sim Beta(2,5) \\ &C \sim Beta(2,10), \quad m \sim Beta(2,5) \end{aligned}$$

For simplicity, we evaluate five strain rates $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ for the same temperature 300, and then multiple temperatures $\{300, 400, 500, 600, 700\}$ for the same

strain rate 0.1. For the MCMC, we use a proposal standard deviation equal to the prior standard deviations scaled to target the 0.234 acceptance rate, though no tempering of the proposal to the prior is done since the prior is not multivariate Gaussian. We use the same 1000 datasets with 60 MCMC steps as in the coupled spring-mass model in Section 3.

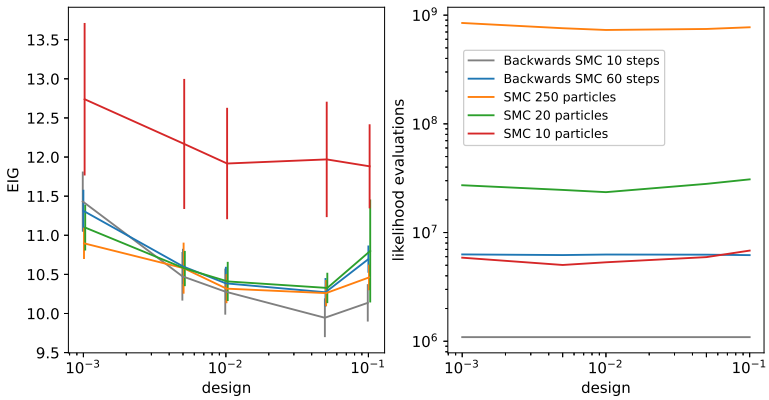


FIG. A.1. *EIG and computational cost results for different strain rates at temperature 300.*

Similar results are achieved as Section 3.3 for the different strain rates, shown in Figure A.1. However, changing the temperatures highlights an advantage of standard SMC, shown in Figure A.2: its adaptivity allows it to use far fewer iterations if they are not needed. There is currently no straightforward method for backwards SMC to do the same, even though as few as 10 iterations yielded good estimates in our testing.

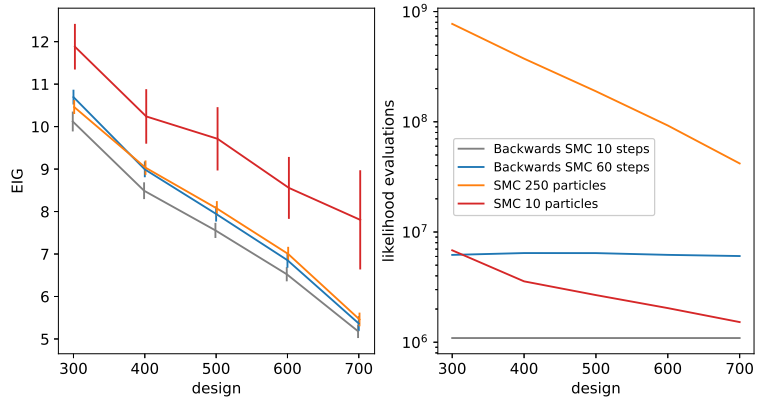


FIG. A.2. *EIG and computational cost results for different temperatures at strain rate 0.1. Here, SMC is able to effectively adapt the number of iterations required at the higher temperatures.*

DATA ASSIMILATION: ADDRESSING SPURIOUS CORRELATIONS AND SCALABILITY ISSUES

ERIC CRISLIP *, MOE KHALIL †, AND KYLE NEAL ‡

Abstract.
Data assimilation in the form of the ensemble Kalman filter (EnKF) is commonly used to combine large-scale physics-based models and real-world observations of a quantity of interest. However, the EnKF is known to be adversely affected by spurious correlations when applied to Numerical Weather Prediction (NWP) models with high-dimensional state spaces. To mitigate the effect of spurious correlations, various localization methods have been proposed in the NWP literature. Unfortunately, the efficacy of and need for these methods in the setting of general PDE-based forward models with large spatial mesh sizes is not well-understood. We conducted a numerical data assimilation experiment on an example PDE model to demonstrate the presence of spurious correlations and need for localization methods outside of the context of NWP. In addition, we compared the computational efficiency of various EnKF implementations in high-dimensional settings for which theoretical complexity bounds are available but empirical costs are unclear.

1. Motivation. Data assimilation is the process in which scientific or engineering models and observations are combined to create estimates of an unkown state that is evolving in time. The ensemble Kalman filter (EnKF) has emerged as the premiere choice of data assimilation technique in the context of expensive and large-scale dynamical systems. One such application area is Numerical Weather Prediction (NWP), where the EnKF has been studied extensively. However, the extension of methods developed in the context of NWP to broader PDE-based systems has been hitherto lacking. The objective of this paper is to investigate the application of the EnKF to general large-scale dynamical systems in two regards: we will examine both the phenomenom of spurious correlations as noted in the NWP literature and the performance of popular algorithms developed to reduce them, as well as compare the empirical computational costs of various EnKF analysis step formulations.

2. Data assimilation.

2.1. Bayesian inference on dynamical systems. Let $\mathbf{u}_t \in \mathbb{R}^p$ denote the unknown state at time t. We model the time-evolution of $\mathbf{u}_t$ by

$$\mathbf{u}_t = \mathcal{M}_t(\mathbf{u}_{t-1}), \tag{2.1}$$

where $\mathcal{M}_t(\cdot)$ is our *forward model* which determines the *evolution distribution* $p(\mathbf{u}_t|\mathbf{u}_{t-1})$. Implicitly, we make a Markovian assumption, where the current state depends only on the state at the previous time-step when conditioning on all previous states. If the initial state $\mathbf{u}_0$ is unknown, we may also define a prior distribution $p(\mathbf{u}_0)$ for it. We model measurements $\mathbf{y}_t \in \mathbb{R}^{q_t}$ of the unknown state by the expression

$$\mathbf{y}_t = \mathcal{H}_t(\mathbf{u}_t) + \boldsymbol{\epsilon}_t, \tag{2.2}$$

where $\mathcal{H}_t(\cdot)$ is the observation operator with measurement noise $\boldsymbol{\epsilon}_t$, which is typically assumed to follow a Gaussian distribution. We will also assume measurements depend only on the current state and are independent of past and future measurements when conditioning on the true state.

The goal of data assimilation is often to obtain the *forecast distribution* $p(\mathbf{u}_t|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})$, which estimates the unknown state at a future time given current knowledge, and the *analysis distribution* $p(\mathbf{u}_t|\mathbf{y}_1, \ldots, \mathbf{y}_t)$, which estimates the unknown state at the time a measurement is collected [12].

*Department of Statistics, The Ohio State University, escrisl@sandia.gov,
†Sandia National Laboratories, mkhalil@sandia.gov,
‡Sandia National Laboratories, kneal@sandia.gov

Under the stated conditional independence assumptions, we can acquire the forecast distribution by integration:

$$p(\mathbf{u}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = \int p(\mathbf{u}_t \mid \mathbf{u}_{t-1}) p(\mathbf{u}_{t-1} \mid \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) d\mathbf{u}_{t-1}. \tag{2.3}$$

The analysis distribution of $\mathbf{u}_t$ can then be computed by Bayes' rule:

$$\begin{aligned} p(\mathbf{u_t}|\mathbf{y}_1, \ldots, \mathbf{y}_t) &= \frac{p(\mathbf{y}_t, \mathbf{u}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_{t-1})}{p(\mathbf{y_t})} && (2.4)\\ &\propto p(\mathbf{y}_t, \mathbf{u}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) && (2.5)\\ &= p(\mathbf{y}_t \mid \mathbf{u}_t) p(\mathbf{u}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) && (2.6) \end{aligned}$$

However, the desired distributions are rarely available in closed form and must be obtained through numerical means.

2.2. Kalman filter. Forecast and analysis distributions are available analytically when the forward model and observation model are both linear and all stochasticitiy is additive Gaussian, i.e.

$$\mathbf{u}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0^f, \mathbf{P}_0^f), \tag{2.7}$$

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{u} + \boldsymbol{\epsilon}_t, \quad \mathbf{H}_t \in \mathbb{R}^{q_t \times p}, \tag{2.8}$$

$$\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t), \tag{2.9}$$

$$\mathcal{M}_t(\mathbf{u}) = \mathbf{M}_t \mathbf{u} + \mathbf{q}_t, \quad \mathbf{M}_t \in \mathbb{R}^{p \times p}, \tag{2.10}$$

$$\mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_t). \tag{2.11}$$

In this setting, our Bayesian prodecure reduces to the *Kalman filter* [8]. When we have a Markovian assumption for the current state and assume measurements are independent when conditioning on the state variable, the analysis distribution is

$$\mathbf{u}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_t^f, \boldsymbol{P}_t^f), \tag{2.12}$$

and the forecast distribution is

$$\mathbf{u}_t \mid \mathbf{y}_1, \ldots, \mathbf{y}_t \sim \mathcal{N}(\boldsymbol{\mu}_t^a, \boldsymbol{P}_t^a), \tag{2.13}$$

where

$$\boldsymbol{\mu}_t^f = \mathbf{M}_t \boldsymbol{\mu}_{t-1}^a, \tag{2.14}$$

$$\mathbf{P}_t^f = \mathbf{M}_t \mathbf{P}_{t-1}^a \mathbf{M}_t' + \boldsymbol{\Lambda}_t, \tag{2.15}$$

$$\mathbf{P}_t^a = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_t^f, \tag{2.16}$$

$$\boldsymbol{\mu}_t^a = \boldsymbol{\mu}_t^f + \mathbf{K}_t (\mathbf{y}_t - \mathbf{H}_t \boldsymbol{\mu}_t^f). \tag{2.17}$$

The matrix $\mathbf{K}_t = \mathbf{P}_t^f \mathbf{H}_t' (\mathbf{H}_t \mathbf{P}_t^f \mathbf{H}_t' + \mathbf{R}_t)^{-1}$ is known as the *Kalman gain*.

2.3. Ensemble Kalman filter. Recall that the goal of data assimilation is to obtain the forecast and analysis distributions for the hidden state. When equations (2.10) and (2.11) do not hold, the analytic expressions of the Kalman filter are invalid and the forecast and analysis distributions must be obtained by other means. A popular approach is to use ensemble-based methods, which approximate the forecast and analysis distributions through a discrete set of representative ensemble members (also known as particles). The *ensemble Kalman filter* (EnKF) is one such method that accomodates nonlinear forward models through the use of particles, yet still utilizes analytic expressions for the analysis update [4].

The EnKF algorithm begins by generating N ensemble members from the prior on the initial state, $\{\mathbf{u}_{0,i}\}_{i=1}^N \overset{iid}{\sim} p(\mathbf{u}_0)$. Evolving these ensemble members in time to the forecast distribution of $\mathbf{u}_1$ is straightforward. We simply pass the ensemble members through the forward model, $\mathbf{u}_{1,i}^f = \mathcal{M}_1(\mathbf{u}_{0,i})$ for $i = 1, \ldots, N$. However, updating our forecast ensemble members for $\mathbf{u}_1$ to the analysis distribution of $\mathbf{u}_1$ is more challenging. The first hurdle is the choice of algorithm. Multiple deterministic updating schemes exist for the EnKF, such as the Square Root Filter [4]. We will utilize a stochastic approach, which updates each ensemble member by

$$\mathbf{u}_{t,i}^a = \mathbf{u}_{t,i}^f + \mathbf{K}_t\big(\mathbf{y}_t + \boldsymbol{\epsilon}_{t,i} - \mathbf{H}_t\mathbf{u}_{t,i}^f\big), \quad i = 1, \ldots, N, \tag{2.18}$$

$$\{\mathbf{e}_{t,i}\}_{i=1}^N \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{R}_t) \tag{2.19}$$

Once our ensemble members have been updated to the analysis distribution of $\mathbf{u}_1$, we can iterate between forecast evolutions and analysis updates for each subsequent timestep.

Though the ability to accomodate nonlinear forward models is a strength of the EnKF, the algorithm still relies on a linear observation operator. If $\mathcal{H}_t$ is nonlinear, one option is to linearize it by taking $\mathbf{H}_t$ to be its Jacobian.

In practice, we do not know the forecast the forecast mean $\boldsymbol{\mu}_t^f$ or covariance $\mathbf{P}_t^f$ and must approximate these moments by the ensemble estimates,

$$\hat{\boldsymbol{\mu}}_f^t = \frac{1}{N}\sum_{i=1}^N \mathbf{u}_{t,i}^f, \tag{2.20}$$

$$\hat{\mathbf{P}}_f^t = \frac{1}{N-1}\sum_{i=1}^N (\mathbf{u}_{t,i}^f - \boldsymbol{\mu}_t^f)(\mathbf{u}_{t,i}^f - \boldsymbol{\mu}_t^f)', \tag{2.21}$$

and use a plug-in estimator for the Kalman gain,

$$\hat{\mathbf{K}}_t = \hat{\mathbf{P}}_t^f\mathbf{H}_t'(\mathbf{H}_t\hat{\mathbf{P}}_t^f\mathbf{H}_t' + \mathbf{R}_t)^{-1}, \tag{2.22}$$

which gives us the analysis update

$$\mathbf{u}_{t,i}^a = \mathbf{u}_{t,i}^f + \hat{\mathbf{K}}_t\big(\mathbf{y}_t + \boldsymbol{\epsilon}_{t,i} - \mathbf{H}_t\mathbf{u}_{t,i}^f\big), \quad i = 1, \ldots, N, \tag{2.23}$$

Under the assumption of the Kalman filter, the forecast ensemble is updated to the analysis ensemble exactly. However, when $\mathcal{M}_t(\cdot)$ is nonlinear, Gaussianity cannot be maintaned even ignoring the Monte Carlo error of the estimated covariance matrix [12]. Instead, the EnKF analysis updates correspond to Bayesian "best linear-in-the observations" theory [11].

3. Localization methods.

3.1. Motivation for localization. As noted, the forecast covariance matrix $\mathbf{P}_t^f$ is typically unkown and must be approximated by the ensemble covariance matrix $\hat{\mathbf{P}}_t^f$. However, $\hat{\mathbf{P}}_t^f$ is rank-deficient whenever $N < p$ and may have high sampling error when the ensemble size N is small, which can cause noisy or unphysical state updates. This is a common setting for the EnKF, as limitations on computational resources often prohibit ensemble sizes greater than 100. In particular, "spurious correlations" are a concern, where nontrivial correlations are estimated where theory suggests none should exist. Spurious correlations are especially apparent when the state space of interest is a large spatial field.

To ameliorate the effects of spurious correlations, a variety of "localization" techniques have been proposed in the literature. These techniques seek to filter the noisy correlation estimates from the matrix $\hat{\mathbf{P}}_t$. As the name "localization" implies, many of these techniques utilize a priori knowledge of the correlation structure of a latent spatial field; namely, that spatially-distant locations should be uncorrelated. However, to accomodate complex correlation structures or state vectors without natural distance metrics, some adaptive "localization" methods have been introduced.

3.2. Distance-based methods. Distance-based localization techniques enjoy wide use in the data assimilation literature, owing to their interpretability, ease of implementation, low computational cost, and ability to incorporate prior domain knowledge. However, they are not without downsides. Distance-based methods generally rely on expert input to determine the choice of length-scales, which may be difficult to specify a-priori, or they must be tuned by potentially expensive procedures. Furthermore, they generally ignore any spatial or temporal non-stationarity in correlation structure, assume a monotonic relationship between distance and correlation strength, and exclude the possibility of genuine long-range dynamics. Still, when the implicit assumptions behind their use are not severely violated, their advantages remain especially attractive.

Many distance-based methods fall into two categories: **B** localization and **R** localization [6].

3.2.1. B localization. Methods for **B** localization act upon the forecast covariance matrix, which is known as the background error matrix in the weather forecasting literature. They typically aim to reduce spurious correlations by modifying the Kalman gain so that

$$\hat{\mathbf{K}}_t = \left[\mathbf{\Gamma} \circ (\hat{\mathbf{P}}_t^f \mathbf{H}_t')\right] \left[\mathbf{\Gamma} \circ (\mathbf{H}_t' \hat{\mathbf{P}}_t^f \mathbf{H}_t) + \mathbf{R}_t\right]^{-1}. \tag{3.1}$$

Here $\circ$ denotes the Schur (entry-wise) product and $\mathbf{\Gamma}$ is a pre-specified correlation matrix that exhibits decay in spatial correlation. A common choice for $\mathbf{\Gamma}$ is the matrix obtained from the compactly-supported Gaspari-Cohn correlation function with half-width parameter c [5, 7],

$$C(r) = \begin{cases} -\frac{1}{4}(\frac{|r|}{c})^5 + \frac{1}{2}(\frac{r}{c})^4 + \frac{5}{8}(\frac{|r|}{c})^3 - \frac{5}{3}(\frac{r}{c})^2 + 1, & 0 \le |r| \le c \\ \frac{1}{12}(\frac{|r|}{c})^5 - \frac{1}{2}(\frac{r}{c})^4 + \frac{5}{8}(\frac{|r|}{c})^3 + \frac{5}{3}(\frac{r}{c})^2 - 5(\frac{|r|}{c}) + 4 - \frac{2}{3}\frac{c}{|r|}, & c < |r| \le 2c \\ 0, & 2c < |r| \end{cases} \tag{3.2}$$

Here the input variable r is generally taken to be Euclidean distance. One benefit of **B** localization is that the Schur product of a positive-definite matrix and a positive-semi-definite matrix is positive-definite whenever the latter has non-zero rows.

3.2.2. R localization. Methods for **R** localization instead act upon the observation variance **R**. They generally inflate the observation variance by a function of physical distance

from the individiual state components, such that the impact of far-away observations on the analysis update equations is limited.

3.3. Adaptive methods. To relax the rigid assumptions implicit in distance-based methods of localization, a variety of adaptive methods have been introduced. These methods generally place emphasis on the magnitude of the estimated forecast correlations, although they may also include some spatial information. While the adaptive nature of these methods is alluring, their statistical efficiency is not well-understood, and their performance may hinge on the choice of hyperparameters that are difficult to interpret. Two popular adaptive methods are the hierarchical filter of Anderson (2006) and the ECO-RAP method of Bishop and Hodyss (2009).

3.3.1. Hierarchical filter. Anderon's hierarchical filter begins by splitting the EnKF ensemble of size N into L sub-ensembles of size $\frac{N}{L}$. At each analysis step, the Kalman gain for each sub-ensemble is computed separately to obtain $\hat{\mathbf{K}}_{t,1}, \ldots, \hat{\mathbf{K}}_{t,L}$. The Kalman gain of each sub-ensemble is shrunken by a measure of disagreement between them. Mathematically, let

$$\kappa_{ij\ell} = \left[\hat{\mathbf{K}}_{t,\ell}\right]_{ij} \tag{3.3}$$

for all $i, j = 1, \ldots, p$ and $\ell = 1, \ldots, L$. The hierarchical filter collects disagreement measures

$$\alpha_{ij} := \underset{\alpha}{\operatorname{argmin}} \sum_{\ell=1}^{L} \sum_{m=1, m\neq\ell}^{L} \left(\kappa_{ij\ell} - \alpha\kappa_{ijm}\right)^2 \tag{3.4}$$

for all $i, j = 1, \ldots, p$. Using the matrix representation $\left[\hat{\boldsymbol{\alpha}}\right]_{ij} = \hat{\alpha}_{ij}$, the analysis update equations 2.23 are then replaced by:

$$\mathbf{u}^a_{t,i} = \mathbf{u}^f_{t,i} + \left[\hat{\boldsymbol{\alpha}} \circ \hat{\mathbf{K}}_t\right]\left[\mathbf{y}_t + \boldsymbol{\epsilon}_{t,i} - \mathbf{H}_t\mathbf{u}^f_{t,i}\right], \quad i = 1, \ldots, N, \tag{3.5}$$

The idea of Anderson's hierarchical filter is that estimates with high variability may not be stable and should not be trusted.

3.3.2. ECO-RAP. Ensemble COrrelations Raised to A Power (ECO-RAP) is another adaptive localization method. The idea behind ECO-RAP is to amplify large correlations and penalize small correlations by exponentiation, then localize the Kalman gain according to the weighted coefficients of the Discrete Cosine Transform applied to the exponentiated correlations. ECO-RAP constructs the correlation matrix

$$\boldsymbol{\Gamma} = (\widetilde{\mathbf{SC}_t^{\odot q}})'(\widetilde{\mathbf{SC}_t^{\odot q}}), \tag{3.6}$$

where the tilde denotes column-wise normalization, the symbol $\odot q$ denotes entry-wise exponentiation by the power q, $\mathbf{S}$ is a smoothing matrix and $\mathbf{C}_t$ is the estimated forecast ensemble correlation matrix for the state vector $\mathbf{u}_t$. We let $\mathbf{S} = \mathbf{DE}$, where $\mathbf{D}, \mathbf{E}$ are $p \times p$ matrices with entries chosen to be

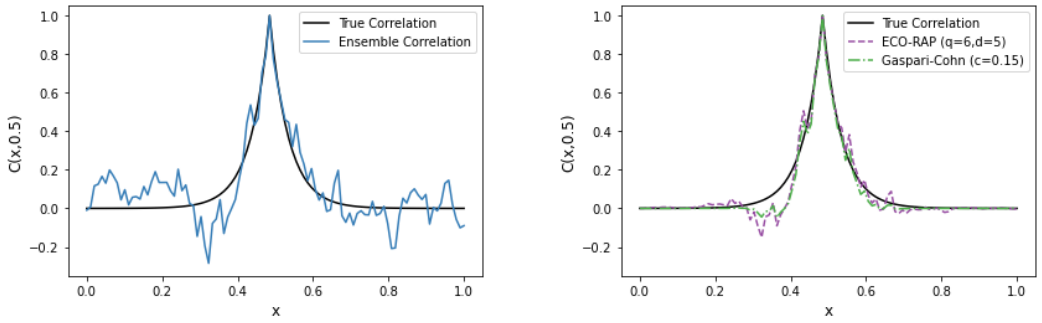
$$\left[\mathbf{D}\right]_{ij} = \begin{cases} e^{-(\frac{i}{d})^2}, & i = j \\ 0, & i \neq j \end{cases} \tag{3.7}$$

$$\left[\mathbf{E}\right]_{ij} = \begin{cases} \frac{1}{\sqrt{p}}, & j = 1 \\ \sqrt{\frac{2}{p}}\cos(\frac{\pi(2i-1)(j-1)}{2p}), & j \neq 1 \end{cases} \tag{3.8}$$

Here d is a parameter to control the degree of spectral smoothing.

4. Numerical experiments for localization.

4.1. Toy problem: Gaussian process with known covariance. We will first illustrate the effects of various localization methods with a simple example where the true correlation function is known analytically. 48 samples were taken of a zero-mean stationary Gaussian process with covariance $C(x, x') = e^{-\frac{|x-x'|}{0.05}}$ over an equally-spaced grid of 101 points on $[0, 1]$. Figure 4.1(a) shows the estimated sample correlations over space as an estimate of $C(x, 0.5)$. Note that estimation is poor for points that are spatially distant. Figure 4.1(b) shows the effect of applying the Gaspari-Cohn and ECO-RAP localization techniques to the correlation estimates. Estimation at spatially distant points appears to be greatly improved, with many spurious correlations shrinking to zero. The hierarchical filter was not considered for this exposition as that method modifies the entire Kalman gain matrix.



(a) Sample correlation function.

(b) Localized correlation function.

Fig. 4.1: Effects of localization on the estimation of a simple correlation function when the data are Gaussian.

4.2. Application to 1-D KPP-Fisher equation: Random diffusitivty. To compare localization methods more comprehensively, another study was conducted using the KPP-Fisher equation in one spatial dimension as a forward model, which is a reaction-diffusion equation of the form

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + u(1-u). \tag{4.1}$$

We modified the equation such that $D = D(x)$ is spatially varying,

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}D(x)\frac{\partial u}{\partial x} + u(1-u). \tag{4.2}$$

This partial differential equation was solved using the Finite Volume method. The state space was defined to be the values at 2500 equally-sized cells over the spatial domain $[0, 1]$. We followed trajectories over the temporal domain $[0, 0.5]$. The initial condition was treated as known and defined to be:

$$u(x, 0) = 1 + 9e^{-800(x-0.2)^2} + 9e^{-800(x-0.8)^2}. \tag{4.3}$$

Measurements were taken with with additive $\mathcal{N}(0, \gamma^2)$ noise at a selected cell with center $x = 0.3002$, hereby referred to as the sensor location. We set $\gamma^2 = 0.00025$ to achieve a reasonable ratio of ensemble member variation to measurement noise level. Measurements

were taken at times $t = 0.1, 0.2, 0.3, 0.4, 0.5$ for a total of five obsevations. Figure 4.2 shows the true ('ground truth') solution as a function of space for four specific time points, while Figure 4.3 illustrates the true solution at the sensor location and an example of measurements taken at that location.

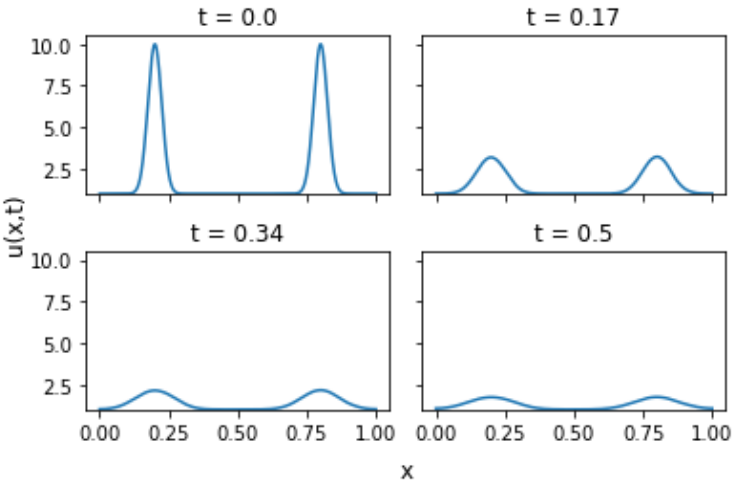


Fig. 4.2: The true solution as a function of space at select times.

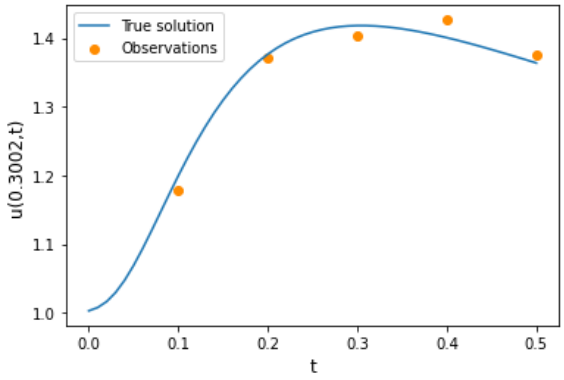
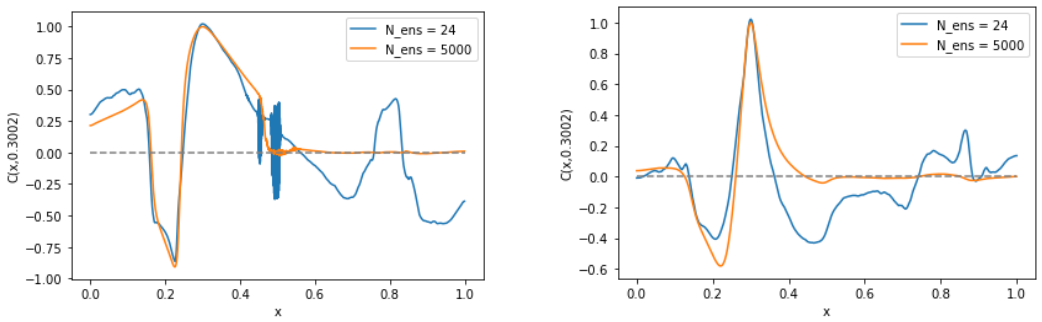


Fig. 4.3: The true solution and noisy measurements at the sensor location for a given run of the EnKF.

To initialize our ensemble members, we considered D unknown and assigned a Gaussian process prior to it, $D(\cdot) \sim \mathcal{GP}\big(m, C\big)$. This prior was set with constant mean function $m(x) = 0.005$ and covariance function $C(x, x') = \sigma^2 e^{|x-x'|/0.1}$ where $\sigma = 0.0005$. Furthermore, we set the true D to be a random realization from this prior. Since estimation of D was not of interest, each ensemble member was associated with a realization of D and the ensemble was treated was marginalized over D. Figure 4.4 provides an example of estimated correlations between the cell at the sensor location and other locations in the state space. The poor estimation of spatially-distant correlations suggest the need for localization.



(a) First timestep.

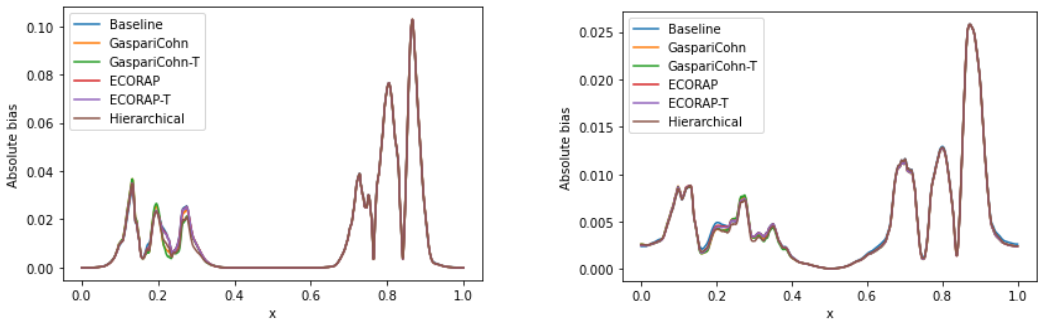
(b) Last timestep.

Fig. 4.4: Estimated correlations with the sensor location.

To assess the performance of localization techniques, we tested the Gaspari-Cohn **B** localization method of [7], the hierarchical filter of [1], and the ECO-RAP method of [2, 3]. Initial guesses for the localization tuning parameters were set to be $c = 0.1$ for the Gaspari-Cohn function and $(q, d) = (6, 5)$ for ECO-RAP. These parameters were then tuned by considering our model as true, generating a synthetic ground truth for comparison, and doing a trial run of the first step of data assimilation. The criterion chosen was the squared deviation from the ensemble mean and ground truth value (i.e. squared bias), which was averaged over space for a scalar metric. To minimize the effects of sampling variability, we repeated the process 5 times, varying the measurement errors, diffusivities, and measurement perturbations, then averaged the result for a final criterion. A grid search was performed over the set of values $0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.25$, and 0.3 for the Gaspari-Cohn half-width parameter and the grid $(q, d) \in \{3, 6, 9\} \times \{4, 5, 6\}$ for the ECO-RAP parameters. The parameters $c^* = 0.05$ and $(q^*, d^*) = (9, 6)$ were selected for all ensemble sizes based off our criterion, which indicates a strong need for localization. The hierarchical filter was set to $K = 4$ sub-ensembles in accordance with the original paper.

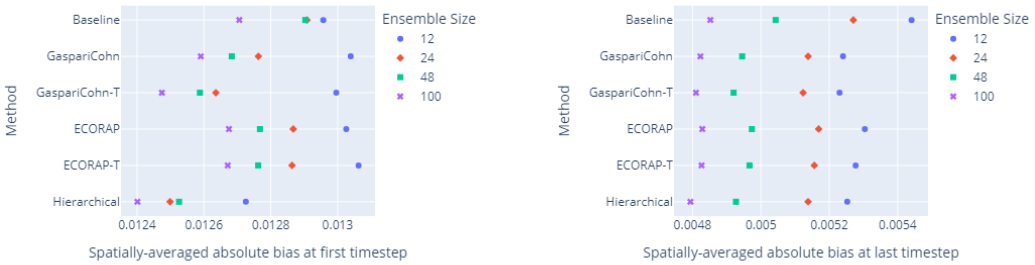
To mitigate the effect of sampling error, the experiment was repeated 50 times, varying the observation noise, ensemble diffusivites, and EnKF measurement perturbations, while keeping the ground truth the same and all tuning parameters fixed. Only 50 repetitions were conducted due to resource constraints. Figure 4.5 shows the absolute bias of the EnKF mean taken as a function of space for $t = 0.1$ and $t = 0.5$, averaged over the different runs of the algorithm. We can see that the bias of the EnKF mean is mostly dominated by the dynamics of system, but there is noticeable variation in performance due to localization in the spatial region $[0.15, 0.3]$.

Figure 4.6 shows the absolute bias averaged over space for a scalar summary of the EnKF performance. Intuition suggests the necessity of localization methods should wane as the ensemble size increases (and thus the estimation of correlations becomes more precise). We see evidence of this in Figure 4.6, as the gap between the baseline and localized EnKF appears to decrease for greater ensemble sizes. In Figure 4.7 we see that estimated variances are uniformly higher when using localization methods, which indicates localization can combat filter divergence. Overall, we can see that all localization methods considered offer improvements of the baseline EnKF in the context of this problem. The hierarchical filter appears to be the preferred choice of localization technique for this problem, with the runner-up being **B** localization with Gaspari-Cohn half-width parameter $c^* = 0.05$.



(a) First analysis step. (b) Last analysis step.

Fig. 4.5: Absolute bias of the EnKF mean averaged over 50 algorithm runs. The ensemble size is fixed at $N = 24$.



(a) After first analysis update. (b) After last analysis update.

Fig. 4.6: Absolute bias of the EnKF mean averaged over space and 50 algorithm runs.

5. Computational efficiency of the EnKF. Multiple methods exist for the computation of the analysis update in (2.18). If we collect the ensemble members and model perturbations into matrices

$$\mathbf{u}_t^a = [\mathbf{u}_{t,1}^a, \ldots, \mathbf{u}_{t,N}^a], \tag{5.1}$$

$$\mathbf{u}_t^f = [\mathbf{u}_{t,1}^f, \ldots, \mathbf{u}_{t,N}^f], \tag{5.2}$$

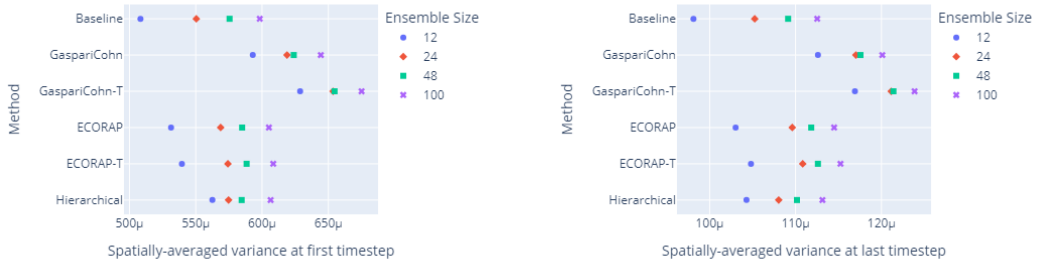
$$\mathbf{Y}_t = [\mathbf{y}_t + \boldsymbol{\epsilon}_{t,1}, \ldots, \mathbf{y}_t + \boldsymbol{\epsilon}_{t,N}], \tag{5.3}$$

then we can write (2.18) as

$$\mathbf{u}_t^a = \mathbf{u}_t^f + \hat{\mathbf{P}}_t^f \mathbf{H}_t' \mathbf{Z}, \tag{5.4}$$

where $\mathbf{Z}$ is the solution to the linear system

$$(\mathbf{H}_t \hat{\mathbf{P}}_t^f \mathbf{H}_t' + \mathbf{R}_t)\mathbf{Z} = \mathbf{Y_t} - \mathbf{H}_t \mathbf{u}_t^f, \tag{5.5}$$



(a) After first analysis update.

(b) After last analysis update.

Fig. 4.7: Estimated ensemble variance averaged over space and 50 algorithm runs.

thus circumventing the invertion of a $q_t \times q_t$ matrix [10]. We will call this the 'direct' method (DIR). An alternative method exploits the fact $\mathbf{R}_t$ is often easy to invert or decompose and reduces the problem to inverting an $N \times N$ matrix using the Sherman-Morrison-Woodbury formula (SMW) [9]. Another approach that only requires the calculation of $\sqrt{\mathbf{R}_t^{-1}}$ is to utilize singular value decompositions (SVD) [9, 10]. Lastly, an iterative method utilizing the Sherman-Morrison formula was proposed in [10] which only requires the inversion of $\mathbf{R}_t$ (ISM). Theoretical computational complexity for these approaches are available and given in Table 5.1.

Analysis method	Theoretical computational cost
DIR	$\mathcal{O}(pN^2 + q_tN^2 + q_t^2N + q_t^3)$
SMW	$\mathcal{O}(pN^2 + q_tN^2 + N^3)$
SVD	$\mathcal{O}(pN^2 + q_tN^2 + N^3)$
ISM	$\mathcal{O}(pN^2 + q_tN^2)$

Table 5.1: Theoretical cost of different methods for the analysis update.

If $\mathbf{R}_t$ is block-diagonal, observations can be assimilated sequentially in independent batches, which only requires inverting square matrices with dimensions of the batch size [7]. Sequential updates can be combined with any of the previously mentioned techniques. However, we do not consider this option as assimilating observations one-by-one was found to be prohibitively slow.

A major limitation of the direct method is the need to access the $q_t \times q_t$ matrix $(\mathbf{H}_t\hat{\mathbf{P}}_t^f\mathbf{H}_t' + \mathbf{R}_t)$. Naive storage of this matrix in memory quickly becomes infeasible for $q_t \geq 5e4$. As such, we only investigate this method up until that cutoff.

Table 5.2 shows empirical computation times for the selected methods when when p, q vary over $5e3, 1e4$, and $5e4$ with $p \geq q$. Computations were performed using Python 3.7.6, utilizing the library numpy 1.18.1 wherever possible, with the exception of scipy version 1.4.1 for solving linear systems and SVD. The observation operator was assumed to be dense and was applied to the state vector before computations. Observations were taken to be independent, i.e. $\mathbf{R}_t$ is diagonal. As we can see from Table 5.2, the SVD and SMW analysis

update techniques are substantially more computationally efficient than DIR and ISM. The poor performance of ISM, despite the attractive theoretical computational complexity, is likely due to the method's sequential Python implementation which does not take advantage of numpy's vectorized operations. Table 5.3 demonstrates the superior performance of the SMW method up until the largest state and observation space considered, where SVD is estimated to perform slightly better but with larger variance in computation times.

	State Dimension	Observation Count		
Method	p	5e3	1e4	5e4
DIR	5e3	3075.4 ± 23.0		
SMW		**56.4 ± 3.1**		
SVD		109.2 ± 8.5		
ISM		2073.3 ± 51.0		
DIR	1e4	3106.8 ± 14.4	10573.1 ± 22.8	
SMW		**72.7 ± 4.6**	**116.2 ± 6.1**	
SVD		121.9 ± 7.2	148.7 ± 5.4	
ISM		2063 ± 88.9	3996.5 ± 103.1	
DIR	5e4	3189.8 ± 20.3	10624.1 ± 25.3	—
SMW		**159.1 ± 6.5**	**209.5 ± 9.8**	**774.7 ± 10.1**
SVD		205.1 ± 11.3	236.6 ± 11.8	850.0 ± 34.1
ISM		2168.8 ± 109.7	4192.6 ± 98.1	10786.1 ± 141.8

Table 5.2: Mean ± standard deviation times for 10 runs averaged over 100 loops for each of analysis update formulations. Units are in miliseconds. Ensemble size was set to $N = 100$ for all calculationss.

	State Dimension	Observation Count		
Method	p	5e5	1e6	5e6
SMW	5e5	**1.16 ± 0.01**		
SVD		2.20 ± 0.02		
SMW	1e6	**1.29 ± 0.20**	**2.13 ± 0.02**	
SVD		2.33 ± 0.24	4.23 ± 0.12	
SMW	5e6	**2.36 ± 0.01**	**3.20 ± 0.01**	41.7 ± 0.37
SVD		3.26 ± 0.64	5.19 ± 44.4	**40.2 ± 1.33**

Table 5.3: Mean ± standard deviation times for 10 runs averaged over 100 loops for each of the two cheaper analysis update formulations. Units are in seconds. Ensemble size was set to $N = 50$ for all calculations.

6. Conclusion. Ensemble Kalman methods are extremely powerful and cost-effective tools for combining physics-based models with real-world measurements to provide estimates of a state variable. They can provide good performance on large-scale systems even with less than one-hundred forward model runs. However, as we have shown, even in non-Numerical Weather Prediction contexts the ensemble Kalman filter is still subject to spurious correlations and can benefit from localization methods. Specifically, we recommend the Gaspari-Cohn localization if filter divergence is the primary concern and a natural distance metric exists, and we recommend Anderson's hierchical filter for general usage or when no such distance metric exists or is appropriate to the dynamics of the problem. Furthermore, we note that the analysis update can be expedited considerably using the Sherman-Morrison-Woodbury formula.

REFERENCES

[1] J. L. Anderson, *Exploring the need for localization in ensemble data assimilation using a hierarchical ensemble filter*, Physica D: Nonlinear Phenomena, 230 (2007), pp. 99–111.
[2] C. Bishop and D. Hodyss, *Ensemble covariances adaptively localized with ECO-RAP. part 1: Tests on simple error models*, Tellus A: Dynamic Meteorology and Oceanography, 61 (2009), pp. 84–96.
[3] ———, *Ensemble covariances adaptively localized with ECO-RAP. part 2: A strategy for the atmosphere*, Tellus A: Dynamic Meteorology and Oceanography, 61 (2009), pp. 97–111.
[4] G. Evensen, *Data Assimilation: The Ensemble Kalman Filter*, Springer Berlin Heidelberg, 2006.
[5] G. Gaspari and S. E. Cohn, *Construction of correlation functions in two and three dimensions*, Quarterly Journal of the Royal Meteorological Society, 125 (1999), pp. 723–757.
[6] S. J. Greybush, E. Kalnay, T. Miyoshi, K. Ide, and B. R. Hunt, *Balance and ensemble Kalman filter localization techniques*, Monthly Weather Review, 139 (2011), pp. 511 – 522.
[7] P. L. Houtekamer and H. L. Mitchell, *A sequential ensemble Kalman filter for atmospheric data assimilation*, Monthly Weather Review, 129 (2001), pp. 123 – 137.
[8] R. E. Kalman, *A new approach to linear filtering and prediction problems*, (1960).
[9] J. Mandel, *Efficient implementation of the ensemble Kalman filter*, University of Colorado at Denver and Health Sciences Center, Center for Computational Mathematics, 2006.
[10] E. D. Nino-Ruiz, A. Sandu, and J. Anderson, *An efficient implementation of the ensemble Kalman filter based on an iterative Sherman-Morrison formula*, 2015.
[11] M. West and J. Harrison, *Bayesian forecasting and dynamic models*, Springer Science & Business Media, 2006.
[12] C. K. Wikle and L. M. Berliner, *A Bayesian tutorial for data assimilation*, Physica D: Nonlinear Phenomena, 230 (2007), pp. 1–16. Data Assimilation.

UNCERTAINTY IN REDUCED FINITE-RATE ABLATION MODELS FOR REENTRY VEHICLES

MAYAH DRAYTON*, RILEIGH BANDY†, AND TERESA PORTONE‡

Abstract.
In this work we will explore the Air-Carbon Ablation (ACA) model implemented by Prata et al to simulate the ablation of the carbon heat shield that protects the vehicle during reentry at hypersonic speeds. We want to reduce the computational cost to use this model by only using the most impactful reactions. We also want to quantify the uncertainty that will occur as we modify the ACA model. By focusing on these reactions, we hope to have a model that is cost-efficient without compromising its accuracy.

1. Introduction. When a vehicle reenters the atmosphere at hypersonic speeds, or above Mach 5, it is exposed to extreme temperatures that vary at different altitudes [1]. To protect these vehicles, a carbon heat shield (an ablator) is placed on the surface. During reentry, a plethora of chemical reactions occur that result in a loss of carbon, changing the surface in response to the high speeds [4]. Since testing the reentry of a vehicle to collect data from the flight is very costly, the design for the heat shield is dependent on computational models [6]. A good model will adequately predict what occurs in flight, and how chemical reactions will impact flight and the ablator [4].

Currently, there are two general ablation model approaches: equilibrium and nonequilibrium, or finite-rate. Equilibrium ablation models are often seen as conservative and tend to play it safe by overcompensating in the estimated loss of material in a heat shield [4]. Finite-rate ablation models are a fairly nascent approach to ablation modeling, so there is still significant uncertainty regarding what chemical reactions are needed and what the parameters for the reaction rates should be. Finite-rate can be the more appropriate approach depending on the type of flight condition we want to simulate. However, finite-rate models are computationally expensive in comparison to equilibrium models since the ablation model queries every location on the surface of the vehicle and time of flight [6]. Thus, finite-rate ablation models are too costly for practical use due to the sheer amount of locations and calculations that need to be performed.

In this work, the Air-Carbon Ablation (ACA) model [6] is used. Since nonequilibrium models are expensive, this work will test if it is possible to work with only the most impactful reactions, in hopes of it being less expensive. Throughout this testing, it is imperative to question if the removal of reactions is appropriate seeing that this model has uncertainty, and how it impacts the uncertainty.

The finding from our can contribute to developing more accurate and cost-efficient computational models for predicting the changes that occur to the protective carbon heat shields during a hypersonic flight. This work could also enhance the reusability of the vehicles after being exposed to extreme temperatures. Our framework is general, allowing its application to future ablation models that could include more complex sets of reactions for complex materials, for example, ceramics, decomposing materials etc.

2. Background. The nomenclature for the ACA model is defined in Table 2.1. The ACA model takes the gas species in the air and the possible chemical reactions occurring at the vehicle's surface to give the probability of 11 different chemical species, with carbon monoxide (CO) being anticipated as the major ablation product, forming. In the ACA

*Department of Electrical & Computer Engineering, North Carolina A&T State University
†Sandia National Laboratories, rjbandy@sandia.gov
‡Sandia National Laboratories, tporton@sandia.gov

TABLE 2.1
Nomenclature

A_v	=	Avogadro constant, mol^{-1}
B	=	total active site density, $\text{mol}\cdot\text{m}^{-2}$
E	=	Activation energy, J
f	=	flux of a species, $\text{mol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$
h	=	Planck's constant, J·s
k_b	=	Boltzmann constant, $\text{J}\cdot\text{K}^{-1}$
m	=	mass of an atom or molecule of a species, kg
P	=	Partial pressure, Pa
R	=	Universal gas constant, $\text{J}\cdot\text{K}^{-1}\cdot\text{mol}^{-1}$
S	=	sticking coefficient
T	=	Temperature, K
x	=	Longitudinal distance from the stagnation point, m
γ	=	reaction probability
Subscript		
G	=	Gas

model, there are 20 chemical reactions that fall into three reaction sets: weakly bonded oxygen (O), strongly bonded oxygen (O*) and bonded nitrogen (N).

The reactions that make up the ACA model are present in Table 2.2 which include the parameters used to define the reaction rates. The ACA model includes four different reaction types, or how these chemical species occur: adsorption, desorption, Eley-Rideal, and Langmuir-Hinshelwood [1, 4]. We are summarizing the reaction rate coefficient, k, defined in [6]. Below are the equations to demonstrate how the table is implemented in the computation of the model.

Adsorption occurs when the chemical species attach to the free surface site(s). The power of B depends on the gas species being adsorbed: O and $N, k = 1$ and $O_2, k = 2$ [1]. The reaction rate with an adsorption reaction type takes the form

$$k_{ad} = \frac{F_G}{B^k} S \exp\left(-\frac{E}{RT}\right), \tag{2.1}$$

where

$$F_G = \frac{1}{4}\sqrt{\frac{8k_bT}{\pi m_G}}.$$

Desorption is the inverse of adsorption, as the reactant detaches from the surface back into the air without taking any carbon from the surface. The reaction rate for desorption takes the form

$$k_{de} = \frac{2\pi m_G k_b^2 T^2}{A_v B h^3} \exp\left(-\frac{E}{RT}\right). \tag{2.2}$$

Eley-Rideal occurs when a chemical species in the air collides with a molecule that's already absorbed on the surface and goes through a transformation almost instantly, allowing a carbon-gas species to form. The reaction rate for Eley-Rideal takes the form

TABLE 2.2
ACA gas-surface chemistry model reaction set.

AD = Adsorption, DE = Desorption, ER = Eley-Rideal, LH = Langmuir-Hinshelwood

#	Reaction	S	γ	E/R	Type
1	$O + (s) \rightarrow O(s)$	0.3	–	0	AD
2	$O(s) \rightarrow O + (s)$	–	–	44277	DE
3	$O + O(s) + C(b) \rightarrow CO + O + (s)$	–	100.0	400	ER
4	$O + O(s) + C(b) \rightarrow CO_2 + (s)$	–	1.0	500	ER
5	$O + (s) \rightarrow O^*(s)$	0.7	–	0	AD
6	$O^*(s) \rightarrow O + (s)$	–	–	96500	DE
7	$O + O^*(s) + C(b) \rightarrow CO + O + (s)$	–	1000.0	4000	ER
8	$O^*(s) + O^*(s) \rightarrow O_2 + 2(s)$	–	10^{-3}	15000	LH
9	$O(s) + O(s) \rightarrow O_2 + 2(s)$	–	5×10^{-5}	15000	LH
10	$N + (s) \rightarrow N(s)$	1.0	–	2500	AD
11	$N(s) \rightarrow N + (s)$	–	–	73971	DE
12	$N + N(s) + C(b) \rightarrow CN + N + (s)$	–	1.5	7000	ER
13	$N + N(s) \rightarrow N_2 + (s)$	–	0.5	2000	ER
14	$N(s) + N(s) \rightarrow N_2 + 2(s)$	–	0.1	21000	LH
15	$N(s) + C(b) \rightarrow CN + (s)$	–	10^8	20676	ER
16	$O_2 + 2(s) \rightarrow 2O(s)$	1.0	–	8000	AD
17	$O_2 + O(s) + C(b) \rightarrow CO + O2 + (s)$	–	100.0	4000	ER
18	$O_2 + O(s) + C(b) \rightarrow CO_2 + O + (s)$	–	1.0	500	ER
19	$O_2 + 2(s) \rightarrow 2O^*(s)$	1.0	–	8000	AD
20	$O_2 + O^*(s) + C(b) \rightarrow CO + O_2 + (s)$	–	1000.0	4000	ER

$$k_{er} = \frac{F_G}{B}\gamma \exp\left(-\frac{E}{RT}\right). \tag{2.3}$$

Langmuir-Hinshelwood occurs on the surface when two chemical species are already adsorbed to the surface, react with each other, and desorb from the surface. Langmuir-Hinshelwood does not produce carbon in the air, nor involve bulk carbon in its reactions. The reaction rate for Langmuir-Hinshelwood takes the form

$$k_{lh} = \sqrt{\frac{A_v}{B}} F_{G,2D}\gamma \exp\left(-\frac{E}{RT}\right), \tag{2.4}$$

where

$$F_{G,2D} = \sqrt{\frac{\pi k_b T}{2m_G}}.$$

When reaction rate parameters were originally fit to data, the environment conditions were different from the hypersonic flight conditions we aim to use in the ACA model. Data was collected using molecular beam experiments, where the measurement of individual reactions and rate parameters is enabled by using a pulsed beam of a gas-species to impact a heated material surface. This then allows the measurement of reaction products. In contrast, the air in flight conditions will have a mix of atomic chemical species, like oxygen and nitrogen, and their molecules. Additionally, temperatures and pressures will vary significantly from those in experiment during flight [3, 5]. Because of these differences between the conditions of the experiments used to fit the parameters originally and flight conditions we want to model, there is uncertainty in the parameter values [6].

The ACA model takes as inputs the incoming fluxes of atomic and molecular oxygen and nitrogen, as well as temperature and pressure at the surface of the vehicle. It outputs fluxes from the surface of the 11 chemical species included in Table 2.2. Further discussion of the numerical solution of the model is presented in [6]. Our goal is to compare the predicted CO probability, which is the ratio of carbon monoxide that is probable for production. The CO probability can be found from the flux of CO off the surface divided by the total oxygen flux to the surface in the form of atomic and molecular oxygen and

$$\text{CO probability} = \frac{f_{\text{CO}}}{f_{\text{O}} + 2f_{\text{O}_2}}. \tag{2.5}$$

Carbon monoxide probability is our quantity of interest because carbon monoxide is the most common way carbon is ablated from the surface.

3. Method. The objective of this study is to remove bonding groups and explore the potential for cost reduction in nonequilibrium models by selectively removing less impactful reactions. We modified the model code by adding a function to remove preset groups of reactions. These sets of reactions are the three different types of bonds as well as a 'nil' set, which provides a full model as it does not remove any reactions. When removing, or zeroing out, reactions, we opt to choose an entire bonding set instead of individual reactions. Specific reactions in Table 2.2 are the foundation of others, so if one is removed, it can implicitly remove that of another. It's important to be cautious of this because we may not get proper results, so in this work we opted to remove all of a bond's reactions.

As discussed in the previous section, parameter values for this model were hand set to fit a specific experiment that was different from the conditions that are experienced in a hypersonic flight. Because of the difference in the environments this leads us to have a level of uncertainty in the parameters when considering reentry flight conditions. Since we are dealing with uncertainty in our inputs, we will have uncertainty in our output. As we remove reactions, we want to study the relationship between reduced models and full models, looking at what has drastically changed, or had little to no change occur while considering the uncertainty in our model, then comparing how output statistics change versus the full model. We tackle this uncertainty by feeding random samples of the parameters in a specified range through the model, then comparing how output statistics change versus the full model.

There are 34 parameters defined to be uncertain. Probability distributions to express uncertainty in these parameters were defined in [1] and are summarized here for completeness.

The adsorption selectivity (S) is assumed to fall between 0 and 1, inclusive

$$S_j \sim U[0,1], \qquad j \in \{1, 10, 16, 19\}, \tag{3.1}$$

for all adsorption-type reactions, save reaction 5 since $S_5 = 1 - S_1$ [4]. S_1 and S_5 should always total to one since oxygen must either bond weakly or strongly to the surface, and it is assumed that all incoming atomic oxygen bonds to the surface.

For pre-exponential factor $\gamma \leq 1$, theoretical bounds hold, and it is assumed to fall between 0 and 1, inclusive

$$\gamma_j \sim U[0,1], \qquad j \in \{4, 8, 9, 13, 14, 18\}. \tag{3.2}$$

When the parameter γ in [6] exceeds one, theoretical bounds do not apply, so the bounds are assumed to be $\pm$ 20% of Prata's nominal value:

$$\begin{aligned}
&\gamma_j \sim U[80, 120], \qquad j \in \{3, 17\} \\
&\gamma_{12} \sim U[1.2, 1.8] \\
&\gamma_{15} \sim U[8 \times 10^8, 12 \times 10^8] \\
&\gamma_{17} \sim U[80, 120] \\
&\gamma_j \sim U[800, 1200], \qquad j \in \{7, 20\}.
\end{aligned} \tag{3.3}$$

The activation energies could be any nonnegative real value, but for the sake of this work they were given bounds an order of magnitude $\pm$ the listed Prata nominal values:

$$\begin{aligned}
&\log_{10} E_j \sim U[3,5], \qquad j \in \{2, 6, 8, 9, 11, 14, 15\} \\
&\log_{10} E_j \sim U[2,4], \qquad j \in \{3, 7, 10, 12, 13, 16, 17, 19, 20\} \\
&\log_{10} E_j \sim U[1,3], \qquad j \in \{4, 18\}.
\end{aligned} \tag{3.4}$$

Activation energies for reactions 1 and 5 are fixed at 0 to reflect the assumption that there is no activation energy for these reactions to take place.

We can generate random samples of the parameters from these probability distributions and evaluate the ACA model. This gives us output samples that we can compute statistics from. In this work we compute the mean of the CO probability and compare it for changes between the models. We compare the 95% confidence intervals of these sample means to judge whether we have statistically significant differences. We utilize the central limit theorem [2] to estimate the shaded region in Fig 4.1. This region is the 95% confidence interval for our mean. Since our computed mean is still considered a random variable, we use the central limit theorem to show where we generally expect to find the true mean:

$$\begin{aligned}
&\hat{\mu} + 2\frac{\hat{\sigma}}{\sqrt{N}} \\
&UpperBounds \\
&\\
&\hat{\mu} - 2\frac{\hat{\sigma}}{\sqrt{N}} \\
&LowerBounds
\end{aligned} \tag{3.5}$$

where $\hat{\mu}$ is the sample mean, $\hat{\sigma}$ is the sample standard deviation, and N is the number of samples. The mean is calculated over several altitudes, from 20 kms to 40 kms, at mulitple spatial points along the surface. We want to look at the several altitudes to note the changes made over the trajectory of the flight.

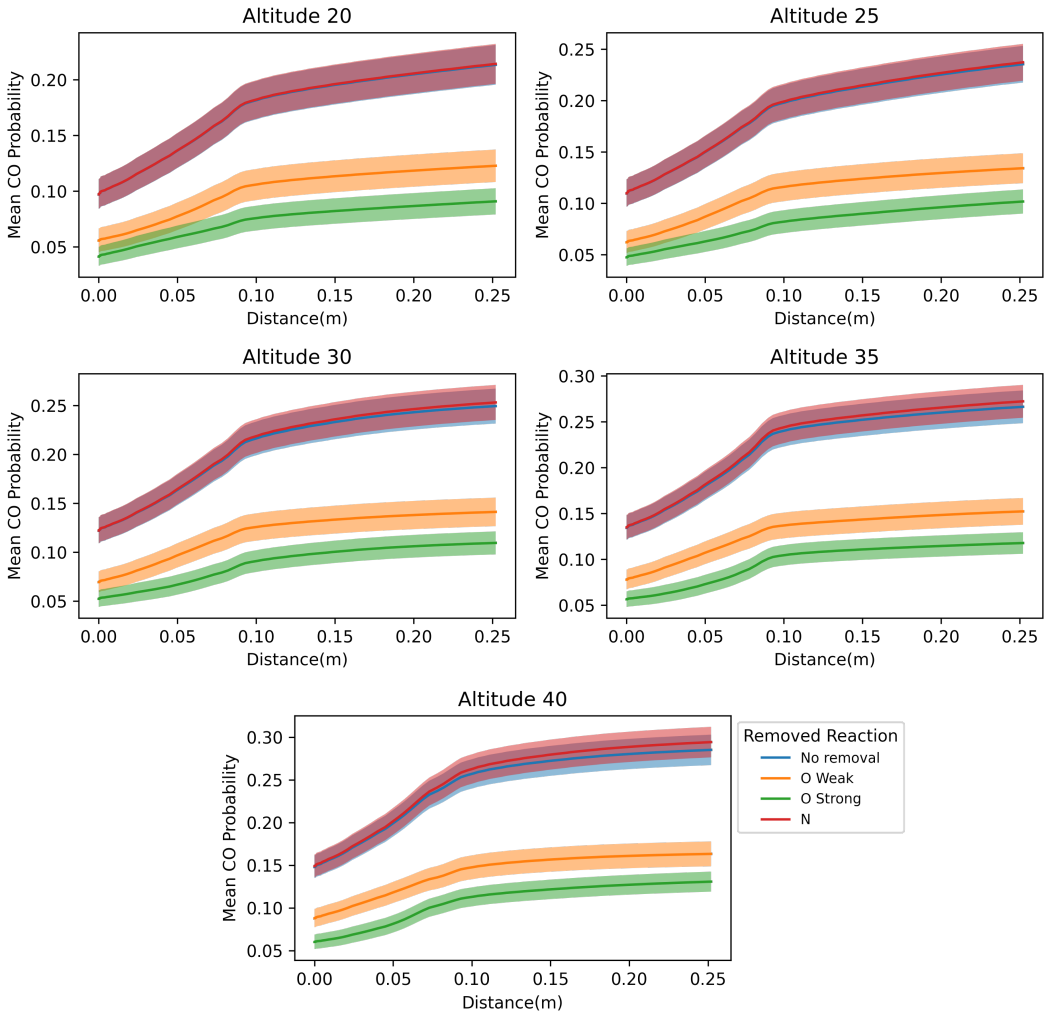


FIG. 4.1. *Plots of modified models comparing the mean CO Probability*

4. Results. In this work, we generated samples and use the same set of samples for each modified model to have consistency when assessing the change that occurred. The number of samples allows us to distinguish between confidence bounds from both the full model and the modified models. Since our goal with dropping equations is to see a difference in the mean, we want to have 95% confidence intervals that are small enough that we can say with confidence if a model is different. If N is too small, the 95% confidence intervals would all overlap, and we could not accurately determine the differences. With this in mind, we decided to generate 1000 samples.

We demonstrate the mean and 95% confidence intervals for the full model and the reduced models in Figure 4.1. We see that removal of nitrogen impacts the mean CO probability the least compared to the full model at all altitudes and surface locations. The results we see are because N only impacts CO through competition with O(s) and O*(s) for surface sites. It is important to note that the higher the altitude and the further from nose of the vehicle we are, the more of a difference in CO probability there is.

When N-bonded reactions are removed, they are also removed from the total site density. This allows for other reactions to increase at the site, since the sites that previously had nitrogen bonds are now replaced with the bonds remaining, O(s) and O*(s). This is why the CO probability for the N-removal model is higher than that of the full model seen in Figure 4.1.

The most important bond group is O*, as its removal drastically decreases the probability of CO occurring. It has the greatest difference in probability in comparison to the rest of the modified models. Why? A major factor is the ranges applied to γ for O(s) and O*(s), based on their nominal values. O*(s) has a γ value 10 times larger than O(s) when comparing reactions 3 and 7, so its reaction rate is 10 time larger. This means O*(s) is much more important to CO production than O(s) in the full model. With the removal of O* bonds, it will heavily under-predict the true amount of CO that will be produced and change the carbon shield's surface. Since under-predicting CO production leads to under-predicting the shield recession, a too-thin shield would be applied, which can be detrimental to the survival of the reentry vehicle.

5. Conclusion. With finite-rate ablation models being expensive to use, we worked to reduce the number of reactions to calculate by removing what we deemed as the least impactful reaction group. We took into consideration the uncertainties in reaction parameters arising from the difference in experiment versus flight conditions. By generating random input samples within given constraints, we propagate them through the model to return the sample mean. We then compared the modified model output means to the full model. We summarize confidence in our statistical results using the Central Limit Theorem to compute 95% confidence intervals. When looking at our results, we consider the differences in the CO probability to determine the impact of dropping reaction groups from our model. Our results demonstrate that a modified model that removes nitrogen reactions would have the least impact in the probability of CO occurring during a flight, while removing O* reaction would under-predict the probability in CO being produced, making it the most impactful reaction group.

For future work, we want to explore which individual reactions we can remove, rather than whole groups. We also want to understand how making changes in the parameter bounds would impact the output distribution.

Acknowledgments. The authors would like to thank Rebekah White and Erin Mussoni for their helpful and instructive conversations. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. LDRD #233072. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] R. Bandy, *Uncertainty Representations in White- and Black-Box Models: Quantifying Model-Form and Measurement Errors in Computational Science*, PhD thesis, University of Colorado, 2024.

[2] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*, John wiley & sons, 2010.

[3] V. J. Murray, P. Recio, A. Caracciolo, C. Miossec, N. Balucani, P. Casavecchia, and T. K. Minton, *Oxidation and nitridation of vitreous carbon at high temperatures*, Carbon, 167 (2020), pp. 388–402.

[4] E. Mussoni, *A Quantitative Analysis of Finite-Rate Surface Ablation Models for Hypersonic Flight*, PhD thesis, University of California, Davis, 2024.

[5] S. Poovathingal, T. E. Schwartzentruber, V. J. Murray, T. K. Minton, and G. V. Candler, *Finite-Rate Oxidation Model for Carbon Surfaces from Molecular Beam Experiments*, AIAA Journal, 55 (2017), pp. 1644–1658. Number: 5 Publisher: American Institute of Aeronautics and Astronautics.

[6] K. S. Prata, T. E. Schwartzentruber, and T. K. Minton, *Air–Carbon Ablation Model for Hypersonic Flight from Molecular-Beam Data*, AIAA Journal, 60 (2022), pp. 627–640. Number: 2 Publisher: American Institute of Aeronautics and Astronautics.

IMPLICATIONS OF THE TWO INTERACTING BLAST WAVE VERIFICATION PROBLEM FOR COMPUTATIONAL SHOCK HYDRODYNAMICS

RAFAEL DE FARIAS*, MARISSA B. P. ADAMS†, AND WILLIAM J. RIDER‡

Abstract. The Two Interacting Blast Wave problem is a crucial benchmark for assessing the performance of hydrodynamic multiphysics codes [1, 2]. This problem is relevant for various applications, from astrophysical systems [3] to experiments on the Z Machine at Sandia National Laboratories [4]. We utilize this problem to evaluate the code credibility of the Sandia Multi-physics/Architecture Adaptive Shock Hydrodynamics (SMASH) code. After implementing the problem in SMASH, we conduct a resolution scan for a self-convergence test, demonstrating convergence with the L_1-norm on the order of unity. Additionally, we vary the ratio of specific heats to investigate the impact of equations of state in preparation for future work.

1. Introduction. In a 1984 paper by Paul Woodward and Phillip Colella, the Two Interacting Blast Wave (TIBW) problem was introduced as a one-dimensional shock physics problem designed to test methods for handling strong shocks [1, 2]. Today, this problem serves as a well-known benchmark for verifying the credibility of hydrodynamics codes [5–7]. The development of verification problems or toy models of hydrodynamical flows, such as the TIBW, was particularly timely in the early 1980s due to the advent of vectorized computers [1]. In 1984, Woodward and Colella focused on simulating fluid flows that generate strong shocks, where substantial entropy is produced. They evaluated the advantages and disadvantages of treating such discontinuities in numerical fluid flows by analyzing the results of such test problems [2].

Today, developers of multiphysics codes, inclusive of hydrodynamics, find themselves in a similar position with the advent of next-generation computational resources, such as graphical processing units (GPUs). To ensure that these codes are capable and trustworthy on new machines, they are expected to undergo processes of Verification, Validation, and Uncertainty Quantification (VVUQ) [8, 9]. In any computational simulation, the credibility of the results is paramount: how can we trust what the computer tells us? VVUQ techniques are essential for ensuring that the models being used are reliable. Without VVUQ, disasters can occur in systems that rely on computational models, as evidenced by the recent Boeing 737 MAX crashes [10].

Verification assesses whether a computational model accurately represents the underlying mathematical model. Within this overarching concept are two components: (1) code verification and (2) solution verification. Both are crucial for establishing credibility, especially for the TIBW problem. Code verification is achieved by comparing simulation results to analytical solutions or established benchmarks. Solution verification involves ensuring that convergence rates and discretization errors are sufficiently accurate [8, 9]. In contrast, validation concerns the underlying physics of the simulations. It assesses whether a computational model accurately represents reality by comparing results to experimental data [8]. Finally, Uncertainty Quantification (UQ) involves identifying and reducing uncertainties in the model. When running an ensemble of simulations, UQ is critical for understanding confidence in the models [8].

For any multiphysics code, these assessments are vital for building trust in the produced models. In this article, we assess methods for verifying a next-generation multiphysics

*University of Rochester, rdefaria@u.rochester.edu,
†Sandia National Laboratories, mbadams@sandia.gov,
‡Sandia National Laboratories, wjrider@sandia.gov

Fig. 2.1: A visualization of the initial pressure conditions for TIBW in ParaView.

Table 2.1: TIBW initial conditions for each region of material over one spatial direction (x), density, corresponding velocity component (u_x), and pressure.

	Left	Middle	Right
x	0 - 0.1	0.1 - 0.9	0.9 - 1
ρ	1.0	1.0	1.0
u_x	0	0	0
p	1000	0.01	100

code: the Sandia Multi-physics/Architecture Adaptive Shock Hydrodynamics (SMASH), for problems like the TIBW, which have no known analytic solution. We present a self-convergence analysis, where high-resolution runs provide a benchmark for comparison. We choose to investigate the TIBW because it is a unique and valuable problem for multiphysics codes, featuring complex shock interactions, rarefaction waves, and contact discontinuities all in one simulation. This problem will help us determine (1) whether there are difficulties in capturing strong shocks and multiple interactions, (2) when solvers fail, and (3) how the sharpness of the contact discontinuity at the final time reveals differences between Eulerian and Lagrangian schemes. These physics and numerical considerations are crucial for any multiphysics code, including hydrodynamics, to test and capture. The major features of this problem are observed in astrophysics, high-energy density physics, computational fluid dynamics, and aerodynamics [3, 4].

The problem is initialized (Section 2) with "two diaphragms" that at time $t = 0$ are removed, initiating two blast waves that propagate towards the center of the tube and interact. Meanwhile, rarefactions propagate towards the ends of the simulation domain. The boundary conditions stipulate that the waves reflect at both ends of the tube. The simulation is run to a time of $t = 0.038$ s. This problem is difficult to solve and analyze for a couple reasons: (1) the problem lacks any symmetry due to the pressure initial conditions, and (2) there is no known analytic solution. This makes solution verification utilizing this problem a challenge as typically such practices involve convergence studies that compare to a known analytic solution.

2. Initial Conditions for the Two Interacting Blast Wave Problem (TIBW). The TIBW problem conceptualizes a 1-D shock tube of length 1 cm partitioned by two diaphragms at $x = 0.1$ cm and $x = 0.9$ cm [2]. Initially, the mass density of all three regions

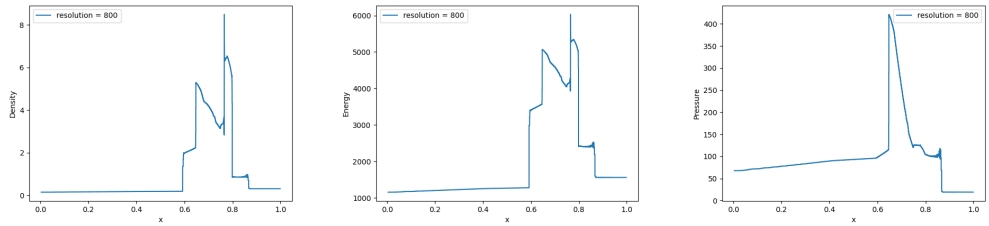


Fig. 4.1: TIBW SMASH results at $t = 0.038$ s for density, energy, and pressure using 800 cells.

is $\rho = 1.0$ g/cm^3 and the velocity is $u = 0.0$ cm/s. The leftmost, center, and rightmost region have initial pressures of $p_L = 10^3$ dyn/cm^2, $p_M = 10^{-2}$ dyn/cm^2, and $p_R = 10^2$ dyn/cm^2, respectively [2]. We detail these initial conditions in Table 2.1. Furthermore, we can visualize a variation on these initial conditions, as illustrated in Figure 2.1. Note that in Figure 2.1 on the left the higher pressure region in red, and the lower pressure region on the very right.

3. Governing Equations. This problem is governed by the 1-D Euler equations,

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = 0 \tag{3.1}$$

$$\frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2 + p)}{\partial x} = 0 \tag{3.2}$$

$$\frac{\partial (\rho E)}{\partial t} + \frac{\partial ([\rho E + p)u]}{\partial x} = 0 \tag{3.3}$$

with total energy per unit mass, E, equal to

$$E = e + \frac{u^2}{2} \tag{3.4}$$

and e being the internal energy. Here eq. (3.1), eq. (3.2), and eq. (3.3) state conservation of mass, momentum, and energy, respectively. These three equations are closed by the ideal gas law, our chosen equation of state (EOS), such that pressure, p, is equal to

$$p = (\gamma - 1)\rho e. \tag{3.5}$$

Here, γ in eq. (3.5) is defined as the adiabatic index. The material described in the original paper is diatomic (assumed to be air) therefore, $\gamma = 1.4$.

4. The Sandia Multi-physics/architecture Adaptive Shock Hydrodynamics (SMASH) Code. All simulated results of the TIBW were performed on the Sandia Multi-physics/architecture Adaptive Shock Hydrodynamics (SMASH) code. SMASH is a next-generation shock physics code developed at Sandia National Laboratories (SNL) at present, in 2024. Using SMASH, TIBW was simulated on a Lagrangian mesh, advancing eq. (3.1) - eq. (3.3). SMASH solves these governing equations to solve for all unknown variables at time

Algorithm 1 "smashed result" method

ratio of resolutions = higher resolution / lower resolution
for each higher resolution data entry
if the data entry index / ratio of resolutions = whole number **then**
smashed result = sum of that entry and the next (ratio of resolution - 1) entries / ratio
end if
return all smashed results

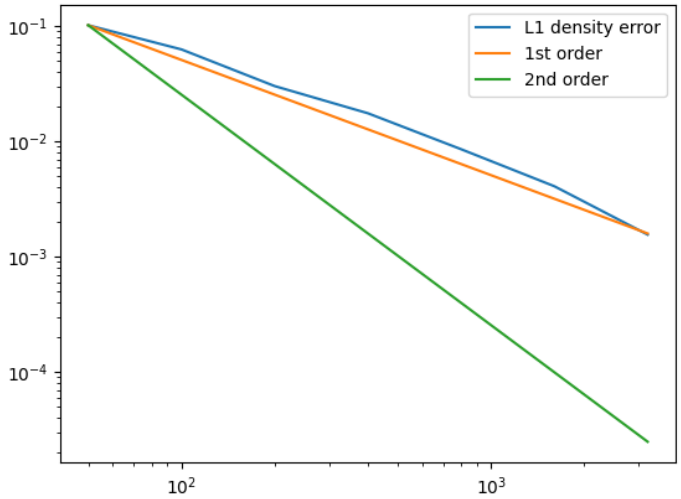


Fig. 5.1: Illustration of order unity convergence in density for the TIBW problem in SMASH using the self convergence methodology for solution verification.

realizations in accordance with some timestep. SMASH uses density, velocity, and internal energy for initial conditions for the hydrodynamic state. Therefore we utilized (eq. (3.5)) to determine the internal energy for a given pressure presented by Woodward and Colella (Table 2.1). A development goal for SMASH is to include pressure as an initial condition to forego this process in the future.

At the final simulation time, the following plots were created (Figure 4.1), computing the EOS variables such as density, energy, and pressure over their respective centroid value on the 1-D computational domain. These results were computed using three "materials" for each spatial region using 800 cells within a domain of length equal to unity. We note comparable features to the original Woodward and Colella paper at appropriate locations. However, we also note the anomalous peak before $x \sim 0.8$ in both the density and energy plots in Figure 4.1. This is likely due to the resolution being quite large compared to the what is presented by Woodward and Colella.

5. Application of self convergence to TIBW. An exact analytic solution for this problem does not exist as of now. Therefore we use a method for comparing lower resolution results to higher resolution results in order to assess accuracy of SMASH's computation. Typically, with an exact solution, the following can be used to calculate the L_1-error (and then comparably L_2, L_∞ errors if desired by raising the norm to some power):

$$L_1 = \frac{\sum V_i(|\rho_{\text{SMASH}} - \rho_{\text{exact}}|)}{\sum V_i}, \tag{5.1}$$

where ρ_{SMASH} would be the computed density from SMASH, and ρ_{exact} would be the analytic formulation of the density. Note that one can replace any other unknown variable aside from density in eq. (5.1).

We adopt an alternative approach to compute our L_1-error:

$$L_1 = \frac{\sum V_i(|\rho_{\text{lower}} - \rho_{\text{higher}}|)}{\sum V_i} \tag{5.2}$$

where ρ_{lower} is the solution in density (replace with any other unknown variable) at a subsequent lower resolution in comparison to a simulation that computed ρ_{higher}.

In order to utilize this version of our L_1-error (eq. (5.2)), a dimensional issue must be resolved. Since the sizes of the data arrays are equal to the resolutions, comparing a higher resolution result to a lower resolution result requires mapping the data from the higher dimension to the lower dimension. The routine described by algorithm 1, coined the "smashed result" (SR) method, was adopted in order to map the results to a lower dimension.

The SR method would allow for finding the L_1 error if there was no error in the solution for the higher resolution; it would be comparing to itself. Recall, the higher resolution solution is not an exact solution. The following theory relating an exact solution, S_∞, and the solution at a particular cell size, S_h, is introduced [9]:

$$S_\infty = S_h + Ah^\alpha \tag{5.3}$$

Equation (5.3) states that with a particular cell size ($h = 1/\Delta x$, where Δx is the resolution in x), the exact solution is equal to the sum of the solution at that cell and an estimate of the discrete error. For the discrete error estimate term, A and α are unknown constants that are problem dependent. Given this theory, we can also say that

$$S_\infty = S_{h/2} + A\left(\frac{h}{2}\right)^\alpha. \tag{5.4}$$

Then, setting both eq. (5.3) and eq. (5.4) equal to one another yields,

$$S_{h/2} + A\left(\frac{h}{2}\right)^\alpha = S_h + Ah^\alpha. \tag{5.5}$$

In order to solve for unknown constants, A and α, we need to introduce a second equation where $x \neq h$:

$$S_{x/2} + A\left(\frac{x}{2}\right)^\alpha = S_x + Ax^\alpha. \tag{5.6}$$

Solving for the systems of equations yields discrete error estimate terms A and α. Inserting these constants into eq. (5.3) yields an exact solution for which an L_1 error may be found. When comparing L_1 norm for all of the resolutions computed for the TIBW using SMASH using this method, we demonstrate convergence to first order accuracy (Figure 5.1).

Table 5.1: The initial conditions for the Sod shock tube, for each region of material over one spatial direction (x), density, corresponding velocity component (u_x), and pressure.

	Left	Right
x	0 - 0.5	0.5 - 1.0
ρ	1.0	0.125
u_x	0	0
p	1.0	0.1

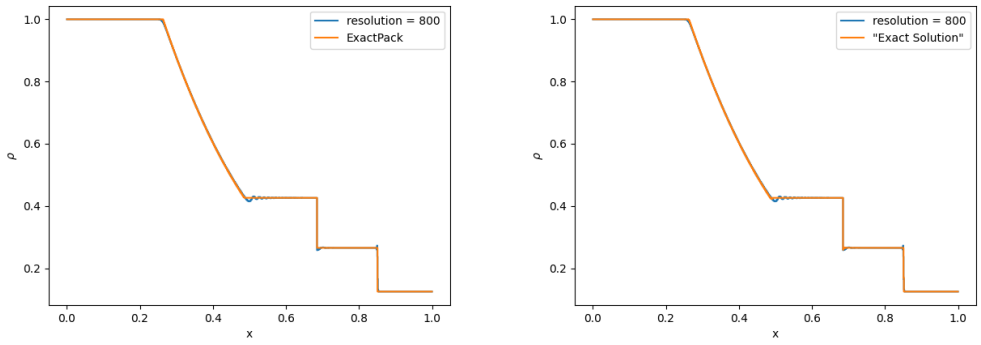


Fig. 5.2: Simulated mass density of the Sod problem using 800 cells, compared with the `ExactPack` exact solution.

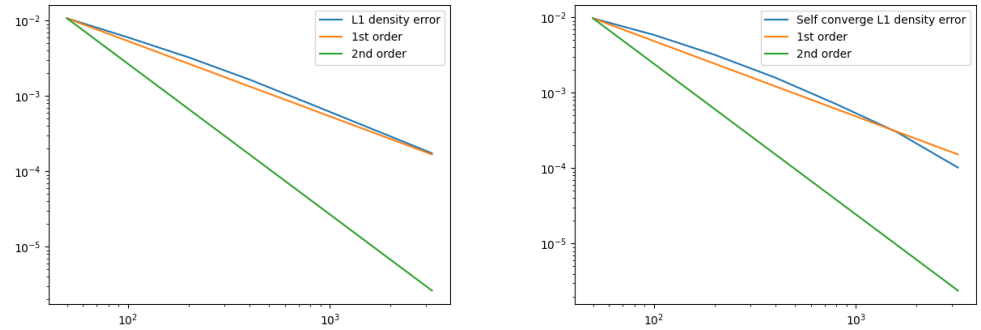


Fig. 5.3: Convergence plots for Sod's mass density using `ExactPack` and SMASH's self convergence routine.

5.1. Verifying the self convergence implementation. In order to assess whether or not our SR method and the self convergence routine is complaint, we apply it to a problem with a known exact analytic solution. In essense, we are verifying our verification tools. We use `ExactPack`[1], a code developed by Los Alamos National Laboratory (LANL) [11], features a method for calculating the analytic solution for Riemann problems (among many other hydrodynamic problems with "exact" solutions), a set of shock physics problems

[1] `https://github.com/lanl/ExactPack`

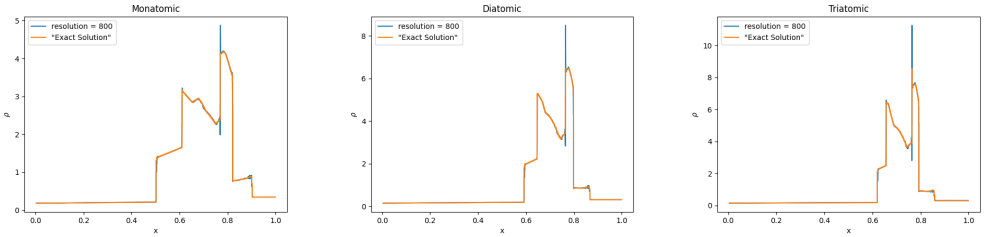


Fig. 6.1: For the same resolution, density is plotting after varying the ratio of specific heats such that the gas is (1) monatomic, (2) diatomic, and finally (3) triatomic.

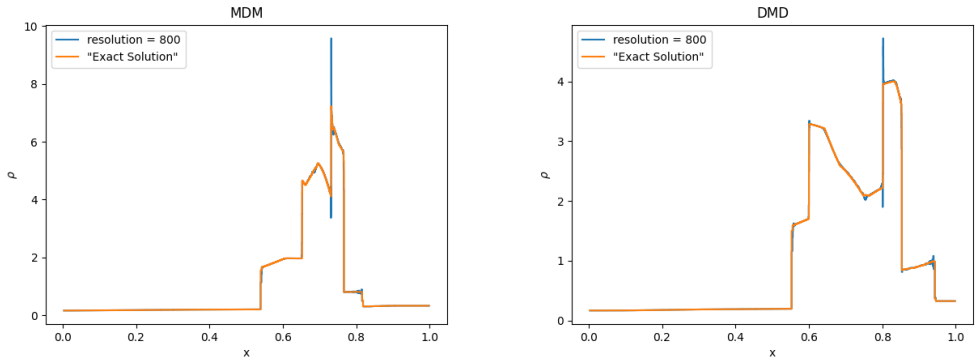


Fig. 6.2: Varying the ratio of specific heats across the three regions of the TIBW problem where (left) is monatomic-diatomic-monatomic, and (right) diatomic-monatomic-diatomic.

governed by the same governing equations as the two interacting blast wave problem. We choose Sod, a comparable problem to the TIBW, but only featuring a single diaphragm and hence only one blast wave [12].

The Sod problem [12] is a Riemann problem with an adiabatic index of $\gamma = 1.4$ simulated for $t = 0.25$ s with the initial conditions tabulated in Table 5.1. The Sod problem is a simple problem that features a single blast wave moving to the right, making contact with the boundary, and a rarefaction then moving to the left. By running Sod both with the `ExactPack` routine, and the SMASH self-convergence routine, a comparison of the solvers was made in Figure 5.2 and Figure 5.3. In Figure 5.2 on the right, the 'Exact Solution' is effectively the SMASH result using the SR method for self convergence studies. We note that in Figure 5.3 that convergence is achieved by comparing to the analytic solution to order unity (left image). However, the self convergence method, despite being approximately order unity, will begin estimating higher order accuracy at higher resolutions. The self convergence method also is not as close to order one as in the case of using the exact solution. Despite this, the self convergence methodology is identified as the only way we can achieve solution verification for problems with no known solution. Since the `ExactPack` solution and the solution developed on SMASH both converge to first order, confidence in the SMASH developed solution solver for solution verification is established as acceptable.

6. Exploration by varying the ratio of specific heats. After the problem outlined by the original paper was simulated and verified (see prior discussions: sections 5 and 5.1), an investigation of SMASH's capabilities beyond the domain of the original problem was

taken. The first thing to explore would be how altering the ratio of specific heats, γ, impacts the results at the final time. Based on the classical equipartition theorem, for an ideal gas, γ is defined by

$$\gamma = 1 + \frac{2}{f} \tag{6.1}$$

where f is equal to the number of degrees of freedom accessible to the particle. Since the problem assumes a diatomic gas, γ was set to be 1.4 for the problem for all three regions. We find exploration of varying γ to be an interesting case study, as we can eventually build up and understanding of how to confront the problem of the vacuum-gas interface in hydrodynamics codes; eventually aiming to consider the LeBlanc problem [12]. To start we begin by varying the degrees of freedom to see how it alters the density calculations. For a monatomic gas, like the noble gasses, $\gamma = 5/3$, whereas for a triatomic gas, $\gamma = 8/6$.

After modifications to the input file were made to allow this investigation, (Figure 6.1 were produced for comparison of two interacting blast waves in a medium filled by a monatomic, diatomic, and triatomic gasses. We observe with increasing degrees of freedom, the shock interaction region at the final time decreases; the problem appears to be shrinking. However, between the diatomic and triatomic cases, the sharp contact discontinuity remains at $x = 0.6$. However, the monotonic case moves this feature closer to the origin, at approximately $x \sim 0.5$.

The original problem assumes that the gas in all three mediums are identical, however, a modification to the problem investigates if the center region was filled with a different ideal gas than the left and right regions. We choose to specify which region will have which flavor of gas, such as a monatomic-diatomic-monatomic (MDM) or a diatomic-monatomic-diatomic (DMD) arrangement. We implement this in the input deck for the SMASH code, and problem produces Figure 6.2. We see for the MDM case that the densities achieved for the whole shock interaction region are higher than the DMD case, almost an order higher (order one versus order ten). We find this interesting in contrast to what is presented in fig. 6.1 where the pure monotonic case yields smaller densities (peaks of approximately five) versus the cases of larger degrees of freedom. It appears that the more degrees of freedom the gas contains between the diaphragms impacts the resulting peak densities; they are slightly larger. This may be due to the energy distribution for such ideal gases, as gases with larger degrees of freedom can store more energy. This allows them to absorb and redistribute the energy more effectively during shock interactions; it may yield a more efficient compression mechanism.

7. Conclusion. We investigated the Two Interacting Blast Wave (TIBW) problem using the Sandia Multi-physics/Architecture Adaptive Shock Hydrodynamics (SMASH) code. Given the absence of an exact analytic solution, we employed a self-convergence method to assess the accuracy of our simulations by comparing lower-resolution results to higher-resolution ones. We also aimed to verify our approach by applying it to the Sod problem using `ExactPack`, which has a known analytic solution. Our findings demonstrated first-order convergence for the TIBW in density, verifying the effectiveness of the SMASH code for this complex problem. This approach not only enhances our confidence in the code's performance but also establishes a framework for future verification efforts in similar computational scenarios, e.g. where exact solutions are difficult to obtain for toy verification problems.

We explored the impact of varying the ratio of specific heats on the shock interaction dynamics, revealing that gases with more degrees of freedom can store and redistribute

energy more effectively, leading to higher peak densities. We also modified the problem to investigate different gas compositions in the shock tube, finding significant differences in density profiles based on the arrangement of gases. These insights highlight the importance of considering molecular characteristics in hydrodynamic simulations, which can have profound implications for modeling real-world phenomena. Our results not only confirm the reliability of the SMASH code for simulating shock physics but also pave the way for future investigations into more complex scenarios, such as the vacuum-gas interface in hydrodynamics.

For future work, we aim to provide a more complete study by considering all permutations of the degrees of freedom for the gases in this problem. We also plan to apply adaptive mesh refinement, and observe the impact of such algorithmic choices on such problems. We also hope to progress, or push, the TIBW to situations where the initial density values are even more disparate; converging to a LeBlanc-like problem. Ultimately, this work contributes to a deeper understanding of shock interactions and enhances the capabilities of multiphysics codes in addressing real-world applications in astrophysics and high-energy density physics.

References.

[1] P. R. Woodward. Trade-Offs in Designing Explicit Hydrodynamical Schemes for Vector Computers. In G. Rodrigue, editor, *Parallel Computations*, pages 153 – 171. Academic Press, 1982.

[2] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, 54(1):115 – 173, 1984.

[3] R. N. Markwick, A. Frank, J. Carroll-Nellenback, B. Liu, E. G. Blackman, S. V. Lebedev, and P. M. Hartigan. Cooling and instabilities in colliding flows. *Monthly Notices of the Royal Astronomical Society*, 508(2):2266–2278, September 2021.

[4] J. L. Brown, C. S. Alexander, J. R. Asay, T. J. Vogler, and J. L. Ding. Extracting strength from high pressure ramp-release experiments. *Journal of Applied Physics*, 114(22):223518, 12 2013.

[5] J. M. Stone, T. A. Gardiner, P. Teuben, J. F. Hawley, and J. B. Simon. Athena: A New Code for Astrophysical MHD. 178(1):137, September 2008.

[6] J. M. Stone, K. Tomida, C. J. White, and K. G. Felker. The Athena++ Adaptive Mesh Refinement Framework: Design and Magnetohydrodynamic Solvers. *The Astrophysical Journal Supplement Series*, 249(1):4, June 2020.

[7] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *The Astrophysical Journal Supplement Series*, 131(1):273, November 2000.

[8] W. L. Oberkampf and T. G. Trucano. Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences*, 38(3):209 – 272, 2002.

[9] W. L. Oberkampf and C. J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, 2010.

[10] W. W. George and A. Migdal. What Went Wrong with Boeing's 737 MAX. *Harvard Business Review*, 2020. Harvard Business School Case 320-104.

[11] J. Thrussell and J. M. Ferguson. ExactPack: A python library of exact analytic solutions. *SoftwareX*, 24:101560, 2023.

[12] J. R. Kamm, J. S. Brock, S. T. Brandon, D. L. Cotrell, B. Johnson, P. Knupp, W. J. Rider, T. G. Trucano, and V. G. Weirs. Enhanced verification test suite for physics simulation codes. *LLNL-TR-411291*, September 2008.

DISCRETE EXTERIOR CALCULUS FOR HODGE-HELMHOLTZ PROBLEM

DOMINIQUE HUGHES*, CHRIS ELDRED†, AND ERIC C. CYR‡

Abstract.
We explore the use of discrete exterior calculus (DEC) as a structure-preserving method for numerically solving PDEs. We briefly explain the DEC forms and operators we will be using. We focus on the Hodge-Helmholtz problem to test the use of DEC. We test examples for all possible forms in the 2-D space; we test double periodic, Dirichlet and Neumann boundary conditions for different forms as well. We also test the addition of a coefficient in the Hodge-Helmholtz problem (often referred to as the Darcy flow problem). We find that most examples successfully converge to the correct solution, however, the implementation of Neumann boundary conditions here leads to some errors.

1. Introduction. Exterior calculus is a system of writing mathematics that has been used recently in structure-preserving numerical methods for partial differential equations (PDEs) (1). Exterior calculus relies on the construction of "forms". These forms are written as a^k, where k represents the type of form in question. You have as many types of forms as you do dimensions in the space you are working in (i.e., a 3-D space has a 0-form, 1-form, 2-form, and 3-form) (1). Because of their construction, these forms and the exterior calculus operators associated with them are particularly useful at capturing the underlying physics and mathematics of a system (1).

The main exterior calculus operators we are using are the exterior derivative, d, and the Hodge-star, $\star$. The exterior derivative takes a form and transforms it to the next higher dimension:

$$da^k = b^{k+1}. \tag{1.1}$$

The Hodge-star takes a form and produces a form in the reflected dimension:

$$\star a^k = b^{N-k}, \tag{1.2}$$

where N is the dimension of the space you are working in. Rather than fully exploring the meaning of these exterior calculus operators here, we will instead define their discrete counterparts, as that is what we will be using in this paper to solve PDE systems.

1.1. Discrete Exterior Calculus (DEC). Using DEC first involves an understanding of grids that we use to model the problem. For DEC, we use a pair of grids that correspond to each other based on some sort of topology and/or geometry rules. The specific rules can change, but the way that they correspond to each other may affect the way you formulate some operators (specifically in this case, the Hodge-star) (2). This pair of grids are often referred to as the primary and dual, or straight and twisted grids (2); we will use straight and twisted terminology. An example of what this mesh might look like is included in figure 1.1.

We can now define the discrete exterior terms we will be using in the context of the mesh. First, the construction of discrete forms: a 0-form is the value of a point on the mesh (a blue or green dot), a 1-form is the value of the line integral for a mesh edge, and a 2-form would be the value of the area integrals for the mesh squares (in the example in figure 1.1 they are squares, but the shape can be different based on grid construction) (1). Some forms

*Sandia National Laboratory, dchughe@sandia.gov
†Sandia National Laboratory, celdred@sandia.gov
‡Sandia National Laboratory, eccyr@sandia.gov

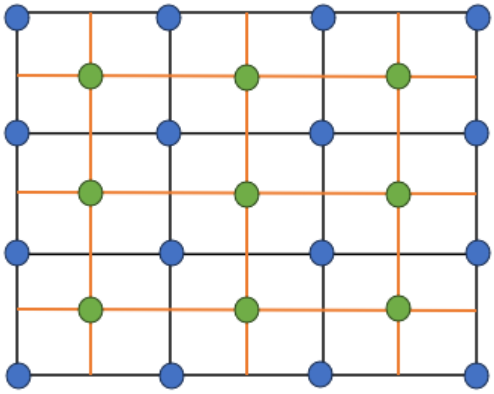


Fig. 1.1: An example of a straight and twisted mesh. The straight mesh is portrayed in blue and black, with the twisted mesh portrayed in orange and green.

have vectors associated with them. The 0-form is always scalar, and the largest dimension form will also be scalar; the forms in between will have associated vectors. This works well for solving PDEs, because some phyical properties are better represented by vectors (like velocity, for example) (1). For more information on this, please refer to (1).

As seen in equation 1.1, the exterior derivative takes a form and produces the next highest form. The discrete exterior derivative produces a k-form by summing the surrounding $(k-1)$-forms (this is done with weights represented by -1 or 1 to help determine and maintain orientation) (1; 2). For example, a 1-form can be computed through the difference between two 0-form values (1).

The Hodge-star, as seen in equation 1.2, takes a k-form on a straight mesh and produces $(N-k)$-form on the twisted mesh (or it can go from twisted to straight). The creation of the Hodge-star is dependent on the way that the grids topology/geometry is determined (2). In this paper, we use the Voronoi Hodge-star, which translates between forms by using the ratio of the volumes between the k-cells associated with the k-forms in question (i.e. a ratio between the k-cell and the $(N-k)$-cell) (2).

Both the discrete exterior derivative and the discrete Hodge-star are represented by sparse matrices (2). We will represent the discrete exterior derivative as D and the discrete Hodge-star as H. To indicate that we are operating on the twisted mesh, we will add a tilde: $\tilde{D}$ or $\tilde{H}$.

1.2. Hodge-Helmholtz Problem. We introduce the Hodge-Helmholtz problem in vector calculus notation. Often referred to as the Poisson problem, it is written as:

$$\nabla^2\phi = \nabla \cdot \nabla\phi = f, \tag{1.3}$$

where we know the function f and we would like to solve for ϕ (3). The general form of the Poisson problem in exterior calculus notation is given as

$$(\delta d + d\delta)\phi^k = f^k, \tag{1.4}$$

where d represents the exterior derivative, and δ represents the codifferential operator, $\star d\star$. We can solve the system for any k-form allowed in the space, as we will demonstrate through this paper.

In practice, the Poisson problem involves additional boundary conditions. These are often Neumann or Dirichlet boundary conditions. Neumann boundary conditions set the first derivative at the boundary equal to a constant and Dirichlet boundary conditions set the value of the function at the boundary equal to a constant. In exterior calculus notation, Neumann conditions would be represented by setting the value of $\delta\phi^k$ or $d\phi^k$ at the boundary, and Dirichlet conditions would be represented by setting the value of ϕ^k or $\star\phi^k$ at the boundary.

Additionally, there can be an additional element added to the Hodge-Helmholtz problem: a coefficient $\mathbb{K}$. The addition of $\mathbb{K}$ to the Poisson problem is often referred to as the Darcy flow problem (3), and it is represented as

$$\nabla \cdot \mathbb{K}\nabla\phi = f \Leftrightarrow d\mathbb{K}\delta\phi^k = f^k. \tag{1.5}$$

$\mathbb{K}$ might represent material coefficients for a fluid dynamics problem, for example (3).

2. 2-D Results. In each of the following cases, we solve the system numerically on a mesh going from 0 to 1 in both the x and y direction, with the center at (0.5, 0.5). Boundary conditions will be specified for each case. We solve our discrete system numerically at different mesh resolutions, starting with 20x20, and doubling our previous mesh size until 640x640. We compute both the L^2-norm and the L^∞-norm on the difference between the numerical and real solution as a measure of error at each grid resolution.

2.1. 0-forms. In the straight 0-form case, the Hodge-Helmholtz problem reduces to:

$$\delta d\phi^0 = f^0. \tag{2.1}$$

Written discretely, we have

$$\tilde{H}\tilde{D}HD\phi^0 = f^0. \tag{2.2}$$

We look at the case of Neumann boundaries in a cylinder, meaning that we maintain periodic boundary conditions in one direction and add Neumann boundary conditions in the other direction. In this case, we will look at an example that is periodic in the x-direction, with Neumann conditions at the y-boundary. In order to add Neumann boundary conditions, we split the discrete exterior derivative on the twisted mesh into the interior and boundary components, and include a term to set the boundary conditions, $\tilde{B}^{n-1}$:

$$\tilde{H}\tilde{D}_I HD\phi^0 + \tilde{H}\tilde{D}_B\tilde{B}^{n-1} = f^0. \tag{2.3}$$

We move the term $\tilde{H}\tilde{D}_B\tilde{B}^{n-1}$ to the right hand side and solve for ϕ^0. In this case, we try two potential functions that are periodic in the x-direction:

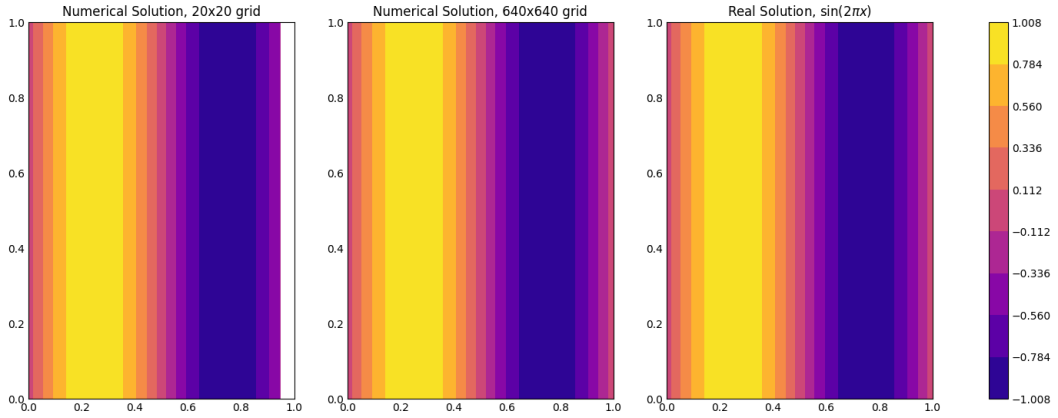
$$f^0 = -4\pi^2 \sin(2\pi x), \tag{2.4}$$

$$f^0 = -4\pi^2 y^2 \sin(2\pi x) + 2\sin(2\pi x), \tag{2.5}$$

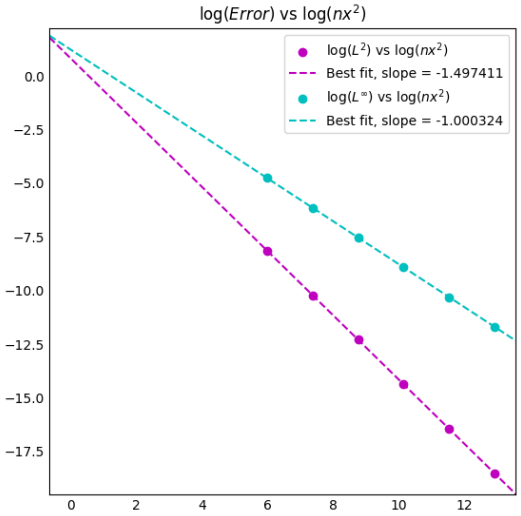
which gives the respective analytical solutions

$$\phi^0 = \sin(2\pi x), \tag{2.6}$$

$$\phi^0 = y^2 \sin(2\pi x). \tag{2.7}$$



(a)

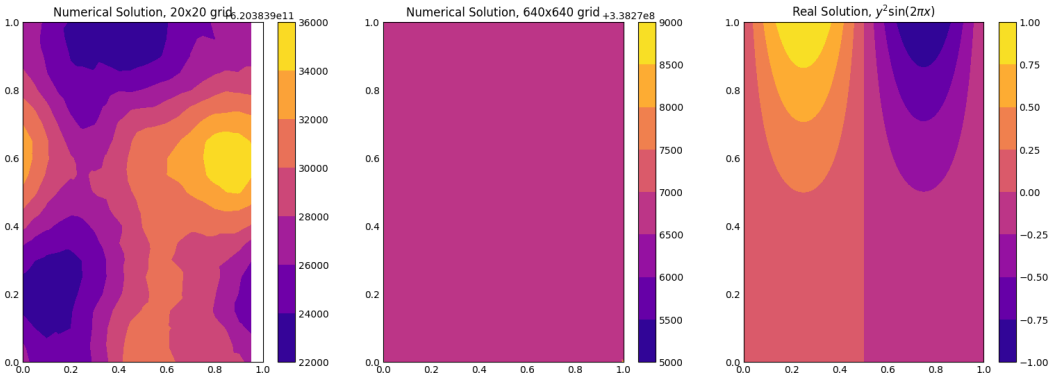


(b)

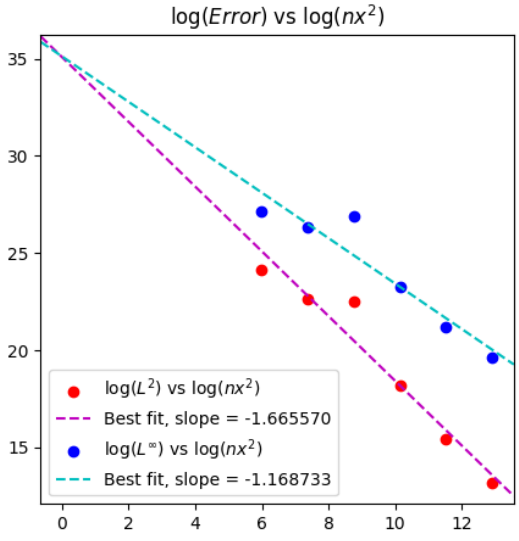
Fig. 2.1: (a) Plots comparing numerical solution at the lowest and highest resolution grid size, as well as providing the analytical solution as reference, for the 2.4/2.6 equations. (b) Plot depicting the log of the error with respect to the log of the total number of grid points.

We look at the results of equations 2.4/2.6 in figure 2.1. We can see in figure 2.1b that we converge to the correct solution linearly in the log scale, with a slope of ≈ -1 for the L^∞-norm, and a slope of ≈ -1.5 for the L^2-norm. This is reflected in figure 2.1a, where we see that even at the 20x20 grid we are getting a fairly good approximation of the solutions.

For equations 2.5/2.7, however, we see in figure 2.2 that we do not achieve the same convergence. Despite the error going down in 2.2b, we can see because it is not linear that it is not decreasing due to a better approximation of the solution. This is confirmed in 2.2a, where we can see that the solution on the 20x20 and 640x640 grid are completely different in shape and scale, and neither match the analytical solution.



(a)



(b)

Fig. 2.2: (a) Plots comparing numerical solution at the lowest and highest resolution grid size, as well as providing the analytical solution as reference, for the 2.5/2.7 equations. (b) Plot depicting the log of the error with respect to the log of the total number of grid points. Plotting the points in red and blue respectively represents no convergence for that grid mesh size.

We see similar results in the case of a cylinder with y-periodic/x-Neumann conditions, and we see no convergence in the case of Neumann boundaries on both x and y. This demonstrates that there is some error going on in our computation of the boundary conditions. We are investigating where this error is being introduced into our problem.

2.2. 1-forms. In the straight 1-form case, the Hodge-Helmholtz problem is

$$(\delta d + d\delta)\phi^1 = f^1. \tag{2.8}$$

Written discretely, we have

$$(\tilde{H}\tilde{D}HD + D\tilde{H}\tilde{D}H)\phi^1 = f^1. \tag{2.9}$$

We use double periodic boundary conditions in this case. Because 1-forms in the 2-D space are vectors with two components, we chose the function f to be

$$f^1 = \begin{bmatrix} -20\pi^2 \sin(2\pi x)\sin(4\pi y) \\ -20\pi^2 \cos(2\pi x)\cos(4\pi y) \end{bmatrix} \tag{2.10}$$

so that our analytical solution must be

$$\phi^1 = \begin{bmatrix} \sin(2\pi x)\sin(4\pi y) \\ \cos(2\pi x)\cos(4\pi y) \end{bmatrix}. \tag{2.11}$$

We see in figure 2.3b that we converge to the correct solution linearly on a log scale, with a slope of ≈ -1.3 for the L^2-norm, and ≈ -0.4 for the L^∞-norm. We can see in 2.3a that we have an extremely close approximation of our analytical solution for each vector component.

2.3. (n-1)-forms. In the twisted $(n-1)$-form case, we write the Hodge-Helmholtz problem as

$$(\delta d + d\delta)\tilde{\phi}^{n-1} = \tilde{f}^{n-1}. \tag{2.12}$$

Because we are working in the 2-D space, an $(n-1)$ form is also a 1-form, but it is constructed slightly differently than the 1-form of the previous section. In this case, a 1-form represents a circulation along an edge, whereas an $(n-1)$-form represents a flux along an edge (2). The 1-form is constructed through the tangent vector along an edge and the $(n-1)$-form is constructed through the normal vector to an edge (2). To maintain clarity, we will continue to use the $(n-1)$-form notation.

We can write this discretely as

$$(HD\tilde{H}\tilde{D} + \tilde{D}HD\tilde{H})\tilde{\phi}^{n-1} = \tilde{f}^{n-1}. \tag{2.13}$$

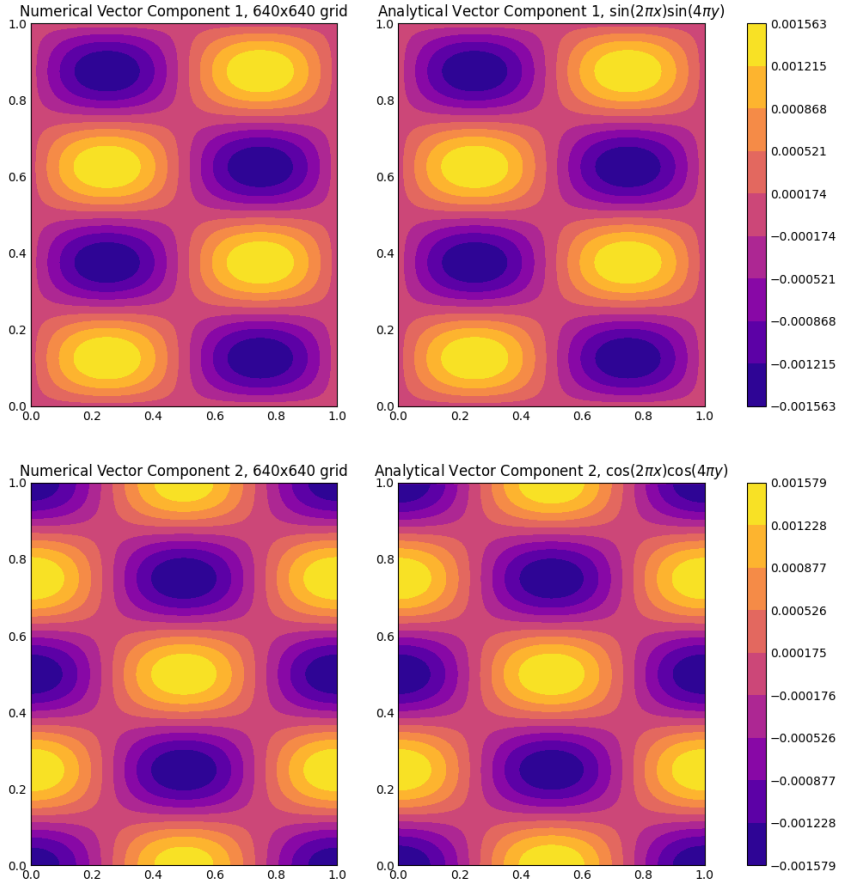
Once again, we will look at a double periodic boundary conditions case, with our function $\tilde{f}^{n-1}$ as a vector with two components,

$$\tilde{f}^{n-1} = \begin{bmatrix} -20\pi^2 \sin(2\pi x)\cos(4\pi y) \\ -20\pi^2 \sin(2\pi y)\cos(4\pi x) \end{bmatrix} \tag{2.14}$$

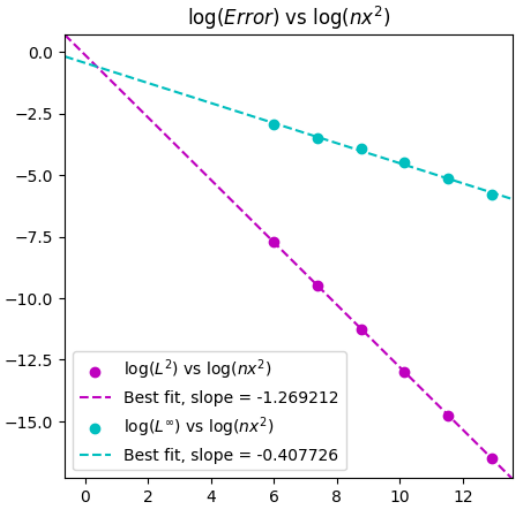
so that our analytical solution is

$$\tilde{\phi}^{n-1} = \begin{bmatrix} \sin(2\pi x)\cos(4\pi y) \\ \sin(2\pi y)\cos(4\pi x) \end{bmatrix}. \tag{2.15}$$

We see in figure 2.4b that we converge to the solution linearly on a log scale, with slopes that are very close to those in the 1-form case (≈ -1.3 for L^2 and ≈ -0.4 for L^∞). We see again in 2.4a that we have an extremely good approximation of our analytical solution at the 640x640 mesh grid.

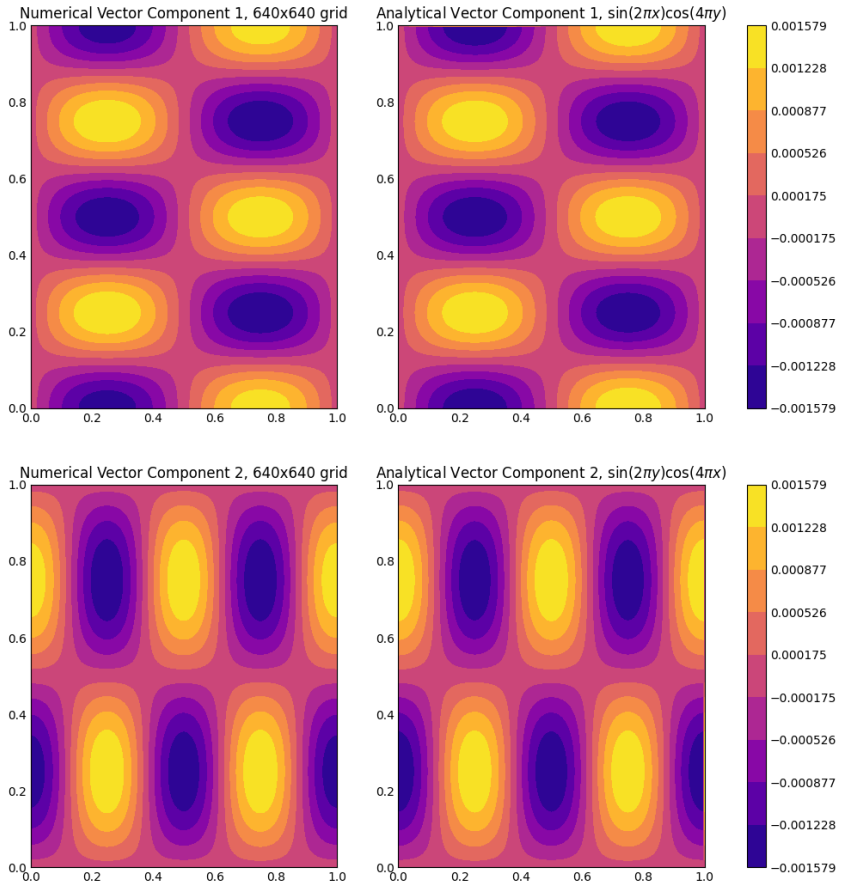


(a)

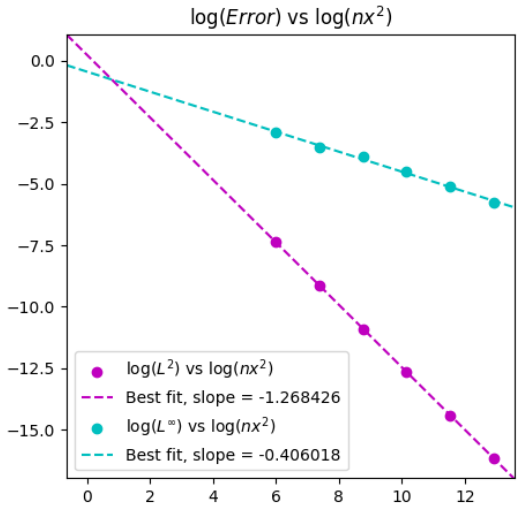


(b)

Fig. 2.3: (a) Plots comparing each vector component in the 1-form at the highest grid resolution to the analytical as well as providing the analytical solution as reference. (b) Plot depicting the log of the error with respect to the log of the total number of grid points.



(a)



(b)

Fig. 2.4: (a) Plots comparing each vector component in the $(n-1)$-form at the highest grid resolution to the analytical as well as providing the analytical solution as reference. (b) Plot depicting the log of the error with respect to the log of the total number of grid points.

2.4. n-forms. In the straight n-form, or straight volume-form case, the Hodge-Helmholtz problem reduces to

$$d\delta\phi^n = f^n. \tag{2.16}$$

Written discretely, we have

$$D\tilde{H}\tilde{D}H\phi^n = f^n. \tag{2.17}$$

Given we are working in 2-D, $n = 2$. We will look at an example with Dirichlet boundary conditions. In order to set Dirichlet boundary conditions, we add a boundary term $\tilde{B}^0$ to the formation of the problem. To incorporate this boundary term, we split the twisted exterior derivative $\tilde{D}$ into its boundary and interior components. We rewrite the problem as such:

$$D\tilde{H}\tilde{D}_I H\phi^n + D\tilde{H}\tilde{D}_B\tilde{B}^0 = f^n \tag{2.18}$$

We choose our function

$$f^n = 2y^2 + 2x^2 - 2\sin(x+y), \tag{2.19}$$

so that our analytical solution must be

$$\phi^n = x^2y^2 + sin(x+y). \tag{2.20}$$

We can see in figure 2.5b that the Dirichlet n-form case converges linearly on the log scale with increasingly fine mesh grids; the slope of the L^2-norm is ≈ -2.5 and the slope of the L^∞-norm is ≈ -2. We can see in figure 2.5a that at the 20x20 mesh grid, we have a fairly good approximation of the solution, but as the mesh grid is refined the boundary more closely matches the analyical solution.

2.5. Darcy flow. As mentioned in the introduction, a common variation of the Hodge-Helmholtz problem is the Darcy flow problem, which has the addition of a multiplied coefficient. We look at a Darcy flow problem listed in (3) for inspiration. We do this on a twisted volume-form set up:

$$\tilde{D}HD\tilde{H}\tilde{\phi}^n = \tilde{f}^n \tag{2.21}$$

For this example, we use Neumann boundary conditions. To implement Neumann boundary conditions in the n-form case, we record the boundary conditions in a twisted (n-1)-form, $\tilde{B}^{n-1}$. We incorporate these terms into the problem formulation by splitting the twisted exterior derivative into its boundary and interior components:

$$\tilde{D}_I HD\tilde{H}\tilde{\phi}^n + \tilde{D}_B\tilde{B}^{n-1} = \tilde{f}^n. \tag{2.22}$$

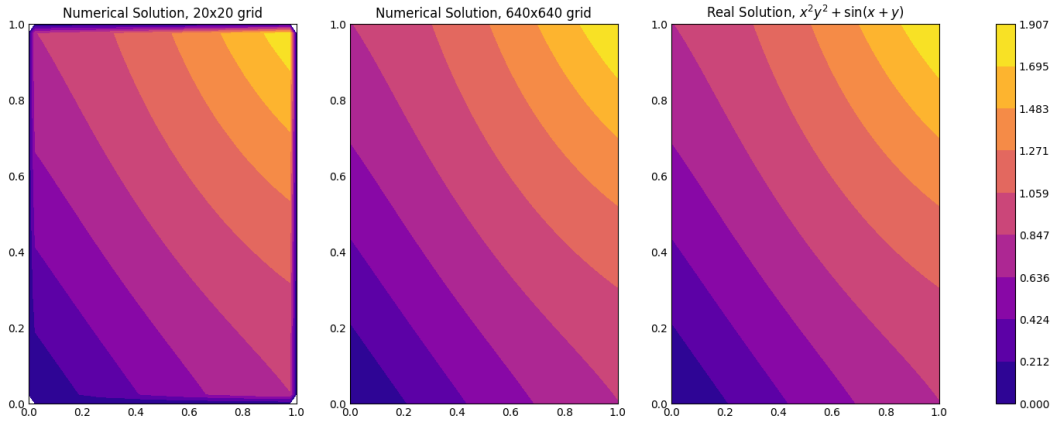
We rewrite the problem including the added coefficient, α:

$$\tilde{D}_I\alpha HD\tilde{H}\tilde{\phi}^n + \tilde{D}_B\tilde{B}^{n-1} = \tilde{f}^n, \tag{2.23}$$

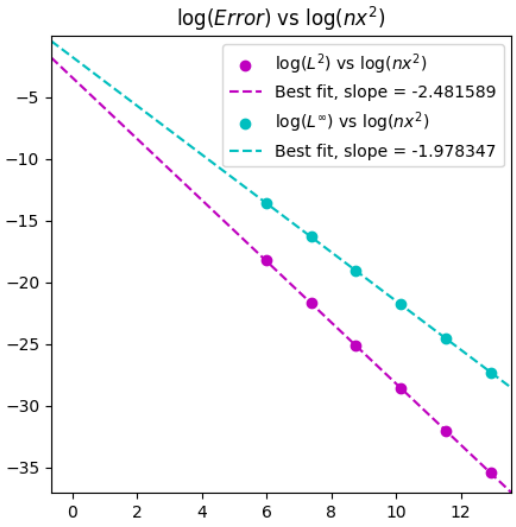
and it is understood from this formulation that our $\tilde{B}^{n-1}$ term must reflect the gradient of $\tilde{\phi}^n$ multiplied by the coefficient α in order to accurately represent the Neumann boundary conditions. Following the example from (3), we choose our analytical solution to be

$$\tilde{\phi}^n = x^3y^4 + x^2 + \sin(xy)\cos(y). \tag{2.24}$$

We choose $\alpha = \sin(xy)$ as our coefficient. We can see in figure 2.6b that we only have convergence to the solution for our final three mesh grids, 160x160, 320x320 and 640x640. The coarser grids do not converge correctly. However, when we visualize the numerical solution for the 640x640 mesh grid in figure 2.6a, we see that though it seems to capture the dynamics correctly, the magnitude of the solution is incorrect. We are investigating what might be the cause of this type of error.



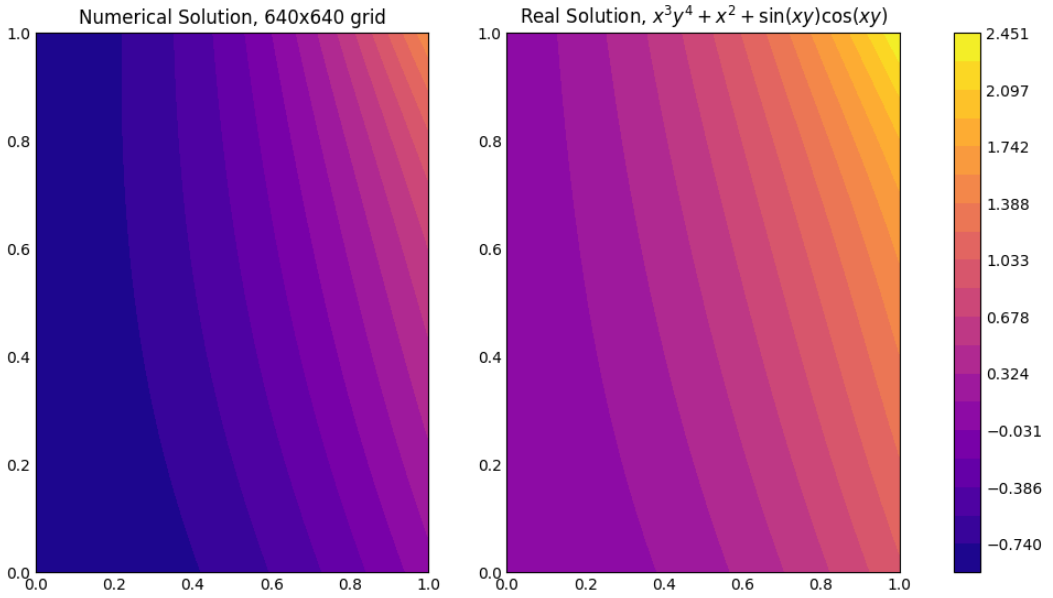
(a)



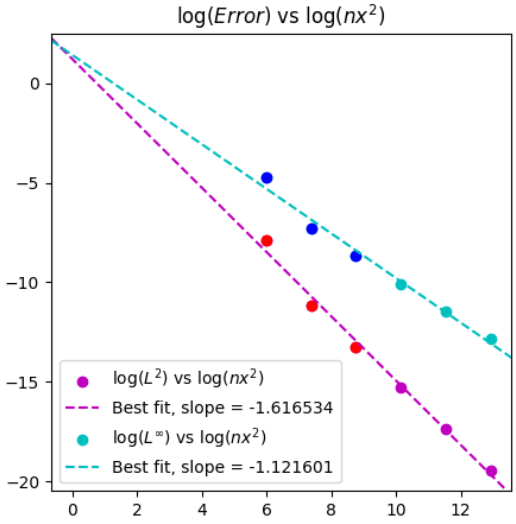
(b)

Fig. 2.5: (a) Plots comparing numerical solution at the lowest and highest resolution grid size, as well as providing the analytical solution as reference. (b) Plot depicting the log of the error with respect to the log of the total number of grid points.

3. Conclusion. In this paper, we explored the use of DEC to numerically solve PDEs, specifically the Hodge-Helmholtz equation. We looked at solving the Hodge-Helmholtz equation for different types of forms, as well as with different boundary conditions. We also looked a Darcy flow case, with an added coefficient. We saw that our double periodic and Dirichlet boundary examples were successful, with the numerical solution closely matching the analytical solution; this is confirmed by both L^∞-norm and L^2-norm calculations, as well as visual displays of both the analytical and numerical solution. However, we notice that the addition of Neumann boundary conditions in particular seems to cause an error.



(a)



(b)

Fig. 2.6: (a) Plots comparing numerical solution at the lowest and highest resolution grid size, as well as providing the analytical solution as reference. (b) Plot depicting the log of the error with respect to the log of the total number of grid points. Plotting the points in red and blue respectively represents no convergence for that grid mesh size.

In the 0-form case, we saw a successful solution in the case of 2.4/2.6, but the addition of the y^2 term in the 2.5/2.7 prevented convergence to the solution. In the Darcy flow case, we saw that our solution converged only for certain grids. Additionally, though the numerical solution appeared to be very similar to the analytic solution, we can see that the magnitude of the values is incorrect. As such, we will be investigating why we receive these errors with the Neumann boundaries in particular.

Once the Neumann boundary implementation has been fixed, we plan to explore other Darcy flow problems. Specifically, we will look at cases where the coefficient α is discontinuous. These are good test problems for DEC, because discontinuity in the coefficient is a good measure to ensure that a method is structure-preserving. Additionally, all of these examples occur on the same square mesh set up. In the future, we would like to try using DEC to solve PDE systems on more complex shapes.

References.

[1] J. Blair Perot and C. J. Zusi, *Differential forms for scientists and engineers*, Journal of Computational Physics, 257 (2014), pp. 1373–1393. Physics-compatible numerical methods.

[2] C. Eldred and W. Bauer, *An interpretation of trisk-type schemes from a discrete exterior calculus perspective*, 2022.

[3] J. J. Kreeft, *Chapter 8: Hodge-Laplace problems*, s.n., 2013, p. 143–176.

A DMD-BASED PARTITIONED SCHEME FOR TIME-DEPENDENT COUPLED PARAMETRIC PDES

EDWARD HUYNH*, PAVEL BOCHEV†, AND PAUL KUBERRY‡

Abstract.
Partitioned methods for coupled problems rely on data transfers between subdomains to synchronize the subdomain equations and enable their independent solution. These methods enable code reuse, increase concurrency, and provide a convenient framework for plug-and- play multiphysics simulations. However, accuracy and stability of partitioned methods depends crucially on the type of information exchanged between the subproblems. These transfer mechanisms are separated by accuracy, performance and intrusiveness gaps that tend to limit the scope of the resulting partitioned methods to specific simulation scenarios. Data-driven system identification methods allow the construction of accurate, computationally efficient and minimally intrusive data transfer surrogates. Moreover, they also serve as a way to evolve the individual subproblems in time. This approach leaves the main computational burden to an offline phase and consequently the application of this approach as the sole additional cost during the online simulation phase. In this paper we formulate and demonstrate a fully data-driven approach for the subdomain solvers which are together coupled with a dynamic flux surrogate-based partitioned method for a model advection-diffusion transmission problem by using Dynamic Mode Decomposition (DMD) and Dynamic Mode Decomposition wth Control (DMD with Control) to learn the dynamics of both the subdomain states and the interface flux from data.

1. Introduction. In general, there are two approaches to solving coupled multiphysics problems [4]. *Monolithic methods* treat the coupled system as a a single entity and advance all of its constituent physics components in time simultaneously [4]. Typically this is accomplished by forming and solving a well-posed monolithic problem in which the coupling conditions are enforced by, e.g., Lagrange multipliers, or by using shared basis functions that are continuous across the interface [2]. Consequently, monolithic methods inherently possess excellent stability and accuracy properties but suffer from being more computationally intensive.

On the other hand, *Partitioned methods* treat the sub-problems comprising the multiphysics system independently and evolve each of them in parallel [4]. Synchronization between the sub-problems is performed through data transference, as in [2]. Partitioned schemes, as a consequence, are more flexible than monolithic ones. In particular, this framework enables practitioners to potentially use different solvers that are uniquely optimized and specialized to the characteristics of each sub-problem.

One way of developing novel partitioned methods is by beginning with a well-posed monolithic formulation. One then uses techniques such as Schur complements to recover highly accurate approximations of the interface fluxes which are typically characterized as the Lagrange multipliers. These fluxes provide boundary conditions that "close" the equations for the constituent physics components and enable their independent advancement in time. That is, once the governing equations have been discretized (such as with finite elements), then the discrete system is advanced using a time integrator. For instance, partitioned schemes have been employed to solve several types of coupling problems involving advection-diffusion equations with varying diffusion regimes [7] [3]. In particular, [3] applies the IVR scheme [7] to reconstruct the interface fluxes to FEM-FEM and ROM-ROM type couplings which are then advanced forward in time using an explicit time integrator, e.g forward Euler, RK4, etc.

There are two different aspects that are important in developing partitioned methods:

1. How to obtain the interface fluxes

*University of Arizona, edhuynh@sandia.gov
†Sandia National Labs, pbboche@sandia.gov
‡Sandia National Labs, pakuber@sandia.gov

2. How to solve each of the coupled problems independently

With respect to the former, there are different methods to obtain the fluxes. One common way is to construct the Schur complement. The IVR (implicit value recovery) scheme developed in [7] has been shown to yield accurate and reliable values of the interface fluxes which may be used to "plug" into the discretized system and then effectively uncouple the governing equations using an explicit time integrator. The IVR scheme is a representative example of reconstruction-based schemes which recover the flux – as opposed to remap-based ones. Other approaches may be based on data driven techiques, such as the Dynamic Flux Surrogate model as proposed in [1].

However, the accuracy and stability properties that are attained by the more precise flux estimates in reconstruction-based partitioned methods incur large costs in the form of additional memory requirements and computational burdens required to form and solve the resulting linear system. IVR [7] and IFR (Interface Flux Recovery) [9] sometimes involve lumped mass matrices which lead to manageable costs due to a sparse dual Schur complement. However, in many other cases realization of the full accuracy potential of these methods requires consistent subdomain mass matrices. The inversion of these matrices results in a dense Schur complement which results in larger memory capacities. At the same time, forming the right hand sides of the associated linear systems for the fluxes yields an additional computational cost. These schemes are also intrusive since they generally require access to the subdomain equations.

As for the latter point, partitioned methods allow for each sub-problem to be solved independently using the method of the practitioners choice, while incorporating the transfer of data within each subdomain through the interface fluxes. While explicit time integrators are used more readily due to their ease of use and because they uncouple the discretized system (as in the case of the IVR scheme), one may also use implicit schemes. However, these are usually too expensive in terms of memory requirements to run, especially for large-scale simulations as in Earth climate modeling.

One way to circumvent the shortcomings in addressing these two aspects is through data-driven system identification techniques. These methods provide a way to close these gaps by enabling the construction of accurate yet computationally efficient and minimally intrusive surrogates for both the dynamics of the interface flux and the subdomain solvers. With this approach, the main computational cost is relegated to an offline phase while leaving the application of the surrogate as the only cost during the online simulation phase. Using only a few matrix-vector multiplications during each step of the online phase, the cost is comparable to that of a remap-based flux methods via linear maps [10]. Moreover, these methods are minimally intrusive because learning the surrogate does not require access to the discretized equations. Finally, using curated data training sets, the surrogate models can produce flux approximations and subdomain solvers whose accuracies are comparable to the ones obtained through, e.g., solution of the dual Schur complements or explicit/implicit time integrators. Formulation of a fully data-driven partitioned scheme requires three key ingredients:

1. A time stepping harness comprising a set of synchronization points in time and a choice of time integration schemes for each subproblem;
2. A model form for performing system identification of the dynamic flux surrogate;
3. A model form for performing system identification of the subdomain solvers.

Even so, this fully data-driven partitioned method is not necessarily intended intended to simulate a given coupled system with arbitrary choices in the boundary and initial data. Instead, this approach is meant to solve parameterized problems in which the states of the system are assumed to have a dependence on a finite dimensional parameter $\mu \in \mathbb{R}^m$. Thus, in this manuscript we develop and demonstrate a fully data-driven partitioned method for

coupled parameterized Partial Differential Equations (μPDEs).

In this manuscript, we provide an initial demonstration for this class of partitioned methods and restrict our attention to a simple, explicit, synchronous time-stepping framework in which all subproblems are advanced in time using identical times steps and the same explicit time integrator. One candidate from which this data may be drawn from is the advection-diffusion equation where the equations are discretized using finite elements and the states are the coefficient vectors to the finite element basis functions. The model form for the subdomain solvers will be using a traditional linear dynamical system with control inputs and then obtain the resulting model using the Dynamic Mode Decomposition with control inputs (DMD with Control) [6, 8]. Similarly, the model form for the flux surrogate is chosen similarly and will be obtained using Dynamic Mode Decomposition (DMD) [6] to identify this model. In fact, we will use the dynamic flux surrogate model as proposed by [1] to obtain the interface fluxes.

To begin our preliminary investigation into the viability of such models, we first consider the case of a coupled μPDE problem with a fixed parameter value and show that, with proper training data, the DMD flux surrogate (DMD-FS) [1] together with the DMD subdomain solvers (DMDc) can be constructed offline to accurately represent the dynamics of both the subdomain states and the interface flux for initial conditions not included in the training datasets. In this manuscript, we restrict our attention to a single parameter for μ and reserve extension of this scheme to the fully-parametric setting in future work. To our knowledge, this work is the first to consider data driven solvers for both the subdomain states and the interface fluxes as a way to improve the accuracy and efficiency of partitioned schemes.

The rest of this article is organized as follows. Section 2 introduces the relevant notation and technical background. Here, we will describe the governing model from which we gather our data. For the convenience of the reader, in subsections 2.2 and 2.3, we also review the classic DMD approach [6] and the DMDc model [8]. We then seque into a discussion of recent data-driven techniques in coupling problems in Section 3, including the IVR scheme and Dynamic Flux Surrogate model in sections 3.1 and 3.2 respectively. Moreover, Section 3.3 will briefly describe the novel model used to approximate the dynamics of the system. Section 4 will describe the numerical experiments that we used to simulate and validate the fully data driven approach and it will also analyze and discuss our results. Finally, Section 5 will summarize our findings and offer future directions for research.

2. Notation and Background. We will summarize the notation used in the manuscript. Section 2.1 describes the model problem from which we extract data. Sections 2.2 and 2.3 introduce the DMD and DMDc models.

2.1. The Model Problem. We will start by introducing the monolithic formulation from which we develop a partitioned method. Let $\Omega \subset \mathbb{R}^\nu$, $\nu = 2, 3$ be a bounded region with Lipschitz continuous boundary Γ. We assume that Ω is divided into two non-overlapping subdomains Ω_1 and Ω_2 which are connected through an interface γ. Consider the advection-diffusion problem:

$$\begin{aligned} \frac{\partial u_i}{\partial t} - \nabla \cdot F_i(u_i) = f_i, \quad \Omega_i \times [0,T] \\ u_i = g_i, \quad \Gamma_i \times [0,T],\ i = 1,2 \\ u_i = u_{i,0}, \quad \Omega_i, t = 0 \end{aligned} \tag{2.1}$$

which satisfies the continuity conditions

$$\dot{u}_1(\boldsymbol{x},t) - \dot{u}_2(\boldsymbol{x},t) = 0 \quad \text{and} \quad F_1(\boldsymbol{x},t)\cdot \boldsymbol{n}_\gamma = F_2(\boldsymbol{x},t)\cdot \boldsymbol{n}_\gamma \quad \text{on } \gamma \times [0,T].$$

Here, $F_i(u_i) = \kappa_i \nabla u_i - \boldsymbol{a} u_i$ is called the total flux function, $f_i = f_i(\boldsymbol{x}, t)$ is a source, $g_i = g_i(\boldsymbol{x}, t)$ is prescribed boundary data, $u_{i,0} = u_{i,0}(\boldsymbol{x})$ is a prescribed initial condition, $\kappa_i = \kappa_i(\boldsymbol{x}, t) > 0$ is the diffusion coefficient, and $\boldsymbol{a} = \boldsymbol{a}(\boldsymbol{x}, t)$ is the advection field.

The corresponding weak formulation is posed as follows: seek $\{u_1, u_2, \lambda\} : (0, T] \mapsto H^1_D(\Omega_1) \times H^1_D(\Omega_2) \times H^{-1/2}(\gamma)$ such that for $u_i = u_{i,0}$ for $t = 0$, $i = 1, 2$, and for $t > 0$

$$
\begin{aligned}
(\dot{u}_1, v_1)_{0,\Omega_1} + \langle \lambda, v_1 \rangle_\gamma &= (f_1, v_1)_{0,\Omega_1} - (F(u_{1,D}), \nabla v_1)_{0,\Omega_1} - Q_1(\dot{g}_1, g_1; v_1), \quad \forall v_1 \in H^1_\Gamma(\Omega_1) \\
(\dot{u}_2, v_2)_{0,\Omega_2} - \langle \lambda, v_2 \rangle_\gamma &= (f_2, v_2)_{0,\Omega_2} - (F(u_{2,D}), \nabla 2_1)_{0,\Omega_2} - Q_2(\dot{g}_2, g_2; v_2), \quad \forall v_2 \in H^1_\Gamma(\Omega_2) \\
\langle \dot{u}_{1,D} - \dot{u}_{2,D}, \mu \rangle_\gamma &= \langle \dot{g}_1 - \dot{g}_2, \mu \rangle_\gamma \quad \forall \mu \in H^{-1/2}(\gamma).
\end{aligned}
$$

Here, Q_i are the bilinear forms which represent the contributions of the boundary data to the right-hand side. This problem has been proven to be well-posed.

We approximate the solutions u_1, u_2, λ by finite element spaces $S^h_1 \times S^h_2$ and G^h_k for $k = 1, 2$. We will define the following notation for each $i = 1, 2$. Let $u^h_i = u^h_{i,D} + g^h_i$, where u^h_i represents the unknown part of u^h_i in the interior and interface nodes and g^h_i is the finite element interpolant of the boundary data. Let the coefficient vector of u^h_i be given by $\boldsymbol{u}_i = (\boldsymbol{u}_{i,D}, \boldsymbol{g}_i)$, where $\boldsymbol{u}_{i,D}$ contains the unknown coefficient values while $\boldsymbol{g}_i$ denotes the known nodal values of the boundary data g_i. We denote Ω^h_i to mean a conforming quasi-uniform partition of Ω_i into finite elements $K_{i,s}$ with vertices $x_{i,r}$ and mesh parameter h_i. Let the total number of nodes in Ω^h_i be n_i. Each Ω^h_i induces a conforming mesh Γ^h_i on the Dirichlet boundary Γ_i with $n_{i,\Gamma}$ nodes. We will further assume that Ω_1 and Ω_2 are meshed independently. As a result, their finite element partitions Ω^h_1 and Ω^h_2 induce two independent finite element meshes on the interface γ denoted by γ^h_1 and γ^h_2, respectively, with $n_{i,\gamma}$ nodes each. For simplicity we restrict attention to spatially coincident discrete interfaces, however, we note that the nodes on γ^h_1 and γ^h_2 are not required to match.

From this, we obtain the matrix system

$$
\begin{bmatrix} M_{1,D} & 0 & G^T_{1,D} \\ 0 & M_{2,D} & -G^T_{2,D} \\ G_{1,D} & -G_{2,D} & 0 \end{bmatrix}
\begin{bmatrix} \dot{\boldsymbol{u}}_{1,D} \\ \dot{\boldsymbol{u}}_{2,D} \\ \boldsymbol{\lambda} \end{bmatrix}
=
\begin{bmatrix} \boldsymbol{f}_{1,D} - F_{1,D}\boldsymbol{u}_{1,D} - Q_{1,\Gamma}(\dot{\boldsymbol{g}}_1, \boldsymbol{g}_1) \\ \boldsymbol{f}_{2,D} - F_{2,D}\boldsymbol{u}_{2,D} - Q_{2,\Gamma}(\dot{\boldsymbol{g}}_2, \boldsymbol{g}_2) \\ -Q_{\gamma,\Gamma}(\dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2) \end{bmatrix}.
$$

2.2. Dynamic Mode Decomposition. Dynamic Mode Decomposition [6] describes a method to infer the evolution operator of a dynamical system using collected time series data $\boldsymbol{x}(t)$ which are also called *snapshots*. Suppose we have s equally spaced snapshots $\boldsymbol{x}_i = \boldsymbol{x}(t_i)$ which satisfies

$$
\boldsymbol{x}_{i+1} = D\boldsymbol{x}_i
$$

for some dynamical operator D. Note that D can either be linear or nonlinear. Then DMD finds an operator A such that $D \approx A$ with

$$
\boldsymbol{x}_{i+1} \approx A\boldsymbol{x}_i \tag{2.2}
$$

To infer the DMD operator A, we set

$$
\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \dots & \boldsymbol{x}_{s-1} \end{bmatrix} \quad \text{and} \quad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_s \end{bmatrix}
$$

so that

$$
\boldsymbol{Y} = A\boldsymbol{X}.
$$

From this, we have $A = \boldsymbol{Y}\boldsymbol{X}^+$, where $\boldsymbol{X}^+$ is the pseudo-inverse of $\boldsymbol{X}$. Equivalently, the operator A is also the least-squares solution to

$$\min \|\boldsymbol{Y} - A\boldsymbol{X}\|_F$$

where the norm is taken under the Frobenius norm.

In a practical setting, the pseudo-inverse can be approximated by using a truncated SVD of $\boldsymbol{X}$. To wit, given $\boldsymbol{X} = U\Sigma V^T$ with n singular values, then by retaining the first $k \leq n$ left singular vectors corresponding to the leading k singular values, then

$$\boldsymbol{X}^+ = V\Sigma^+ U^T \approx V_k \Sigma_k^+ U_k^T = \boldsymbol{X}_k^+.$$

Using this approximation to the psuedo-inverse, then we have the approximation to A:

$$A = \boldsymbol{Y}\boldsymbol{X}^+ \approx \boldsymbol{Y}\boldsymbol{X}_k^+ = A_k.$$

A commonly accepted standard to measure accuracy of this operator is through the criterion defined by

$$1 - E_k(\boldsymbol{X}) \leq \epsilon \tag{2.3}$$

where $0 < \epsilon < 1$ is a prescribed tolerance, k is the smallest positive integer such that (2.3) is satisfied and

$$E_k(\boldsymbol{X}) := \frac{\sum_{i=1}^k \sigma_i}{\sum_{i=1}^n \sigma_i}$$

is the *relative snapshot energy* where σ_i is the ith singular value.

2.3. Dynamic Mode Decomposition with Control Inputs. The Control-Input model was first developed in [8] and is given as

$$\boldsymbol{u}^{i+1} = A\boldsymbol{u}^i + B\boldsymbol{\lambda}^i.$$

The model has been used to numerically simulate various physical models, including ones for photoelectric current [5]. Here, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times p}$. To solve for A and B, we treat the problem as a similar one to a DMD one: let

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{u}^0 & \boldsymbol{u}^1 & \dots & \boldsymbol{u}^{n_S-1} \\ \boldsymbol{\lambda}^0 & \boldsymbol{\lambda}^1 & \dots & \boldsymbol{\lambda}^{n_S-1} \end{bmatrix}$$

and

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{u}^1 & \boldsymbol{u}^2 & \dots & \boldsymbol{u}^{n_S} \end{bmatrix}$$

and let $\tilde{A} = \begin{bmatrix} A & B \end{bmatrix}$. Then the system can be expressed as

$$\boldsymbol{Y} = \tilde{A}\boldsymbol{X}.$$

Using a similar technique to the previous section and letting $\boldsymbol{X} = U\Sigma V'$, then $\tilde{A}$ is obtained from the formula

$$\tilde{A} = \boldsymbol{Y}V\Sigma^+ U.$$

Moreover, the operator $\tilde{A}$ may be found from solving

$$\min_{\tilde{A}} \|\boldsymbol{Y} - \tilde{A}\boldsymbol{X}\|_F$$

where the norm is taken with respect to the Frobenius norm. The matrices A and B are obtained from $\tilde{A}$ by taking appropriate submatrices.

For the purposes of coupling, the subdomain DMD operators that are used to obtain the states in $\boldsymbol{u}_{1,D}$ and $\boldsymbol{u}_{2,D}$ are given by A_1^{CI} and A_2^{CI} respectively. Thus, the two subdomain DMD models are given by

$$\begin{cases} \boldsymbol{u}_1^i = A_1^{CI}\boldsymbol{u}_1^{i-1} + B_1^{CI}\boldsymbol{\lambda}^{i-1} \\ \boldsymbol{u}_2^i = A_2^{CI}\boldsymbol{u}_2^{i-1} + B_2^{CI}\boldsymbol{\lambda}^{i-1} \end{cases}.$$

3. Fully Data-Driven Model for Coupling Problems. Here, we discuss the methodology by which we construct the partitioned scheme for coupled problems.

3.1. IVR Scheme. The IVR scheme was first formulated [7] as a way to accurately reconstruct the fluxes in a coupled advection-diffusion problem. First, we solve for $\boldsymbol{\lambda}$ through the equation

$$\begin{aligned} S\boldsymbol{\lambda} = G_{1,D}M_{1,D}^{-1}[\boldsymbol{f}_{1,D} - F_{1,D}\boldsymbol{u}_{1,D} - Q_{1,\Gamma}(\dot{\boldsymbol{g}}_1, \boldsymbol{g}_1)] \\ -G_{2,D}M_{2,D}^{-1}[\boldsymbol{f}_{2,D} - F_{2,D}\boldsymbol{u}_{2,D} - Q_{2,\Gamma}(\dot{\boldsymbol{g}}_2, \boldsymbol{g}_2)] + Q_{\gamma,\Gamma}(\dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2), \end{aligned}$$

where $S = G_{1,D}M_{1,D}^{-1}G_{1,D}^T + G_{2,D}M_{2,D}^{-1}G_{2,D}^T$ is the dual Schur complement of the matrix on the LHS of the matrix system.

Solving for $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\boldsymbol{u}_{1,D}, \boldsymbol{u}_{2,D}; \dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2)$ in this equation, we substitute this into the matrix system above to obtain the coupled system of ODEs

$$\begin{bmatrix} M_{1,D} & 0 \\ 0 & M_{2,D} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{u}}_{1,D} \\ \dot{\boldsymbol{u}}_{2,D} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{1,D} - F_{1,D}\boldsymbol{u}_{1,D} - G_{1,D}^T\boldsymbol{\lambda}(\boldsymbol{u}_{1,D}, \boldsymbol{u}_{2,D}; \dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2) - Q_{1,\Gamma}(\dot{\boldsymbol{g}}_1, \boldsymbol{g}_1) \\ \boldsymbol{f}_{2,D} - F_{2,D}\boldsymbol{u}_{2,D} + G_{2,D}^T\boldsymbol{\lambda}(\boldsymbol{u}_{1,D}, \boldsymbol{u}_{2,D}; \dot{\boldsymbol{g}}_1, \dot{\boldsymbol{g}}_2) - Q_{2,\Gamma}(\dot{\boldsymbol{g}}_2, \boldsymbol{g}_2) \end{bmatrix}.$$

We note that if we use an explicit scheme to approximate the time derivative, then the system effectively decouples and we may solve the problems independently.

3.2. Dynamic Flux Surrogate Model. We note that by solving for $\boldsymbol{\lambda}$ at each step in the IVR scheme, this vector may be used to solve for the states at the next time step. However, depending on the dimension of the system, solving for $\boldsymbol{\lambda}$ exactly may be computationally expensive. Instead, one can opt to use a simplified method of obtaining the next flux values via DMD. This rest of this section is based on the model of [1].

Suppose $\{\boldsymbol{u}_{1,D}^t, \boldsymbol{u}_{2,D}^t, \boldsymbol{\lambda}^t\}_{t=0}^{T_n}$ are given. Then let

$$\boldsymbol{y}^{i-1} = \begin{bmatrix} \boldsymbol{\lambda}^{i-1} \\ \boldsymbol{u}_{1,D}^i \\ \boldsymbol{u}_{2,D}^i \end{bmatrix}.$$

The next set of states $\boldsymbol{y}^i$ is obtained through propogation under a DMD operator A_F:

$$\boldsymbol{y}^i = A_F\boldsymbol{y}^{i-1}.$$

Unlike traditional DMD, the state vector is staggered so that the current flux value uses information of the states in the entire domain as well as the previous flux values.

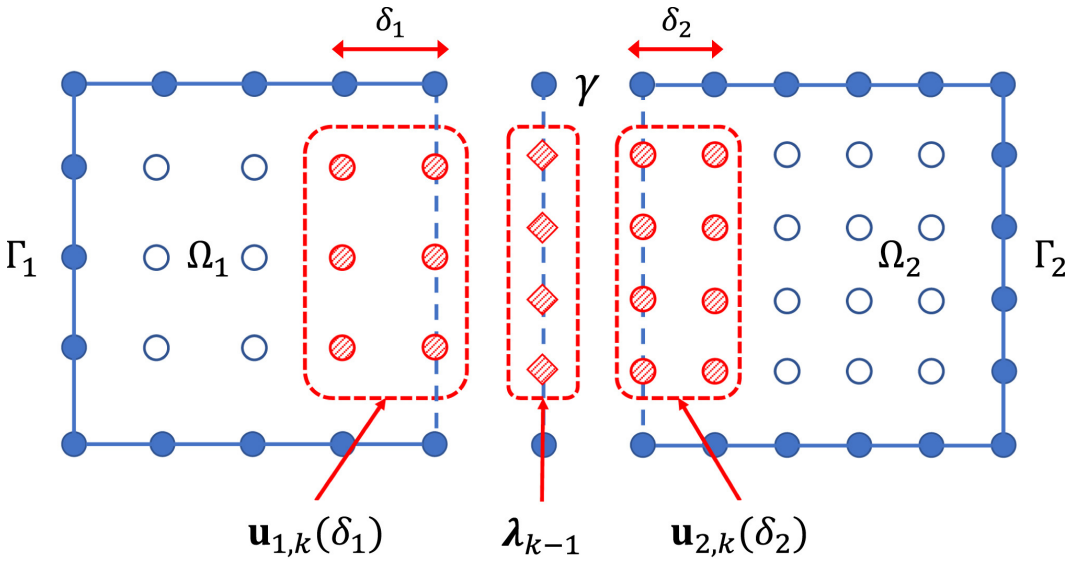


Fig. 3.1: The DMD-FS framework figure, obtained from [1], demonstrates the use of staggered states $\boldsymbol{u}_{j,\delta_j} = \boldsymbol{u}_{j,k}(\delta_j) \subset \boldsymbol{u}_{j,k} = \boldsymbol{u}_{j,D}$, $j = 1, 2$, which include interface patches of the subdomain solution coefficients. These states at the current time t_i are used together along with the coefficients of the interface flux $\boldsymbol{\lambda}^{i-1}$ at the previous time t_{i-1} to form the DMD-FS operator.

However, using the entire state domain as part of $\boldsymbol{y}$ may still be computationally expensive, so instead one can opt to use *patches* around γ. Let $\delta_i \in \mathbb{N}$ denote the number of patches used around the interface γ for subdomains $i = 1, 2$ and define

$$\boldsymbol{y}^{i-1} = \begin{bmatrix} \boldsymbol{\lambda}^{i-1} \\ \boldsymbol{u}^i_{1,\delta_1} \\ \boldsymbol{u}^i_{2,\delta_2} \end{bmatrix},$$

where $\boldsymbol{u}^i_{j,\delta_j} \subset \boldsymbol{u}^i_{j,D}$ for $j = 1, 2$, $\boldsymbol{u}^i_{j,D}$ represents the solution coefficients in subdomain j at time t_i, and $\boldsymbol{\lambda}^{i-1}$ are the coefficients of the flux vector at time t_{i-1}. Then the flux-surrogate model proposed by [1] is given by

$$\boldsymbol{y}^i = A_{FS}\boldsymbol{y}^{i-1}.$$

This framework develops a DMD operator based on a staggered approach where the solution states at the current time in each subdomain are concatenated together with the states of the flux at the previous time to form the state vectors used in DMD. See Figure 3.1 for a pictorial representation of this idea.

Denote $X_{FS} = \begin{bmatrix} \boldsymbol{y}^0 & \boldsymbol{y}^1 & \dots & \boldsymbol{y}^{t_{FS}-1} \end{bmatrix}$ and $Y_{FS} \begin{bmatrix} \boldsymbol{y}^1 & \boldsymbol{y}^2 & \dots & \boldsymbol{y}^{t_{FS}} \end{bmatrix}$. Then A_{FS} solves

$$Y_{FS} = A_{FS}X_{FS}.$$

Setting $X = U_{FS}\Sigma_{FS}V'_{FS}$ where U_{FS}, V_{FS} are orthogonal matrices and $\Sigma_{FS} = diag(\sigma_1, \sigma_2, \dots, \sigma_n)$ with $\sigma_i \geq 0$ for all $i \geq 1$, then A_{FS} is given by

$$A_{FS} = YV_{FS}\Sigma^{-1}_{FS}U'_{FS}.$$

3.3. Fully Data-Driven DMD Control-Flux-Input Model. Consider a scenario where λ_is are not known exactly. We consider the DMD Flux surrogate in lieu of the given flux data. Then using the previous sections, we define the coupled $DMDc - DMDFS$ model:

$$\begin{cases} \boldsymbol{u}_1^{i+1} = A_1 \boldsymbol{u}_1^i + B_1 \boldsymbol{\lambda}^i \\ \boldsymbol{u}_2^{i+1} = A_2 \boldsymbol{u}_2^i + B_2 \boldsymbol{\lambda}^i \\ \boldsymbol{\lambda}^i = \left[\tilde{A} \begin{bmatrix} \boldsymbol{\lambda}^{i-1} \\ \boldsymbol{u}_1^i \\ \boldsymbol{u}_2^i \end{bmatrix} \right] \Big|_{1:n_\gamma^F}, \end{cases}$$

where A_i, B_i come from the DMDc model and $\tilde{A}$ is a DMD operator.

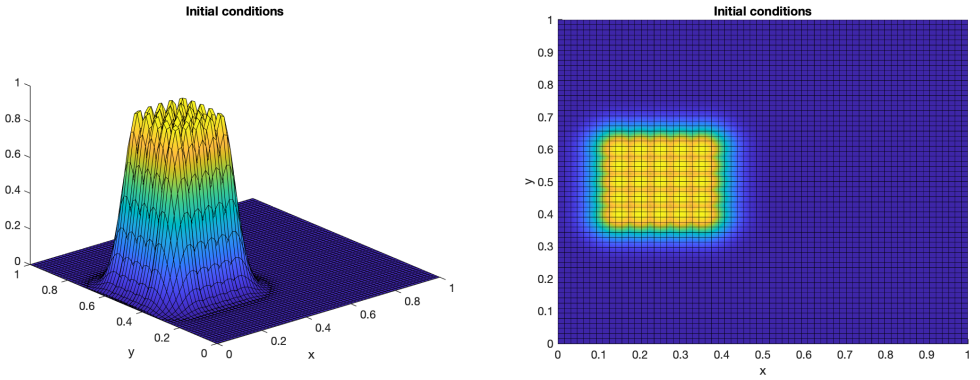
4. Results. In this section, we discuss the implementation of the models.

4.1. Brief Description of Training Data. The data used to train the models arises from numerically solving the discrete equations. Here, we discretize the governing equations using a finite element method over a 64×64 mesh of cells on the domain $\Omega = [0,1]^2$. This yields 65 nodes in each subdomain. We consider an interface $\gamma = \{(x,y) \in [0,1]^2 : x = 0.5$ and the boundary condition on Γ is Dirichlet, i.e. $u|_\Gamma = 0$ with a zero source term $f = 0$. The aforementioned discretization yields 63 interior degrees of freedom. The diffusion coefficients are set to be $\kappa_i = 1$ in both subdomains. We assume that the cells are coincident on γ and the aforementioned discretization yields 63 interior degrees of freedom on γ, i.e. $n_\gamma^F = 63$. Discretizing with finite elements yields a system of ODEs which are then decoupled between the subdomains using the IVR scheme of [7]. Afterwards, the ODEs are solved using an explicit scheme – which we choose to be forward Euler with an appropriate time step that satisfies the CFL condition. The finite element coefficients at each node $\{\boldsymbol{u}^i\}$ are obtained from the time integration while the fluxes $\{\boldsymbol{\lambda}^i\}$ are obtained through the IVR scheme via inversion of the dual Schur complement at each time step.

In the context of μ-PDEs, we consider a set of initial conditions which can potentially form a basis for initial conditions which are unseen and may be closely approximated as a linear combination of the trained initial conditions. In view of this, we consider Gaussians with centers contained in the square $[0.15, 0.35] \times [0.4, 0.6]$ and which are uniformly spaced throughout. Note that these initial conditions are fully contained in the left subdomain Ω_1. In particular, we consider 25 Gaussians with standard deviations $\sigma = 0.05$, see Figure 4.1. The choice of σ comes from considerations of the underlying mesh. For a full justification of using Gaussian initial conditions with specified means and standard deviations, see [1]. Since we are interested in seeing what happens when each of the Gaussians make a full revolution in Ω, then we set the time domain to be $[0, 2\pi]$. Based on considerations of the CFL condition, we set a uniform time step $\Delta t = \frac{2\pi}{1866} \approx 0.00336719469$, i.e. $n_T = 1866$.

Each of the initial conditions are then substituted in the governing equations and the resulting problem is solved as detailed in the previous paragraph. Afterwards, this yields a set of trajectory data for the solution coefficients (which we call states) and the fluxes. For the rest of this section, we will consider data obtained through the the FOM as the "ground truth" and they will serve as a benchmark to measure against for our experiments. Note that we may obtain this data for different initial conditions, such as that of the cone and the slotted cylinder, using this method – which will be important in performing the error analysis with respect to our numerical solvers.

4.2. Reproducibility. In this section, we demonstrate the reproducibilty results of the DMD subdomain solvers in two cases: (i) $\{\boldsymbol{\lambda}^i\}$ are known (ii) $\{\boldsymbol{\lambda}^i\}$ are unknown. In



(a) Set of Initial Conditions (b) Top-Down View

Fig. 4.1: The 25 initial conditions used for training the DMD operator

understanding how much we may expect the data-driven model to accurately simulate the dynamics of the given model and as a "proof of concept", we opted to train the DMD models over the 25 initial conditions with snapshot energy set at relatively high values. In future work, we hope to evaluate the accuracy of the model at lower energies. We speculate that the "right" value of the snapshot energy which allows the data-driven scheme to accurately approximate the dynamics is correlated with the associated Peclet number of (2.1). As a precaution, we would like to mention that this is still preliminary work and that there may be potential bugs in the coding implementation of the models. This may affect the quantities that are reported in this manuscript, for example the error values found in the following tables. Nevertheless, we believe it important to be transparent in reporting these values and emphasize that these errors should be taken as "suggestive" at best of the quality of the models. That being said, future work will be performed in the implementation to provide fully accurate results of these models.

Using MATLAB, we train two DMDc solvers over each of the subdomains Ω_0 and Ω_1. We consider the domain to be divided into 65 by 65 grid, with Ω_0 containing the left half (33 by 65) and Ω_1 being the right half (33 by 65). They are coincident on the boundary, where there are $n_\gamma^F = 63$ λ values.

The training data contains the state and flux values and contains 1866 time points (from 0 to 2π). The number of modes r for each is calculated using the snapshot energy criterion:

$$\frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{s} \sigma_i^2} > \epsilon$$

where $\epsilon \in (0, 1)$ is a prescribed tolerance and r is the least number of modes needed for this inequality to hold. Consider the above set of initial conditions indicated in Figure 4.1. The dataset contains 25 Gaussians with centers uniformly spaced in the square $[0.15, 0.35] \times [0.4, 0.5]$ with standard deviations $\sigma = 0.05$. Moreover, it contains time-series data for each of these Gaussians which evolve according to the PDE model problem. The training of the DMDc operator is performed with this dataset and with all trajectories simultaneously. For our results with the DMDc subdomain solvers, we set $\epsilon = 0.999999999$. This yields the number of retained modes to be $r_0 = 162$, $r_1 = 109$.

Initial Condition	Domain			
Gaussian Center	Ω_1		Ω_2	
	Number of Modes	L^∞-Error	Number of Modes	L^∞-Error
$(0.15, 0.4)$	162	0.0002892805964	109	0.0002320102399
$(0.35, 0.6)$	162	0.0000797235010	109	0.0044511526012

Table 4.1: Table of L^∞ Errors when the flux values are known exactly

Initial Condition	Domain				Interface Flux
Gaussian Center	Ω_1		Ω_2		γ
	Number of Modes	L^∞-Error	Number of Modes	L^∞-Error	L^∞-Error
$(0.15, 0.4)$	162	0.2362	109	0.2615	0.0985
$(0.35, 0.6)$	162	0.1931	109	0.2615	0.0411

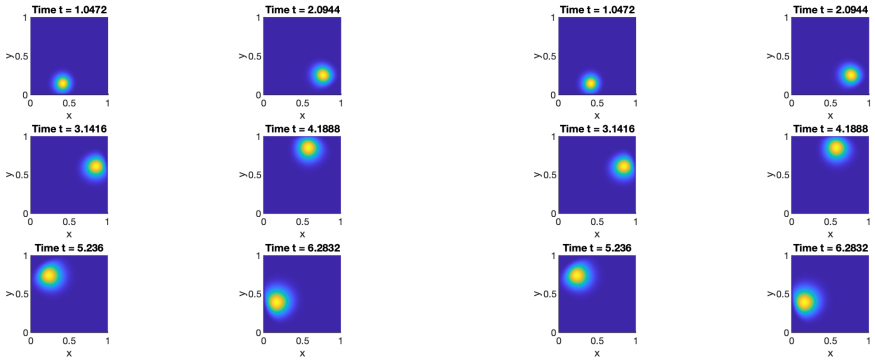
Table 4.2: Table of L^∞ Errors when the flux values are not known

For the fully data-driven numerical solver, we train the DMD-FS to approximate the flux using 9 uniformly spaced Gaussians with $\sigma = 0.05$. In particular, these Gaussians are centered at $(x, 0.5)$ with $x \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45\}$. Moreover, we use data from 5 patches around the interface and set the snapshot criterion at $\epsilon = 0.9999999999999$. This yields a flux model which captures the value of the flux at the order of $1e-6$.

After training the DMD model, we then reproduce several of the initial conditions and the flux values in the training data $\{\boldsymbol{\lambda}^i\}$ via propagation by the approximate operator $\tilde{A}$. With the fully data-driven model, we also propagate the flux values which are then fed into the subdomain solvers as inputs to produce the next states. In particular, we compare the DMD reproduction of the initial data with centers $(0.15, 0.4)$ and $(0.35, 0.6)$ assuming (i) the known values of $\{\boldsymbol{\lambda}^i\}_{i=1}^{n_\gamma^F}$ and (ii) the flux values are not known. To characterize the error, we assume the data given from the full-order model (FOM), i.e. the data obtained from running a finite element solver, is the "ground truth" and measure it using the L^∞ norm. In this manuscript, we opt to measure the absolute errors between the ground truth data and the data produced by the model. In future work, we intend to report the relative errors. The list of L^∞ errors when the fluxes are known is summarized in Table 4.1 while the one when the fluxes are unknown is given in Table 4.2.

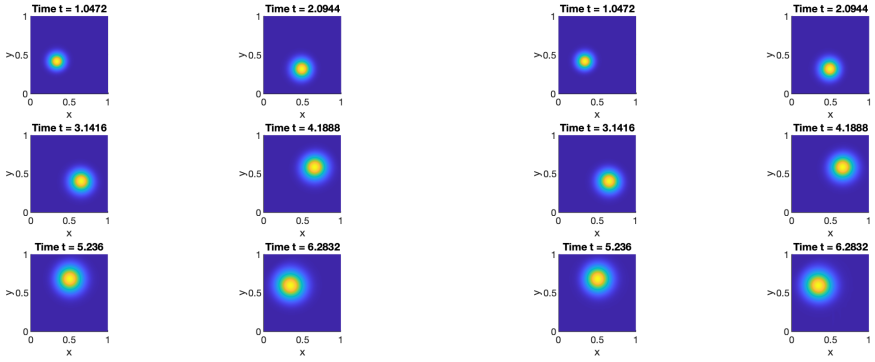
Figures 4.2, 4.3, 4.4 and 4.7 gives 6 different time points of the predicted numerical solutions in comparison to the FOM on $[0, 2\pi]$ in the cases of known and unknown fluxes. We consider additional diagnostics to analyze the behavior of the $DMDc-DMDFS$ model for the initial condition $(0.15, 0.4)$ which are given in Figures 4.5 and 4.6. Since there is possibly an error in the approximated fluxes, we also provide error plots for both the state and flux values. Figure 4.5 considers the set of trajectories for each state and flux errors as a function of time and Figure 4.6 uses a "top-down" perspective of the errors as a function of time. In the latter plot, it is assumed that the flux values are contained in the last $n_\gamma^F = 63$ values.

4.3. Predictability. Using the both the trained DMDc model with known flux values and the $DMDc-DMDFS$ for unknown fluxes developed in Section 4.2, we then test the results of the model on several examples which are unseen: a cone and a slotted cylinder. One way to justify the use of our models to solve these problems is to recall the parametric



(a) Ground Truth via FOM (b) Reproduction via DMDc

Fig. 4.2: Results with Initial Gaussian Centered at $(0.15, 0.4)$ when fluxes are known



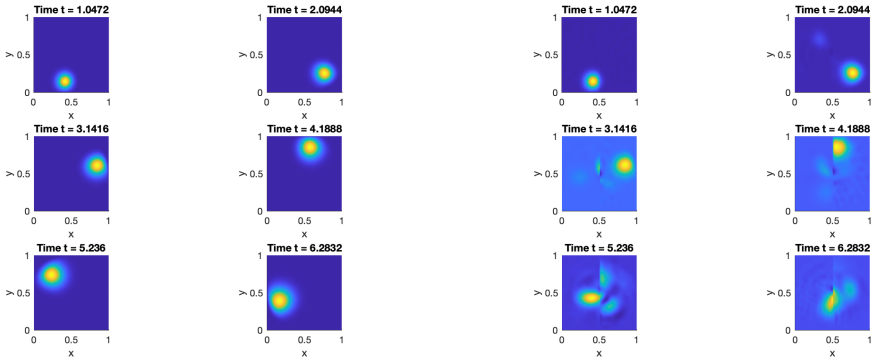
(a) Ground Truth via FOM (b) Reproduction via DMDc

Fig. 4.3: Results with Initial Gaussian Centered at $(0.35, 0.6)$ when fluxes are known

setting. In the framework of μPDEs, one can think of each of these initial conditions as some sort of linear combination of the 25 initial conditions used to train the DMDc models. Thus, due to this assumption and the linearity of the models, these initial conditions can be propagated starting from their initial states.

However in order for this assumption to be valid, these initial conditions are assumed to be contained within the domain spanned by the initial conditions, which is again $[0.15, 0.35] \times [0.4, 0.6]$. Thus, we consider a cone and a slotted cylinder centered at $(0.25, 0.5)$ with radius $r = 0.1$, see Figures 4.8 and 4.10. Once again, we use L^∞ errors in space-time and also consider the state and flux errors, which are summarized for each model in Tables 4.3 and 4.4.

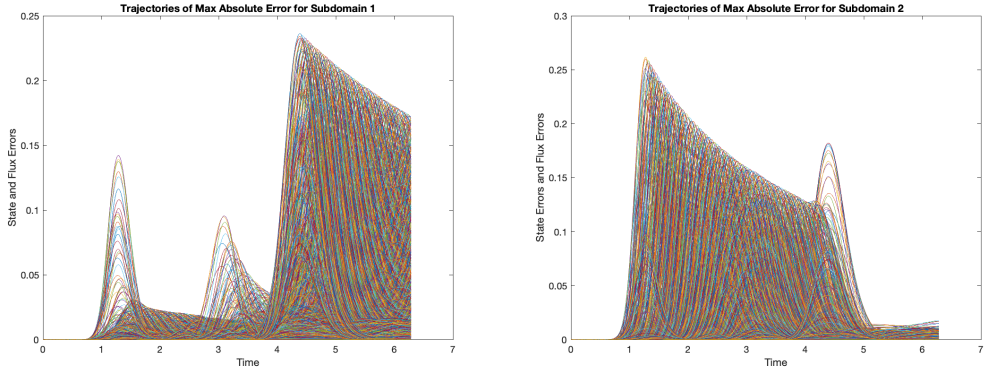
4.4. Analysis and Discussion. In this section, we will discuss our results in the cases when (i) the fluxes are known and (ii) when the fluxes are unknown. These each correspond



(a) Ground Truth via FOM

(b) Prediction via DMDc-DMDFS

Fig. 4.4: Results with Initial Gaussian Centered at $(0.15, 0.4)$ when fluxes are unknown



(a) Subdomain 1 Trajectory Errors

(b) Subdomain 2 Trajectory errors

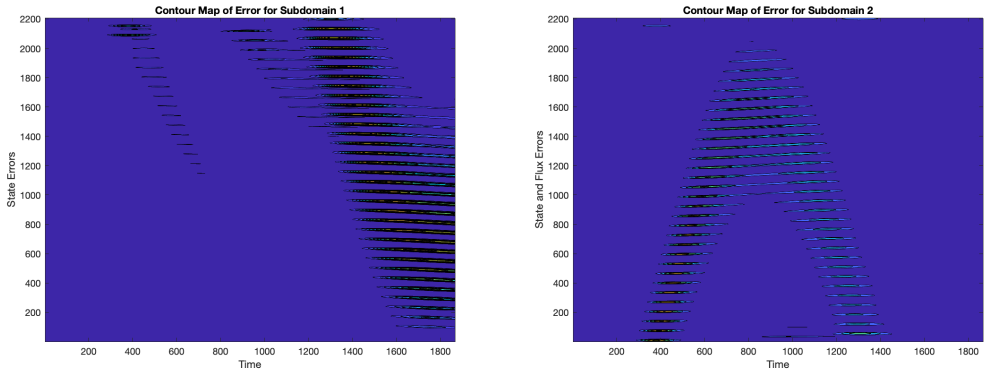
Fig. 4.5: Trajectory Errors for Initial Gaussian Centered at $(0.15, 0.4)$ when fluxes are unknown

Initial Condition	Ω_1 L^∞-Error	Ω_2 L^∞-Error
Cone	0.1172318743106	0.0002662479286
Slotted Cylinder	0.5469193774605	0.0088659661582

Table 4.3: L^∞ Errors when the fluxes are known

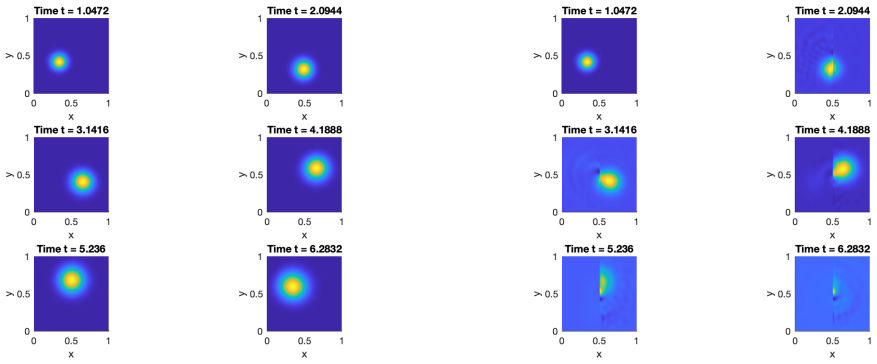
Initial Condition	Ω_1 L^∞-Error	Flux Error	Ω_2 L^∞-Error
Cone	0.2278	0.0812	0.2429
Cylinder	0.5469	0.0576	2.422

Table 4.4: L^∞ Errors when the fluxes are unknown



(a) Subdomain 1 Contour Plot Errors (b) Subdomain 2 Trajectory errors

Fig. 4.6: Contour Plot Errors for Initial Gaussian Centered at $(0.15, 0.4)$ when fluxes are unknown



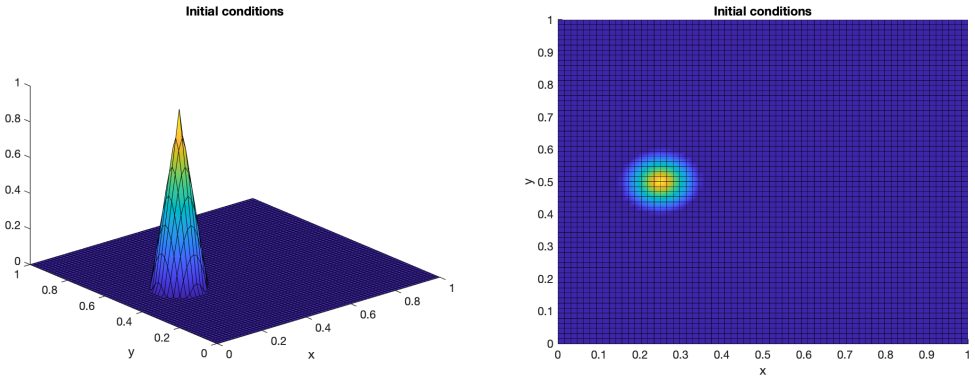
(a) Ground Truth via FOM (b) Prediction via DMD

Fig. 4.7: Results with Initial Gaussian Centered at $(0.35, 0.6)$ when fluxes are unknown

to the DMDc and DMDc-DMDFS solvers.

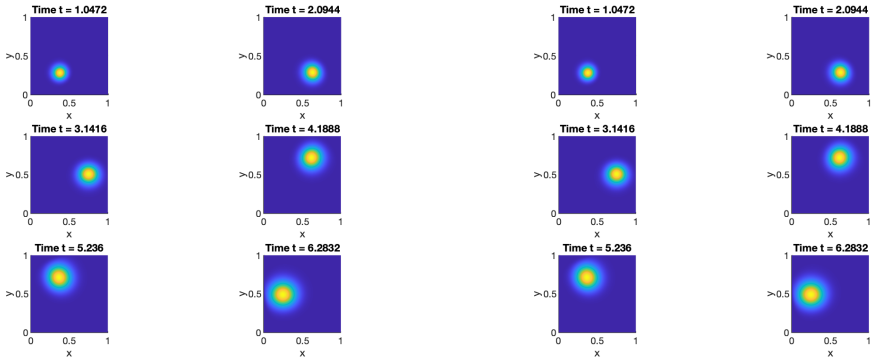
In the known fluxes regime, the trained model from 25 initial Gaussians reproduces the dynamics of the data within 10^{-3} accuracy, as indicated in Table 4.1. These results seem to hold with both subdomain solvers. Qualitatively, Figures 4.2 and 4.3 show that the reproduction given by the DMDc subdomain solvers do not differ markedly from the ground truth as produced by the full-order model. Next, prediction over unseen data with the cone and slotted cylinder initial conditions yield better errors in the Ω_2 relative to Ω_1, although the errors suffer more relative to the reproduction case. Nevertheless, the qualitative pictures as shown in Figures 4.9 and 4.11 demonstrates that the dynamics are still well-preserved.

When the fluxes are unknown and must be numerically approximated from the flux



(a) Cone Initial Condition (b) Top-Down View

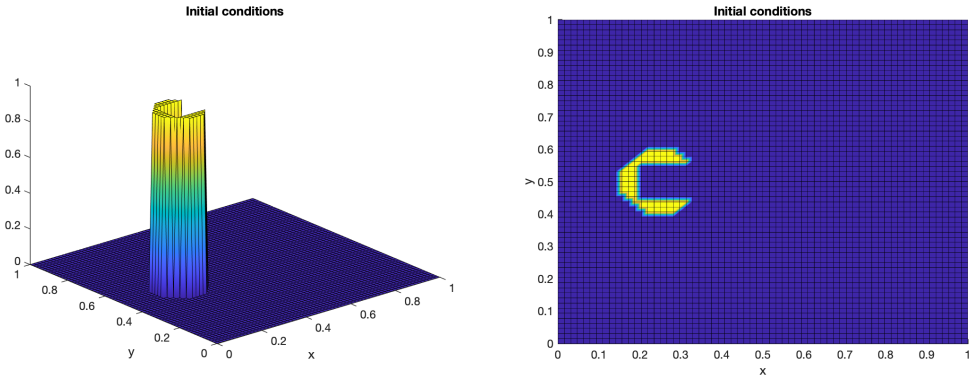
Fig. 4.8: The cone initial condition used for predicting with the DMD operator



(a) Ground Truth via FOM (b) Prediction via DMDc

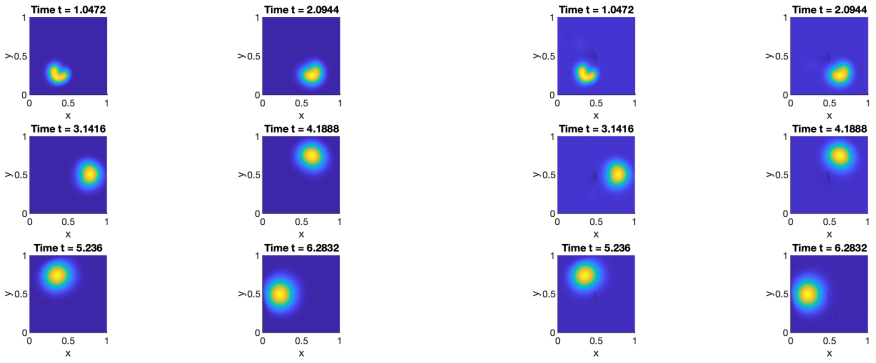
Fig. 4.9: Results with Cone Centered at $(0.25, 0.5)$ when fluxes are known

surrogate model, we then then use these approximations to advance the subdomain solvers. In terms of reproduction of the initial data, Table 4.2 shows that there is an increase in error relative to the known case. Here, we notice that the interface flux errors are within the order of 10^-2, however the state errors in Ω_1 and Ω_2 are only at the order of 10^{-1}. These errors affect the qualitative behavior, where we see in Figures 4.4 and 4.7 that the dynamics develop numerical artifacts closer to the boundary. These artifacts persist over time and lead to a loss in accuracy. To further analyze where these errors occur, we provided trajectory error plots for both Ω_1 and Ω_2 as well as a top-down view via contour plots. Figure 4.5 shows that the errors in Ω_1 remain small and bounded up until the Gaussian crosses the boundary. Upon reaching the boundary, the subdomain state errors increase significantly and slowly decay. On the other hand, the errors in Ω_2 remain larger at the beginning and slowly decay over time. These are also corroborated from Figure 4.6 which shows the propagation of



(a) Cylinder Initial Condition (b) Top-Down View

Fig. 4.10: The cylinder initial condition used for predicting with the DMD operator

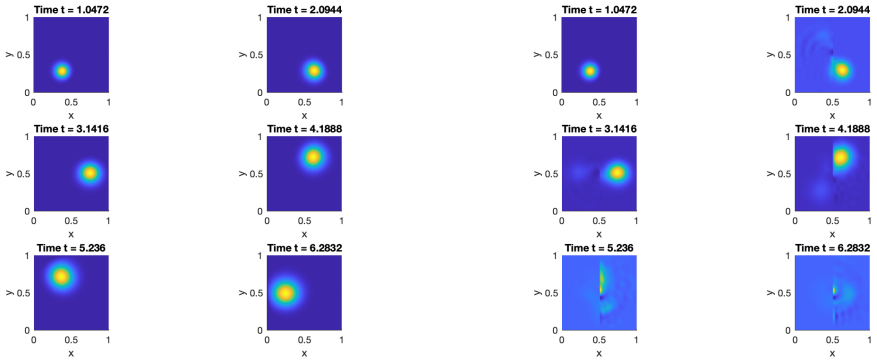


(a) Ground Truth via FOM (b) Prediction via DMDc

Fig. 4.11: Results with Slotted Cylinder Centered at $(0.25, 0.5)$ when fluxes are known

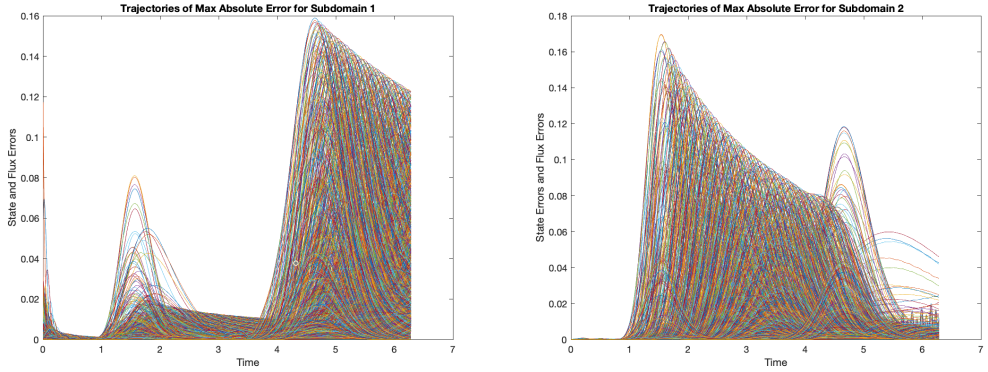
errors in time over Ω_1 and the evolution of errors in Ω_2.

Next, in terms of prediction with unseen data, Table 4.4 yield similar-to-worse accuracy compared to the reproductive case and to the analogous known flux regime. Here, both the errors in Ω_1 are at the order of 10^{-1} while the error in Ω_2 is much higher in the case of the slotted cylinder. Figures 4.12 and 4.15 demonstrate a clear loss in accuracy compared to the data produced by the full-order model, thereby showing the model fails to accurately capture the dynamics. We provide a similar analysis as in the reproductive case by showcasing the trajectory error plots and contour plots. In the case of the cone, Figures 4.5 and 4.6 are similar to the reproductive case. However, the trajectory error plot for the slotted cylinder in Figure 4.16 shows that while the errors in Ω_1 remain relatively stable, the errors in Ω_2 seem to exponentially increase by later times. The corresponding contour plot of the cylinder errors in Figure 4.17 shows that these errors relatively larger closer to the boundary.



(a) Ground Truth via FOM (b) Prediction via DMDc-DMDFS

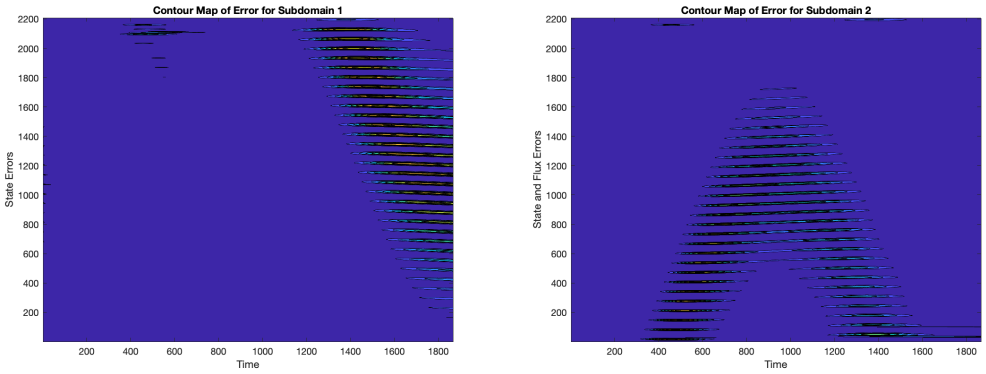
Fig. 4.12: Results with Cone when fluxes are unknown



(a) Subdomain 1 Trajectory Errors (b) Subdomain 2 Trajectory errors

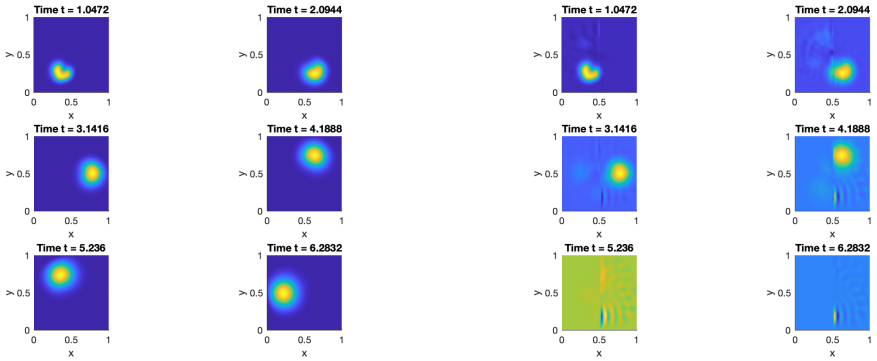
Fig. 4.13: Trajectory Errors for Cone when fluxes are unknown

In summary, when the Lagrange multipliers are known and the DMDc models are used to approximate the subdomain states, the model can be trained well enough to obtain accurate representations. However, when the fully data-driven model is implemented with the flux values numerically approximated via the DMDFS, we find that there is a sharp increase in errors that lead to massive departures from the original dynamics – even when the initial DMDFS solver was trained to an order of 10^{-6} in error. This suggests two things: (1) the implementation of the model was incorrect or (2) the inherent dynamics of the flux and subdomain states is unstable. While we do not report it in detail here, we tested (2) out by perturbing the original flux data by adding a vector with values randomly sampled from $[-10^{-6}, 10^{-6}]$ and evolved the initial conditions with these perturbed flux values. We found that this did not yield significantly different errors from the ones produced and reported in this manuscript. This suggests that the implementation may be incorrect and will require



(a) Subdomain 1 Contour Plot Errors (b) Subdomain 2 Trajectory errors

Fig. 4.14: Contour Plot Errors for Cone when fluxes are unknown

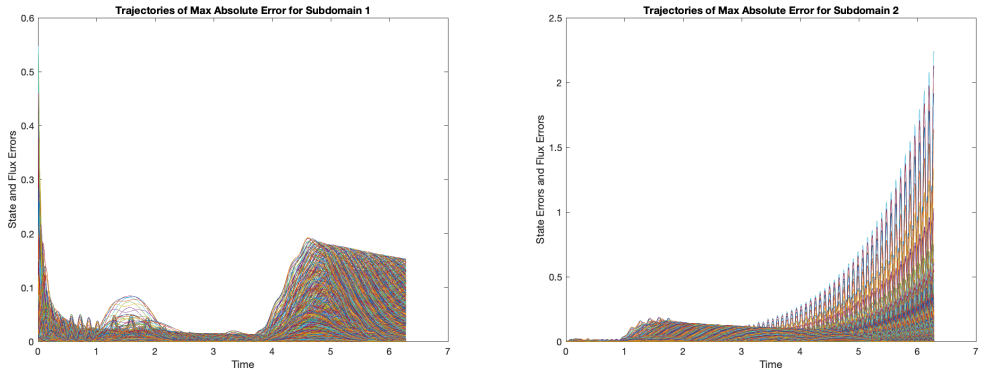


(a) Ground Truth via FOM (b) Prediction via DMDc-DMDFS

Fig. 4.15: Results with Slotted Cylinder when fluxes are unknown

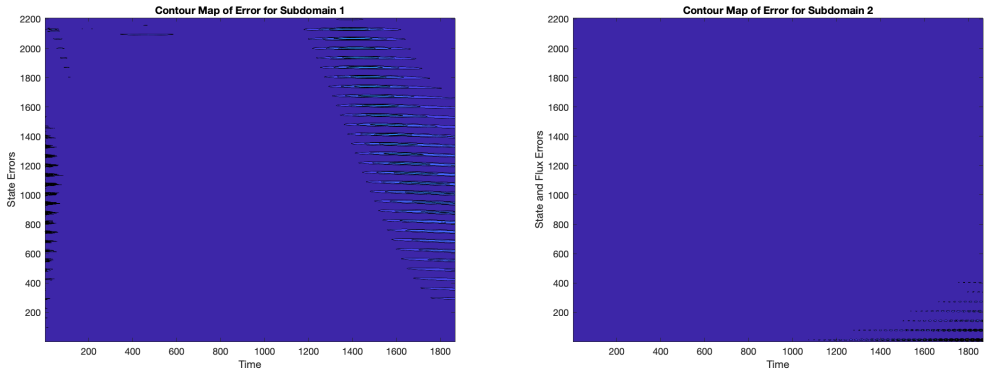
further debugging. Once this is done, we expect that the errors from the DMDc-DMDFS models will be better.

5. Conclusion. We introduced a new model to numerically simulate the dynamics of a coupled problem, which have potential applications to parametric problems. Our model is fully data-driven and based on the dynamic mode decomposition and its control variant. Our simulations in two cases where either the fluxes are known or unknown, we find that the model performs better when the fluxes are known exactly and accurately capture the underlying dynamics. In contrast, the fully data-driven model struggled to give similar results, even with an accurate flux surrogate. Our post-investigation revealed that there may be a programming issue and requires further debugging. Once debugged, we hope to redo the simulations and expect to obtain better numerical results.



(a) Subdomain 1 Trajectory Errors (b) Subdomain 2 Trajectory errors

Fig. 4.16: Trajectory Errors for Slotted Cylinder when fluxes are unknown



(a) Subdomain 1 Contour Plot Errors (b) Subdomain 2 Trajectory errors

Fig. 4.17: Contour Plot Errors for Cylinder when fluxes are unknown

REFERENCES

[1] P. Bochev, J. Owen, P. Kuberry, and J. Connors, *Dynamic flux surrogate-based partitioned methods for interface problems*, Computer Methods in Applied Mechanics and Engineering, 429 (2024), p. 117115.

[2] J. M. Connors and K. C. Sockwell, *A multirate discontinuous-galerkin-in-time framework for interface-coupled problems*, SIAM Journal on Numerical Analysis, 60 (2022), pp. 2373–2404.

[3] A. de Castro, P. Bochev, P. Kuberry, and I. Tezaur, *Explicit synchronous partitioned scheme for coupled reduced order models based on composite reduced bases*, Computer Methods in Applied Mechanics and Engineering, 417 (2023), p. 116398.

[4] C. A. Felippa, K.-C. Park, and C. Farhat, *Partitioned analysis of coupled mechanical systems*, Computer methods in applied mechanics and engineering, 190 (2001), pp. 3247–3270.

[5] J. Hanson, P. Bochev, and B. Paskaleva, *Learning compact physics-aware delayed photocurrent models using dynamic mode decomposition*, Statistical Analysis and Data Mining: The ASA Data Science Journal, 14 (2021), pp. 521–535.

[6] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*, SIAM, 2016.
[7] K. Peterson, P. Bochev, and P. Kuberry, *Explicit synchronous partitioned algorithms for interface problems based on lagrange multipliers*, Computers & Mathematics with Applications, 78 (2019), pp. 459–482.
[8] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Dynamic mode decomposition with control*, SIAM Journal on Applied Dynamical Systems, 15 (2016), pp. 142–161.
[9] K. C. Sockwell, K. Peterson, P. Kuberry, P. Bochev, and N. Trask, *Interface flux recovery coupling method for the ocean–atmosphere system*, Results in Applied Mathematics, 8 (2020), p. 100110.
[10] P. A. Ullrich and M. A. Taylor, *Arbitrary-order conservative and consistent remapping and a theory of linear maps: Part i*, Monthly Weather Review, 143 (2015), pp. 2419 – 2440.

AN INEXACT WEIGHTED PROXIMAL TRUST-REGION METHOD

LEANDRO FARIAS MAIA*, ROBERT BARALDI†, AND DREW P. KOURI‡

Abstract. In [R. J. Baraldi and D. P. Kouri, *Math. Program.*, 201:1 (2023), pp. 559-598], the authors introduced a trust-region method for minimizing the sum of a smooth nonconvex and a nonsmooth convex function, the latter of which has an analytical proximity operator. While many functions satisfy this criterion, e.g., the ℓ^1-norm defined on ℓ^2, many others are precluded by either the topology or the nature of the nonsmooth term. Using the δ-Fréchet subdifferential, we extend the definition of the inexact proximity operator and enable its use within the aforementioned trust-region algorithm. Moreover, we augment the analysis for the standard trust-region convergence theory to handle proximity operator inexactness with weighted inner products. We first introduce an algorithm to generate a point in the inexact proximity operator and then apply the algorithm within the trust-region method to solve an optimal control problem constrained by Burgers' equation.

1. Introduction. Our goal is the efficient numerical solution to the nonsmooth, nonconvex optimization problem

$$\min_{x\in X} \ \{f(x)+\phi(x)\}, \tag{1.1}$$

where X is a real Hilbert space, $f : X \to \mathbb{R}$ is a smooth nonconvex function and $\phi : X \to (-\infty, +\infty]$ is a proper, closed and convex function. The method developed in [1] solves (1.1) via a trust-region method that permits inexact evaluations of the smooth objective function f and its gradient. The convergence analysis for the trust-region method is contingent on the proximity operator of ϕ being exact, i.e. analytically computable. In this paper, we extend the inexact proximal trust-region algorithm in [1] to leverage certain inexact evaluations of the proximity operator of ϕ. The proximity operator of ϕ at x with stepsize r is defined as the unique solution to the minimization problem

$$\mathrm{Prox}_{r\phi}(x) \coloneqq \operatorname*{arg\,min}_{y\in X} \left\{ \frac{1}{2r}\|y-x\|^2 + \phi(y) \right\} \tag{1.2}$$

and is a powerful tool to deal with composite nonsmooth optimization problems like (1.1). Algorithms that leverage the proximity operator, like ISTA and FISTA [5], are typically first-order and only apply to convex optimization problems. Like [1], these methods require that the proximity operator be computed exactly—a requirement that is often computationally expensive or even impossible to satisfy. For instance, if X is finite dimensional with dot product defined by a diagonal matrix D and ϕ is the ℓ^1 norm, then the proximity operator has an analytical expression. In contrast, if the dot product is defined by a non-diagonal symmetric positive definite (spd) matrix M, then evaluating the proximity operator requires a specialized iterative method to obtain an approximate solution. Although preliminary studies on optimization with inexact proximity operators exist [8, 9, 10, 11, 14, 15], these works only apply to convex problems and often exhibit slow convergence rates. Using these methods as motivation, we develop a proximal Newton method—tailored to the case of diagonal and non-diagonal dot products—that can handle inexact proximity operators. We define our inexact proximity operator as

$$\begin{aligned}\mathrm{Prox}_{r\phi}(x,\delta) \coloneqq \Big\{ p\in X \;\Big|\; & \frac{1}{2r}\|p-x\|^2+\phi(p) \\ & \le \frac{1}{2r}\|z-x\|^2+\phi(z)+\delta\|z-p\| \ \ \forall\, z\in X \Big\}\end{aligned} \tag{1.3}$$

*Department of Industrial and Systems Engineering, Texas A&M University, leandro.maia@tamu.edu
†Optimization and Uncertainty Quantification, Sandia National Laboratories, rjbaral@sandia.gov
‡Optimization and Uncertainty Quantification, Sandia National Laboratories, dpkouri@sandia.gov

The inclusion of the term $\delta\|z-u\| \geq 0$ on the right-hand side of (1.3) allows the proximity operator to be computed inexactly.

The remainder of the article is structured as follows. In Section 3, we review the inexact trust-region method introduced in [1], presenting the main components of the algorithm. We also include a lemma that enables the use of our inexact proximal map. In Section 4, we lay out our definition of inexact proximity operator based on the Fréchet subdifferential. In Section 4, we present the main technical results of this work, where we establish the relationship between the δ-Fréchet subdifferential [13] and the δ-proximal map. We conclude with Section 5, where we present numerical results for the optimal control of Burgers' equation.

2. Notation and Problem Assumptions. Throughout, X denotes a real Hilbert space with inner product $\langle x, y\rangle$ and norm $\|x\| = \langle x, x\rangle^{1/2}$. To simplify the presentation, we associate the topological dual space X^* with X through Riesz representation. We assume that $\phi : X \to (-\infty, +\infty]$ is proper, closed and convex with effective domain

$$\operatorname{dom}\phi \coloneqq \{x \in X \,|\, \phi(x) < +\infty\}.$$

We further assume that $f : X \to \mathbb{R}$ is Fréchet differentiable on an open set containing $\operatorname{dom}\phi$ and its derivative is Lipschitz continuous on that set. We denote the derivative of f at $x \in X$ by $f'(x) \in X^*$ and its gradient (i.e., Riesz representer) by $\nabla f(x) \in X$. Finally, we assume that the total objective function $F(x) \coloneqq f(x) + \phi(x)$ is bounded from below.

The main focus of this paper is the use of inexact proximity operators arising from weighted inner products. To this end, we define $a : X \times X \to \mathbb{R}$ to be a symmetric, coercive and continuous bilinear form, i.e., there exists $0 < \alpha_1 \leq \alpha_2 < \infty$ such that

$$a(x,y) = a(y,x), \quad \alpha_1\|x\|^2 \leq a(x,x), \qquad \text{and} \quad a(x,y) \leq \alpha_2\|x\|\|y\| \qquad \forall\, x,\, y \in X.$$

We associate a with the invertible (cf. the Lax-Milgram lemma), self-adjoint, positive and continuous linear operator $A : X \to X$ defined by its action:

$$\langle Ax, y\rangle = a(x,y) \qquad \forall\, x,\, y \in X.$$

Recall that $a(\cdot,\cdot)$ defines an inner product on X and its associated norm $\|\cdot\|_a = \sqrt{a(\cdot,\cdot)}$ is equivalent to $\|\cdot\|$. We denote the (inexact) proximity operator associate with a by $\operatorname{Prox}^a_{r\phi}$.

For our numerical results, we will work with $X = \mathbb{R}^n$ endowed with the dot product

$$\langle x, y\rangle = \langle x, y\rangle_M \coloneqq x^\top My = \sum_{i=1}^n \sum_{j=1}^n m_{i,j}x_iy_j,$$

where $M \in \mathbb{R}^{n\times n}$ is a non-diagonal spd matrix. In many common choices of ϕ, the proximity operator associated with the M-weighted dot product lacks an analytical form. In this case, we replace $\langle\cdot,\cdot\rangle$ with the equivalent dot product

$$a(x,y) = \langle x, y\rangle_D \coloneqq x^\top Dy = \sum_{i=1}^n d_ix_iy_i,$$

where $D = \operatorname{diag}(d) \in \mathbb{R}^{n\times n}$ is a positive diagonal matrix. In this setting, the linear operator A is given by

$$A = M^{-1}D.$$

For clarity, we denote $\mathbb{R}^n$ endowed with the M-weighted dot product by X_M and analogously for X_D. We further denote the proximity operator defined on X_M by $\operatorname{Prox}^M_{r\phi}$ and similarly on X_D by $\operatorname{Prox}^D_{r\phi}$.

3. A Proximal Trust-Region Method. Trust-regions are iterative methods for computing approximate solutions to general nonconvex optimization problems [7] and are ideal for leveraging inexact computations [1, 12]. In the recent work [1], the authors developed a proximal trust-region method for nonsmooth optimization problems with the form (1.1) that exploits inexact function and gradient evaluations with guaranteed convergence. At the k-th iteration of [1, Algorithm 1], we compute a trial iterate x_k^+ that approximately solves the trust-region subproblem

$$\min_{x\in X}\{m_k(x) \coloneqq f_k(x) + \phi(x)\} \qquad \text{subject to} \qquad \|x - x_k\| \le \Delta_k, \tag{3.1}$$

where $x_k \in \operatorname{dom}\phi$ is the current iterate, f_k is a smooth local model of f around x_k, and $\Delta_k > 0$ is the trust-region radius. In particular, we require that the trial iterate x_k^+ satisfies the fraction of Cauchy decrease condition: there exist positive constants κ_{rad} and κ_{fcd}, independent of k, such that

$$\|x_k^+ - x_k\| \le \kappa_{\text{rad}}\Delta_k \tag{3.2a}$$

$$m_k(x_k) - m_k(x_k^+) \ge \kappa_{\text{fcd}} h_k \min\left\{\frac{h_k}{1+\omega_k}, \Delta_k\right\}, \tag{3.2b}$$

where ω_k is a measure of the curvature of f_k given by

$$\omega_k \coloneqq \sup\{|\omega(f_k, x_k, s)| \,|\, 0 < \|s\| \le \kappa_{\text{rad}}\Delta_k\},$$

$\omega(g, x, s)$ is the curvature of the Fréchet differentiable function $g : X \to \mathbb{R}$ at $x \in X$ in the direction $s \in X$, i.e.,

$$\omega(g, x, s) \coloneqq \frac{2}{\|s\|^2}[g(x+s) - g(x) - \langle \nabla g(x), s\rangle],$$

and h_k is our stationarity metric given by

$$h_k \coloneqq \frac{1}{r_0}\|\text{Prox}_{r_0\phi}(x_k - r_0\nabla f(x_k)) - x_k\|. \tag{3.3}$$

Given a trial iterate x_k^+ that satisfies (3.2), we decide whether to accept or reject x_k^+ using the ratio of computed (cred_k) and predicted (pred_k) reductions,

$$\rho_k \coloneqq \frac{\text{cred}_k}{\text{pred}_k}. \tag{3.4}$$

The computed reduction cred_k is an approximation of the actual reduction

$$\text{ared}_k \coloneqq F(x_k) - F(x_k^+) \tag{3.5}$$

and the predicted reduction pred_k is the decrease predicted by the model m_k,

$$\text{pred}_k \coloneqq m_k(x_k) - m_k(x_k^+). \tag{3.6}$$

If $\rho_k \ge \eta_1$, we accept x_k^+, i.e., $x_{k+1} = x_k^+$. Otherwise, we reject it, setting $x_{k+1} = x_k$. We furthermore use ρ_k to increase or decrease the trust-region radius Δ_k. Here, $\eta_1 \in (0, 1)$ is a user-specified parameter.

Algorithm 1 states the inexact proximal trust-region method from [1, Algorithm 1]. To ensure convergence, Algorithm 1 requires the following assumptions on the inexact

evaluations f and its gradient. The first assumption ensures that the computed reduction cred_k is a sufficiently accurate approximation of the actual reduction ared_k.

ASSUMPTION 3.1 (Inexact Objective). *There exists a positive constant κ_{obj}, independent of k, such that*

$$|\text{ared}_k - \text{cred}_k| \leq \kappa_{\text{obj}} \left[\eta \min\{\text{pred}_k, \theta_k\}\right]^{\zeta} \tag{3.7}$$

where

$$\zeta > 1, \quad 0 < \eta < \min\{\eta_1, (1 - \eta_2)\} \quad \textit{and} \quad \lim_{k \to +\infty} \theta_k = 0$$

are used provided parameters.

Note that all quantities on the right-hand side of (3.7) are available when computing cred_k, enabling one to avoid the computation of the actual reduction ared_k and hence the objective function F. Similarly, the following assumption enables approximations of the gradient of f within a prescribed tolerance that depends on the state of the algorithm.

ASSUMPTION 3.2 (Inexact Gradient). *The model $f_k : X \to \mathbb{R}$ has Lipschitz continuous derivatives on an open set containing* $\operatorname{dom} \phi$ *and there exists a positive constant κ_{grad}, independent of k, such that*

$$\|g_k - \nabla f(x_k)\| \leq \kappa_{\text{grad}} \min\{h_k, \Delta_k\},$$

where $g_k \coloneqq \nabla f_k(x_k)$.

Algorithm 1 Nonsmooth Trust-Region Algorithm

Require: $x_1 \in \operatorname{dom}\psi$, $\Delta_1 > 0$, $0 < \eta_1 < \eta_2 < 1$, and $0 < \gamma_1 \leq \gamma_2 \leq \gamma_3$

for $k = 1, 2, \ldots$ **do**
  Model Selection: Choose m_k that satisfies Assumption 3.2
  Step Computation: Compute $x_k^+ \in X$ satisfying equation (3.2)
  Computed Reduction: Compute cred_k satisfying Assumption 3.1
  Step Acceptance and Radius Update: Compute ρ_k as in (3.4)
  if $\rho_k < \eta_1$ **then**
    $x_{k+1} \leftarrow x_k$
    $\Delta_{k+1} \in [\gamma_1 \Delta_k, \gamma_2 \Delta_k]$
  else
    $x_{k+1} \leftarrow x_k^+$
    if $\rho_k \in [\eta_1, \eta_2)$ **then**
      $\Delta_{k+1} \in [\gamma_2 \Delta_k, \Delta_k]$
    else
      $\Delta_{k+1} \in [\Delta_k, \gamma_3 \Delta_k]$
    end if
  end if
end for

A critical component of the convergence theory for Algorithm 1 is that the trial iterate satisfies (3.2). In the smooth case (i.e., $\phi \equiv 0$), this condition is satisfied if x_k^+ produces at least a fraction of the decrease achieved by the steepest descent step. In the nonsmooth case, [1] instead defines sufficient decrease using the proximal gradient step. In particular, [1, Lemma 7] illustrates the relationship between the proximal gradient stepsize, r, and the required model decrease. Before stating this important lemma, we first define the proximal gradient path

$$p_k(r) \coloneqq \text{Prox}_{r\phi}(x_k - r g_k) - x_k \quad \text{and} \quad x_k(r) \coloneqq x_k + p_k(r). \tag{3.8}$$

We also define the quantity

$$Q_k(r) := \langle g_k, p_k(r)\rangle + (\phi(x_k(r)) - \phi(x_k)), \tag{3.9}$$

which is used to measure the model decrease of the iterate $x_k(r)$.

The main descent lemma derived in [1] plays the same role in our analysis, and we defer the proof of this lemma to maintain the focus of our exposition. We directly quote [1, Lemma 7] below.

LEMMA 3.1. *The function $r \mapsto Q_k(r)$ is continuous and nonincreasing. In fact, if $r > t > 0$ then $Q_k(t) > Q_k(r)$ whenever $x_k(t) \neq x_k(r)$. Moreover, if $h_k > 0$ then*

$$Q_k(r) = \langle g_k, p_k(r)\rangle + (\phi(x_k(r)) - \phi(x_k)) \leq -\frac{1}{r}\|p_k(r)\|^2 < 0$$

Lemma 3.1 is an inherent property of the proximal gradient map, which is used in [1] to verify the conditions defining the Cauchy point [1, Eq. (25)]. In the subsequent sections, we extend Algorithm 1 and Lemma 3.1 to handle the case of inexact proximity operators.

4. Inexact Proximity Operator. We first present a general definition of inexact proximity operators, which is motivated by the δ-Fréchet subgradient. We will later demonstrate how to compute a δ-proximity operator for the case of weighted inner products.

DEFINITION 4.1 (δ-Proximity Operator). *Let $x \in \mathrm{dom}\,\phi$, $r > 0$, and $\delta > 0$ be given. We define the δ-proximity operator at x as the following set-valued map*

$$\begin{aligned}\mathrm{Prox}_{r\phi}(x,\delta) := \Big\{ p \in X \;\Big|\; &\frac{1}{2r}\|p - x\|^2 + \phi(p) \\ &\leq \frac{1}{2r}\|z - x\|^2 + \phi(z) + \delta\|z - p\| \;\; \forall\, z \in X \Big\}.\end{aligned} \tag{4.1}$$

As claimed, the notion of δ-proximity operator introduced in Definition 4.1 is closely related the the δ-Fréchet subdifferential, which we define next.

DEFINITION 4.2 (δ-Subgradient). *Let $x \in \mathrm{dom}\,\phi$ and $\delta > 0$ be given. We say that $s \in X$ is a δ-Fréchet subgradient, or simply a δ-subgradient, of ϕ at x if*

$$\phi(y) \geq \phi(x) + \langle s, y - x\rangle - \delta\|y - x\|$$

for all $y \in X$. The set $\partial\phi_\delta(x)$ consists of all δ-subgradient of ϕ at x.

It is straightforward to observe that by setting $\delta = 0$ we recover the standard definitions of subgradient and proximity operator. By replacing the proximity operator with the inexact version defined in Definition 4.1, we redefine the proximal gradient path (3.8), and hence the Cauchy point, as

$$\tilde{p}_k(u,r) := u - x_k \quad \text{and} \quad \tilde{x}_k(u,r) := x_k + \tilde{p}_k(u,r), \tag{4.2}$$

where $u \in \mathrm{Prox}_{r\phi}(x_k - rg_k, \delta)$. Analogously, we update the quantity Q_k in (3.9) to

$$\tilde{Q}_k(u,r) := \langle g_k, \tilde{p}_k(u,r)\rangle + (\phi(\tilde{x}_k(u,r)) - \phi(x_k)) \tag{4.3}$$

We derive the close relationship between the δ-subdifferntial and the δ-proximity operator in Theorem 4.4, but we first require the following technical lemma.

LEMMA 4.3. *Let $g : X \to \mathbb{R}$ be a continuously Fréchet differentiable convex function,*

$$H(x) := g(x) + \phi(x),$$

and $\delta \geq 0$*, then*

$$\partial_\delta H(x) = \nabla g(x) + \partial_\delta \phi(x).$$

Moreover, x^* *is a* δ*-minimizer of* H*, i.e.,* $H(x^*) \leq H(x) + \delta\|x - x^*\|$ *for all* $x \in X$*, if and only if*

$$-\nabla g(x^*) \in \partial_\delta \phi(x^*).$$

Proof. This is a direct consequence of [6, Corollary 17.3]. □

We now state and prove the main result of this section.

THEOREM 4.4. *Let* $x \in X$ *and* $\delta \geq 0$*. Then,*

$$u \in \operatorname{Prox}_{r\phi}(x, \delta) \qquad \Longleftrightarrow \qquad x \in (I + r\partial_\delta \phi)(u).$$

Proof. Let $\varphi(\cdot) = \frac{1}{2r}\| \cdot - x\|^2 + \phi(\cdot)$, and recall Definition 4.1 implies $u \in \operatorname{Prox}_{r\phi}(x, \delta)$ satisfies

$$\varphi(u) \leq \varphi(z) + \delta\|z - u\| \quad \forall\, z \in X \tag{4.4}$$

i.e., u is a δ-minimizer of φ. Applying Lemma 4.3, expression (4.4) implies

$$-\nabla g(u) = r^{-1}(x - u) \in \partial_\delta \phi(u) \qquad \Longleftrightarrow \qquad x \in u + r\partial_\delta \phi(u) = (I + r\partial_\delta \phi)(u),$$

concluding the proof. □

The proximity operator is present in two components of Algorithm 1, through the stationarity metric h_k and in defining the Cauchy point using Q_k. To ensure convergence, we require that $\delta > 0$ and $u \in \operatorname{Prox}_{r\phi}(x_k - r g_k, \delta)$ are chosen so that the Cauchy point produces sufficient decrease. When using inexact proximity operators, the stationarity metric (3.3) becomes

$$\tilde{h}_k := \frac{1}{r_0}\|\tilde{p}_k(u, r_0)\| \qquad \text{for some} \qquad u \in \operatorname{Prox}_{r\phi}(x_k - r_0 g_k, \delta). \tag{4.5}$$

To address the accuracy of $\tilde{h}_k$, we notice that the reverse triangle inequality ensures that

$$h_k \leq \tilde{h}_k + |h_k - \tilde{h}_k|.$$

Consequently, by choosing $\delta > 0$ so that

$$|h_k - \tilde{h}_k| \leq \kappa_{\text{grad}} \min\{\tilde{h}_k, \Delta_k\}, \tag{4.6}$$

holds, we ensure that

$$h_k \leq (1 + \kappa_{\text{grad}})\tilde{h}_k.$$

Hence, if the limit inferior of $\tilde{h}_k$ is zero, then so is the limit inferior of h_k. Given the computable quantity $\tilde{h}_k$, we further modify the trust-region algorithm to ensure that the trial iterate x_k^+ satisfies

$$\|x_k^+ - x_k\| \leq \kappa_{\text{rad}} \Delta_k \tag{4.7a}$$

$$m_k(x_k) - m_k(x_k^+) \geq \kappa_{\text{fcd}} \tilde{h}_k \min\left\{\frac{\tilde{h}_k}{1 + \omega_k}, \Delta_k\right\}. \tag{4.7b}$$

Finally, we modify Assumption 3.2 to account for $\tilde{h}_k$ as follows.

ASSUMPTION 4.1 (Inexact Gradient). *The model $f_k : X \to \mathbb{R}$ has Lipschitz continuous gradient on an open set containing* $\operatorname{dom}\phi$ *and there exists a positive constant κ_{grad}, independent of k, such that*

$$|h_k - \tilde{h}_k| \leq \kappa_{\text{grad}} \min\{\tilde{h}_k, \Delta_k\}$$
$$\|g_k - \nabla f(x_k)\| \leq \kappa_{\text{grad}} \min\{\tilde{h}_k, \Delta_k\}.$$

With these changes we are able to produce a new version of Algorithm 1 that allows inexact computation of the proximity operator. In particular, line 2 in Algorithm 1 is modified to *"Choose $\tilde{h}_k$ and m_k that satisfy Assumption 4.1"*. Notice that $\tilde{h}_k$ and g_k must be computed in tandem to satisfy the conditions in Assumption 4.1, which can be accomplished by a straight forward modification of [1, Algorithm 4]. In order to ensure that the Cauchy point achieves sufficient decrease, we additionally enforce the proximal gradient descent condition

$$\tilde{Q}_k(u, r) \leq -\frac{\kappa_{\text{dec}}}{r}\|\tilde{p}_k(u, r)\|^2, \tag{4.8}$$

where κ_{dec} is a positive constant that is independent of k. Note that if $u = \text{Prox}_{r\phi}(x_k - rg_k)$, then (4.8) is satisfied with $\kappa_{\text{dec}} = 1$ (cf. Lemma 3.1). To demonstrate how we enforce these conditions, we restrict our attention to the case of weighted inner products.

5. Weighted Proximal Operators. To motivate this section, consider the finite dimensional case $X = X_M$. When ϕ is separable, it is typically much easier to compute the proximity operator defined on X_D (recall D is a diagonal matrix), i.e.,

$$\text{Prox}^D_{r\phi}(x) = \arg\min_{y \in X} \left\{ \frac{1}{2r}\|y - x\|^2_D + \phi(y) \right\},$$

because the optimization problem defining $\text{Prox}^D_{r\phi}(x)$ can be reduced to solving one-dimensional optimization problems for each component of $\text{Prox}^D_{r\phi}(x)$. For example, this is the case when $\phi(\cdot) \equiv \|\cdot\|_1$ or when the ϕ is the indicator function for bound constraints.

We consider the more general setting of replacing $\text{Prox}_{r\phi}$ with $\text{Prox}^a_{r\phi}$ defined using the inner product induced by the bilinear form $a(\cdot, \cdot)$ and assume that $\text{Prox}^a_{r\phi}$ has an analytical form. To this end, we define an algorithm, listed as Algorithm 2 that computes a δ-proximity operator using $\text{Prox}^a_{r\phi}$. Algorithm 2 is the usual a-weighted proximal gradient algorithm applied to compute $\text{Prox}_{r\phi}(x)$ and consequently, the iterates $\{u_\ell\}$, ignoring the stopping conditions, converge strongly to $\text{Prox}_{r\phi}(x)$ [4, Corollary 28.9]. Notice that we employ a modified stopping condition that helps to ensure that the computed solution is a δ-proximity operator.

Algorithm 2 Weighted Proximal Gradient

Require: The point $x \in X$, the proximity parameter $r > 0$, and $\varepsilon > 0$

1: $\ell \leftarrow 0$
2: $u_0 \leftarrow \text{Prox}^a_{r\phi}(x)$
3: **while** $\varepsilon < \|u_\ell - u_{\ell+1}\|_a$ **do**
4: $\quad u_{\ell+1} \leftarrow \text{Prox}^a_{r\phi}(u_\ell - A^{-1}(u_\ell - x))$
5: $\quad \ell \leftarrow \ell + 1$
6: **end while**

To satisfy Assumption 4.1, we must bound $|h_k - \tilde{h}_k|$, which we can bound in terms of the error in the δ-proximity operator using the reverse triangle inequality, i.e.,

$$|h_k - \tilde{h}_k| \leq r_0^{-1} \|\mathrm{Prox}_{r\phi}(x_k - r_0 g_k) - \tilde{x}_k(u, r_0)\|,$$

where again $u \in \mathrm{Prox}_{r\phi}(x_k - r_0 g_k, \delta)$ for some $\delta > 0$. As we now demonstrate, Algorithm 2 yields an element that approximates the true proximity operator, $\mathrm{Prox}_{r\phi}(x, \delta)$, to arbitrary precision. For this result, we recall the definition $\varphi(\cdot) \coloneqq \frac{1}{2r}\|\cdot - x\|^2 + \phi(\cdot)$ from the proof of Theorem 4.4.

LEMMA 5.1. *Algorithm 2 converges in finitely many iterations. Moreover, if* $\alpha_1 \leq \sqrt{2}$ *and Algorithm 2 exits at iteration* ℓ*, i.e.,*

$$\|u_\ell - u_{\ell+1}\|_a \leq \varepsilon, \tag{5.1}$$

then the following error bound holds

$$\|u_\ell - \mathrm{Prox}_{r\phi}(x)\| \leq \alpha_1^{-1/2} \left(1 - \tfrac{1}{2}\alpha_1^{-2}\right)^{-1} \varepsilon.$$

Proof. By [4, Corollary 28.9], u_ℓ converges strongly and therefore there exists ℓ for which (5.1) holds for given $\epsilon > 0$. Denote the proximal gradient operator associated with the a-inner product by

$$G_a(y, t) = \tfrac{1}{t}\left(y - \mathrm{Prox}^a_{t\phi}\left(y - \tfrac{t}{r}A^{-1}(y - x)\right)\right)$$

and note that $y \mapsto G_a(y, t)$ is strongly monotone for all $t \in (0, 2r/\alpha_1^2)$ [3, Lemma 2]. Consequently, $y \mapsto G_a(y, r)$ is strongly monotone since $\alpha_1 \leq \sqrt{2}$ and

$$a(G_a(u_\ell, r) - G_a(p, r), u_\ell - p) \geq \tfrac{1}{r}\left(1 - \tfrac{1}{2}\alpha_1^{-2}\right)\|u_\ell - p\|_a,$$

where $p = \mathrm{Prox}_{r\phi}(x)$. The optimality of p for φ ensures that $G_a(p, r) = 0$ and so

$$\tfrac{1}{r}\left(1 - \tfrac{1}{2}\alpha_1^{-2}\right)\|u_\ell - p\|_a \leq \|G_a(u_\ell, r)\|_a = \tfrac{1}{r}\|u_{\ell+1} - u_\ell\|_a.$$

The result then follows from (5.1) and the equivalence of $\|\cdot\|_a$ and $\|\cdot\|$. □

In Theorem 5.3, we demonstrate that Algorithm 2 applied to $x_k - rg_k$ outputs $u_{\ell+1}$ satisfying the following inequality

$$\begin{aligned}\langle g_k, u_{\ell+1} - x_k\rangle + \tfrac{1}{2r}\|u_{\ell+1} - x_k\|^2 + \phi(u_{\ell+1}) \\ \leq \langle g_k, z - x_k\rangle + \frac{1}{2r}\|z - x_k\|^2 + \phi(z) + \delta\|z - u_{\ell+1}\|\end{aligned}$$

for every $z \in X$, or equivalently

$$u_{\ell+1} \in \mathrm{Prox}_{r\phi}(x_k - rg_k, \delta).$$

Before presenting the main theorem of this section, we prove the following preliminary result, which we use to facilitate the proof of our main result.

LEMMA 5.2. *Consider the sequence* $\{u_\ell\}$ *generated by Algorithm 2. Then,*

$$\frac{1}{r}(A - I)(u_\ell - u_{\ell+1}) \in \partial\varphi(u_{\ell+1}), \tag{5.2}$$

which by definition is equivalent to

$$\varphi(z) \geq \varphi(u_{\ell+1}) + \frac{1}{r}\langle (A - I)(u_\ell - u_{\ell+1}), z - u_{\ell+1}\rangle \quad \forall\, z \in X. \tag{5.3}$$

Proof. Recall that $u_{\ell+1} = \mathrm{Prox}^a_{r\phi}(u_\ell - A^{-1}(u_\ell - x))$. By Theorem 4.4, we have that

$$\frac{1}{r}(u_\ell - u_{\ell+1} - A^{-1}(u_\ell - x)) \in \partial_A \phi(u_{\ell+1}) = A^{-1}\partial\phi(u_{\ell+1})$$

and by adding $(1/r)A^{-1}(u_{\ell+1} - x)$ on both sides of the previous set inclusion (in the Minkowski sense), we obtain

$$\begin{aligned}\frac{1}{r}(u_\ell - u_{\ell+1} - A^{-1}(u_\ell - u_{\ell+1})) &\in \frac{1}{r}A^{-1}(u_{\ell+1} - x) + A^{-1}\partial\phi(u_{\ell+1}) \\ &= A^{-1}\partial\varphi(u_{\ell+1}).\end{aligned}$$

This is equivalent to (5.2), concluding the proof. □

Finally, we present our main result, which shows that if $u_{\ell+1}$ satisfies the stopping conditions of Algorithm 2, then it is a δ-proximity operator.

THEOREM 5.3. *If $u_{\ell+1}$ satisfies the stopping condition (5.1) with*

$$\varepsilon \le r\delta\sqrt{\alpha_1}/(1+\alpha_2),$$

then $u_{\ell+1} \in \mathrm{Prox}_{r\phi}(x, \delta)$, i.e., for all $z \in X$,

$$\varphi(z) + \delta\|z - u_{\ell+1}\| \ge \varphi(u_{\ell+1}).$$

Proof. First note that the existence of α_1 and α_2 ensure that

$$\frac{1}{r}\|(A-I)(u_\ell - u_{\ell+1})\| \le \frac{1+\alpha_2}{r}\|u_\ell - u_{\ell+1}\| \le \frac{1+\alpha_2}{r\sqrt{\alpha_1}}\|u_\ell - u_{\ell+1}\|_a \le \frac{1+\alpha_2}{r\sqrt{\alpha_1}}\varepsilon \le \delta.$$

Now, for any $z \in X$, we have that

$$\begin{aligned}\varphi(z) + \delta\|z - u_{\ell+1}\| &\ge \varphi(z) + \frac{1}{r}\|(A-I)(u_\ell - u_{\ell+1})\|\|z - u_{\ell+1}\| \\ &\ge \varphi(z) + \frac{1}{r}\langle (A-I)(-u_\ell + u_{\ell+1}), z - u_{\ell+1}\rangle \\ &\ge \varphi(u_{\ell+1})\end{aligned}$$

where in the first line we applied the bound on δ, in the second line we used the Cauchy-Schwarz inequality and for the final inequality we applied Lemma 5.2. □

6. Numerics. We apply our inexact weighted proximity operator within Algorithm 1 to solve optimal control of Burgers' equation:

$$\min_{z \in L^2(\Omega)} \int_\Omega ([S(z)] - w)^2(x)\mathrm{d}x + \frac{\alpha}{2}\int_\Omega z^2(x)\mathrm{d}x + \beta\int_\Omega |z|(x)\mathrm{d}x \tag{6.1}$$

where $\Omega = (0,1)$ is the physical domain, $\alpha = 10^{-4}$ and $\beta = 10^{-2}$ are penalty parameters, $w(x) = -x^2$ is the target state, and $S(z) = u \in H^1(\Omega)$ solves the weak form of Burgers' equation

$$\begin{aligned}-\nu u'' + uu' &= z + f \quad \text{in } \Omega, \\ u(0) = 0, &\quad u(1) = -1,\end{aligned}$$

where $f = 2(\nu + x^3)$ and $\nu = 0.08$. We discretize the state u and z using continuous piecewise linear finite elements on a uniform mesh with $n = 512$ intervals. To compute $S(z)$, we solve

the discretized Burgers' equation using Newton's method globalized with a backtracking line search. We exit the Newton iteration when the relative residual falls below $10^{-4}\sqrt{\epsilon_{\text{mach}}}$, where ϵ_{mach} is machine epsilon. We will refer to PDE solves using this tolerance as "exact" PDE solves.

Given that the controls are in $L^2(\Omega)$, we employ the weighted dot product $\langle\cdot,\cdot\rangle = \langle\cdot,\cdot\rangle_M$, where $M \in \mathbb{R}^{n\times n}$ is the mass matrix and for Algorithm 2, we employ the dot product $a(\cdot,\cdot) = \langle\cdot,\cdot\rangle_D$ weighted by the lumped mass matrix $D \in \mathbb{R}^{n\times n}$, i.e.,

$$M = \frac{h}{6}\begin{pmatrix} 4 & 1 & \dots & 0 & 0 \\ 1 & 4 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 4 & 1 \\ 0 & 0 & \dots & 1 & 4 \end{pmatrix} \in \mathbb{R}^{n\times n} \quad \text{and} \quad D = h\begin{pmatrix} \frac{5}{6} & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & \frac{5}{6} \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

Recall that the A operator generated by the bilinear form $a(\cdot,\cdot)$ is $M^{-1}D$. For this A, $\alpha_1 = 1$ and $\alpha_2 = 3$. Consequently, the preceding theory applies. Furthermore, we approximate the L^1-norm by the quantity

$$\beta\int_\Omega |z|(x)\mathrm{d}x \approx \phi(z) = \beta h(\tfrac{5}{6}|z_1| + |z_2| + \ldots + |z_{n-1}| + \tfrac{5}{6}|z_n|).$$

We notice that the D-weighted proximity operator of ϕ is the usual soft-thresholding operator

$$\text{Prox}^D_{r\phi}(z) = \text{sign}(z)\odot\max\{|z| - \beta r, 0\}.$$

In Table 6.1, we summarize the performance of Algorithm 1 for different values of scaling (κ_{grad}) for the inexact gradient tolerance. To approximately solve the trust-region subproblem (3.1), we employ the nonlinear conjugate gradient solver introduced in [2, Algorithm 4]. The table includes the wallclock time in seconds (`time (s)`), the number of trust-region iterations (`iter`), the number of evaluations of f (`obj`), the number of evaluations of ∇f (`grad`), the number of applications of the Hessian $\nabla^2 f$ (`hess`), the number of evaluations of the proximity operator (`prox`), and the average number of iterations of Algorithm 2 (`av-piter`). As κ_{grad} decreases, we require additional accuracy from our approximate proximity operator as computed by Algorithm 2. As seen in Table 6.1, our implementation of Algorithm 1 is robust to inexact proximity operators.

κ_{grad}	time (s)	iter	obj	grad	hess	prox	av-piter
1e2	0.4708	18	37	19	173	347	10.78
1e1	0.4447	16	33	17	141	281	16.63
1e0	0.5229	18	37	19	173	347	28.50
1e-1	0.4247	15	31	16	125	248	47.07
1e-2	0.5815	17	35	19	157	315	60.00
1e-3	0.3498	13	27	15	93	183	53.15
1e-4	0.4034	13	27	17	93	185	69.31

TABLE 6.1
Results for the Burgers' equation for $\kappa_{\text{grad}} \in \{100, 10, 0.1, 0.01, 0.001, 0.0001\}$

To conclude, we incorporate inexact PDE solves with our inexact proximity operator computations. In particular, we terminate the Newton iteration for solving Burgers' equation

when the relative residual falls below

$$\min\{10^{-2}, \tau\},$$

where τ is computed by the trust-region algorithm and corresponds to the right-hand sides in Assumptions 3.1 and 3.2. We do this fixing to $\kappa_{\text{grad}} = 1$ and choosing $\kappa_{\text{obj}} = 10^3$. Since the dominant cost of Newton's method is the linear system solves at each iteration, we compare the average number of linear system solves per iteration between our runs with exact and inexact PDE solves. In particular, our method averaged 5.3125 linear system solves per trust-region iteration when using inexact PDE solves, compared with 7.7222 linear system solves when using exact PDE solves. The reduction in linear system solves is a byproduct of our relaxed tolerances. In particular, many of the PDE solves—the primary cost in PDE-constrained optimization—only require the relative residual to be smaller than 10^{-2}. Relaxing the PDE solver tolerance has the potential to make previously intractable problems solvable.

7. Conclusion and Future Work. In this work, we introduced an inexact proximity operator—motivated by the δ-Fréchet subdifferential—for use within the inexact trust-region algorithm from [1]. Additionally, we extended the inexact trust-region algorithm, Algorithm 1, from [1], to leverage our inexact proximity operator. Our numerical results suggest that our algorithm is robust to inexact proximity operator evaluations. As future research, we hope to develop similar theory that extends beyond the weighted proximity operator problem studied here.

REFERENCES

[1] R. Baraldi and D. Kouri, *A proximal trust-region method for nonsmooth optimization with inexact function and gradient evaluations*, Math. Program., (2023).

[2] R. J. Baraldi and D. P. Kouri, *Efficient proximal subproblem solvers for a nonsmooth trust-region method*, Optimization Online, (2024), pp. 1–29.

[3] ———, *Local convergence analysis of an inexact trust-region method for nonsmooth optimization*, Optimization Letters, 18 (2024), pp. 663–680.

[4] H. H. Bauschke, P. L. Combettes, H. H. Bauschke, and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, 2017.

[5] A. Beck, *First-order methods in optimization*, SIAM, 2017.

[6] C. Clason and T. Valkonen, *Introduction to nonsmooth analysis and optimization*, 2023.

[7] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods*, SIAM, 2000.

[8] O. Devolder, F. Glineur, and Y. Nesterov, *First-order methods of smooth convex optimization with inexact oracle*, Mathematical Programming, 146 (2014), pp. 37–75.

[9] L. Farias Maia, D. H. Gutman, and R. C. Hughes, *The inexact cyclic block proximal gradient method and properties of inexact proximal maps*, Journal of Optimization Theory and Applications, (2024).

[10] X. Hua and N. Yamashita, *An inexact coordinate descent method for the weighted l1-regularized convex optimization problem*, Pacific Journal of Optimization, 9 (2013).

[11] K. Jiang, D. Sun, and K.-C. Toh, *An inexact accelerated proximal gradient method for large scale linearly constrained convex sdp*, SIAM Journal on Optimization, 22 (2012), pp. 1042–1064.

[12] D. P. Kouri and D. Ridzal, *Inexact trust-region methods for PDE-constrained optimization*, Frontiers in PDE-constrained optimization, (2018), pp. 83–121.

[13] B. S. Mordukhovich, *Variational Analysis and Generalized Differentiation I: Basic Theory*, vol. 330, Springer Science & Business Media, 2006.

[14] S. Salzo, S. Villa, et al., *Inexact and accelerated proximal point algorithms*, Journal of Convex analysis, 19 (2012), pp. 1167–1192.

[15] R. Tappenden, P. Richtárik, and J. Gondzio, *Inexact coordinate descent: complexity and preconditioning*, Journal of Optimization Theory and Applications, 170 (2016), pp. 144–176.

DOMAIN DECOMPOSITION-BASED COUPLING OF OPERATOR INFERENCE REDUCED ORDER MODELS VIA THE SCHWARZ ALTERNATING METHOD

IAN MOORE*, CHRISTOPHER R. WENTLAND†, ANTHONY GRUBER‡, AND IRINA TEZAUR§

Abstract. This paper presents and evaluates an approach for coupling subdomain-local reduced order models (ROMs) constructed via non-intrusive operator inference (OpInf) with each other and with subdomain-local full order models (FOMs), following a domain decomposition of the spatial geometry on which a given partial differential equation (PDE) is posed. Joining subdomain-local models is accomplished using the overlapping Schwarz alternating method, a minimally-intrusive multiscale coupling technique that transforms a monolithic problem into a sequence of subdomain-local problems, which communicate through transmission boundary conditions imposed on the subdomain interfaces. After formulating the overlapping Schwarz alternating method for OpInf ROMs, we evaluate the method's accuracy and efficiency on several test cases involving the heat equation in two spatial dimensions. We demonstrate that the method is capable of coupling arbitrary combinations of OpInf ROMs and FOMs, and that moderate speed-ups over a monolithic FOM are possible when performing OpInf ROM coupling.

1. Introduction. Despite advancements in both computer architectures and algorithms, the modeling and simulation of complex physical systems often requires tremendous computational resources. These requirements may preclude many-query analyses such as engineering design or uncertainty quantification. While projection-based reduced order models (ROMs) have shown promise to mitigate this difficulty, traditional intrusive ROMs, e.g., Galerkin [12, 36] and least squares Petrov-Galerkin (LSPG) projection ROMs [4], have their own shortcomings, including a lack of systematic refinement mechanisms, a lack of robustness, stability, and accuracy in the predictive regime, and lengthy implementation time requirements.

This paper presents an approach for mitigating the aforementioned difficulties by enabling domain decomposition- (DD-)based coupling of subdomain-local ROMs with each other and/or with subdomain-local full order models (FOMs). Our approach is based on the following ingredients: (i) a decomposition of the physical domain of interest into two or more overlapping subdomains, (ii) the construction of subdomain-local ROMs and/or FOMs in each of the subdomains, and (iii) the rigorous coupling of the subdomain-local models via the Schwarz alternating method [33]. The Schwarz alternating method is based on the simple idea that, if the solution to a partial differential equation (PDE) is known in two or more regularly shaped domains, these local solutions can be used to iteratively build a solution for the union of the subdomains, with information propagating between the subdomains through carefully constructed transmission boundary conditions (BCs). We choose the Schwarz alternating method since it has a number of advantages over competing multiscale coupling methods. These advantages include its concurrent nature, its ability to couple non-conformal meshes with different element topologies and different time integrators with different time steps for dynamic problems without introducing non-physical artifacts into the solution, and its non-intrusive implementation into existing codes [24, 25].

Building on our past work in developing the Schwarz alternating method as a means to couple FOMs [24, 25], intrusive projection-based ROMs [3] and physics-informed neural networks (PINNs) [37], we focus our attention herein on advancing the method to work with a non-intrusive model order reduction (MOR) technique known as operator inference

*Virginia Tech, ianm9123@vt.edu,
†Sandia National Laboratories, crwentl@sandia.gov
‡Sandia National Laboratories, adgrube@sandia.gov
§Sandia National Laboratories, ikalash@sandia.gov

(OpInf) [11, 16, 27]. Unlike traditional intrusive MOR, which requires access to the underlying FOM code in order to project the governing PDE(s) onto a reduced subspace, OpInf works by assuming a functional form (usually linear or quadratic [10]) for the ROM in terms of to-be-learned reduced operators, and solving an optimization problem offline for these operators. This procedure significantly reduces both the development time and the time-to-impact.

DD-based FOM-ROM and ROM-ROM couplings such as those proposed herein have the potential of improving the predictive viability of projection-based ROMs, by enabling the spatial localization of ROMs (via domain decomposition) and the online integration of high-fidelity information into these models (via FOM coupling). While DD-based couplings between ROMs and FOMs are not new, the majority of the literature on this topic has focused on developing intrusive coupling methods (e.g., Lagrange multipliers, optimization-based coupling) used to couple intrusive ROMs; the interested reader is referred to [3, 5, 7, 8, 13, 20, 28] and the references therein for more details. Related past work on data-driven couplings using Schwarz-like methods has focused on intrusive ROMs [6, 14, 15, 29], or on utilizing the coupling to accelerate NN training [18, 19, 37]. The proposed approach is most similar to the recent work by Farcas *et al.* [9], which develops a DD-based coupling of subdomain-local OpInf ROMs by learning appropriate reduced operators responsible for the coupling, and demonstrates the method on a formidable three-dimensional (3D) combustion problem. In this approach, each subdomain problem is solved once rather than by performing an iteration to convergence as done within our Schwarz framework. As a result, the subdomain-local solutions must be extended to the full domain and smoothly combined to achieve a continuous solution.

The remainder of this paper is organized as follows. In Section 2, we describe the overlapping version of the Schwarz alternating method applied to our model problem, the two-dimensional (2D) unsteady heat equation. In Section 2.2, we present some OpInf preliminaries. In Section 3, we describe our software implementations of the proposed Schwarz-based coupling approach applied to OpInf ROMs, hereafter called the OpInf-Schwarz method, which makes use of the open source `FEniCSx` [2] and `OpInf Python` libraries. Numerical results are presented in Section 4. We conclude with a summary and a discussion of future work in Section 5.

2. The Schwarz Alternating Method Applied to Operator Inference ROMs. In the following sections and subsections, we consider the specific model problem of the 2D heat equation, towards addressing the challenges that are encountered in this novel combination of the Schwarz method and operator inference. We stress that neither of these techniques inherently require an assumption of linearity, noting that the authors of the original operator inference paper [27] directly contrasted their method with the linearity assumption of Dynamic Mode Decomposition (DMD) [32]. While non-linearity will likely present further challenges to overcome for the OpInf-Schwarz coupling method, the heat equation provides an initial unsteady test problem that can suggest the feasibility of the method for more complicated problems.

2.1. Schwarz Alternating Method Preliminaries. Consider the heat equation specified as,

$$\begin{aligned} \dot{u}(x,t) - \Delta u(x,t) &= 0, && \text{in } \Omega \times [0,T], \\ u(x,t) &= g(x), && \text{on } \partial\Omega \times [0,T], \\ u(x,0) &= v(x), && \text{in } \Omega, \end{aligned} \tag{2.1}$$

where $\Omega \in \mathbb{R}^p$ is an open bounded domain for $p = 1, 2, 3$ with boundary $\partial\Omega$, $g(x)$ are is a given boundary condition function, $v(x)$ defines the initial condition, and $T > 0$. Suppose we

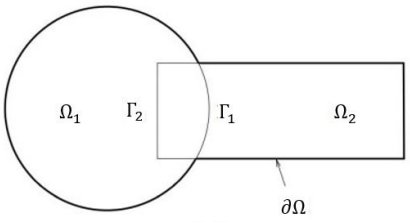


Fig. 2.1: Illustration showing an overlapping domain decomposition of a 2D domain Ω for the application of the Schwarz alternating method. Note $\Gamma_1, \Gamma_2 \not\subset \partial\Omega$.

decompose the domain Ω into two overlapping subdomains Ω_1 and Ω_2, such that $\Omega_1 \cup \Omega_2 = \Omega$ and $\Omega_1 \cap \Omega_2 \neq \emptyset$ and, for later use, define $\overline{\Omega_1}$ as the closure of Ω_1 and similarly for Ω_2. Suppose also that we decompose the time interval $[0, T]$ into a set of non-overlapping time intervals $I_n = [t_n, t_{n+1}]$, where $T \geq t_{n+1} > t_n \geq 0$, so that $\cup_n I_n = [0, T]$. For a given time interval I_n, the overlapping Schwarz algorithm solves the following sequence of subdomain-local problems:

$$\left\{ \begin{array}{rcll} \dot{u}_1^{(k+1)} - \Delta u_1^{(k+1)} & = & 0, & \text{in } \Omega_1 \times [t_n, t_{n+1}] \\ u_1^{(k+1)} & = & g, & \text{on } (\partial\Omega \cap \overline{\Omega_1}) \times [t_n, t_{n+1}] \\ u_1^{(k+1)} & = & u_2^{(k)}, & \text{on } \Gamma_1 \times [t_n, t_{n+1}], \end{array} \right. \tag{2.2}$$

and

$$\left\{ \begin{array}{rcll} \dot{u}_2^{(k+1)} - \Delta u_2^{(k+1)} & = & 0, & \text{in } \Omega_2 \times [t_n, t_{n+1}] \\ u_2^{(k+1)} & = & g, & \text{on } (\partial\Omega \cap \overline{\Omega_2}) \times [t_n, t_{n+1}] \\ u_2^{(k+1)} & = & u_1^{(k+1)}, & \text{on } \Gamma_2 \times [t_n, t_{n+1}], \end{array} \right. \tag{2.3}$$

for Schwarz iteration $k = 0, 1, 2, ...$, subject to initial conditions $u_1(x, 0) = v|_{\Omega_1}$ and $u_2(x, 0) = v|_{\Omega_2}$. In (2.2), u_i for $i = 1, 2$ denotes the solution in subdomain Ω_i, and Γ_i is the so-called Schwarz boundary (see Figure 2.1). It is common to set $u_1^{(0)} = v|_{\partial\Omega_1}$ on Γ_1 and $u_2^{(0)} = v|_{\partial\Omega_2}$ on Γ_2 when initializing the Schwarz iteration process, to ensure solution compatibility with the initial condition. The iterative process in (2.2) and (2.3) continues until a set of pre-determined criteria are met. In the present work, convergence criteria are based on the Euclidean norm of the solution differences between consecutive Schwarz iterations; that is, Schwarz is deemed converged when $\epsilon_{\text{abs}}^{(k)} < \delta_{\text{abs}}$ and $\epsilon_{\text{rel}}^{(k)} < \delta_{\text{rel}}$ for some pre-specified Schwarz tolerances $\delta_{\text{abs}}, \delta_{\text{rel}} > 0$, where

$$\epsilon_{\text{abs}}^{(k)} := \sqrt{||u_1^{(k)} - u_1^{(k-1)}||^2 + ||u_2^{(k)} - u_2^{(k-1)}||^2}, \tag{2.4}$$

and

$$\epsilon_{\text{rel}}^{(k)} := \sqrt{\frac{||u_1^{(k)} - u_1^{(k-1)}||^2}{||u_1^{(k)}||^2} + \frac{||u_2^{(k)} - u_2^{(k-1)}||^2}{||u_2^{(k)}||^2}}, \tag{2.5}$$

for Schwarz iteration $k = 1, 2, ...$, until convergence.

A key advantage of the Schwarz alternating method is that it allows for the subdomains Ω_1 and Ω_2 to be discretized using different meshes and/or element types [24, 25]. In the

case where Ω_1 and Ω_2 are discretized using different meshes and do not have a coincident interface, applying the Schwarz boundary condition on the Schwarz boundaries Γ_i requires the construction of a projection operator; this can be done via a simple application of finite element interpolation functions readily available in most codes [23, 25].

The Schwarz iteration process (2.2)–(2.3) is converged within each time interval $[t_n, t_{n+1}]$ before moving on to the next time interval. A key advantage of this time-stepping approach is that different time integrators and time steps can be used in different subdomains; for a detailed discussion of Schwarz time-stepping and related machinery, the reader is referred to [25, 23]. In the numerical experiments presented herein, we restrict attention to the case where all subdomains have the same time integrator and time step, as our goal is to assess the method's viability when coupling subdomain-local operator inference-based ROMs with each other and with subdomain-local FOMs while eliminating other confounding factors.

2.1.1. FOM-FOM Schwarz Coupling. Continuing the example of the heat equation from (2.1), a spatially discretized monolithic FOM for the heat equation typically appears in the following form after a boundary lift:

$$\dot{\mathbf{x}} = \mathbf{K}\mathbf{x} + \mathbf{B}\mathbf{g}, \tag{2.6}$$

where $\mathbf{x} \in \mathbb{R}^N$ is a discretized vector corresponding to the unconstrained state degrees of freedom (DoFs), and $\mathbf{g} \in \mathbb{R}^m$ discretizes the Dirichlet boundary condition. The matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ comes from discretizing the continuous Laplace operator Δ, and $\mathbf{B} \in \mathbb{R}^{N\times m}$ deals with effects of the boundary condition. The full state representation $\mathbf{u} \in \mathbb{R}^{N+m}$ for all DoFs is obtained by augmenting the unconstrained solution $\mathbf{x}$ with the known boundary condition $\mathbf{g}$.

If a domain decomposition such as in Figure 2.1 is imposed, we may consider the subdomain-local discretized problems:

$$\dot{\mathbf{x}}_i = \mathbf{K}_i\mathbf{x}_i + \mathbf{B}_i\mathbf{y}_i, \quad i = 1, 2, \tag{2.7}$$

on Ω_1 and Ω_2. Let N_i and m_i denote the number of finite element (FE) nodes on the interior and boundary of Ω_i, respectively. Note that the boundary of Ω_i includes Γ_i, as shown in Figure 2.1. We introduce the vector $\mathbf{y}_i \in \mathbb{R}^{m_i}$ to handle the subdomain-local boundary by defining $\mathbf{y}_i = [\mathbf{g}_i \ \gamma_i]^T$, where $\mathbf{g}_i$ discretizes $g(x)$ on $(\partial\Omega \cap \overline{\Omega_i})$ and γ_i discretizes the subdomain-local boundary condition on Γ_i.

The remaining quantities in (2.7) follow naturally from the monolithic terms in (2.6), with subdomain-local state representation $\mathbf{x}_i \in \mathbb{R}^{N_i}$ as well as stiffness and boundary matrices $\mathbf{K}_i \in \mathbb{R}^{N_i\times N_i}$ and $\mathbf{B}_i \in \mathbb{R}^{N_i\times m_i}$. We now use the Schwarz method to set up the subdomain-local problems as follows:

$$\begin{cases} \dot{\mathbf{x}}_1^{(k+1)} &= \mathbf{K}_1\mathbf{x}_1^{(k+1)} + \mathbf{B}_1\mathbf{y}_1^{(k+1)} \\ \gamma_1^{(k+1)} &= \mathbf{x}_2^{(k)}\big|_{\Gamma_1}, \end{cases} \tag{2.8}$$

and

$$\begin{cases} \dot{\mathbf{x}}_2^{(k+1)} &= \mathbf{K}_2\mathbf{x}_2^{(k+1)} + \mathbf{B}_2\mathbf{y}_2^{(k+1)} \\ \gamma_2^{(k+1)} &= \mathbf{x}_1^{(k+1)}\big|_{\Gamma_2}, \end{cases} \tag{2.9}$$

for Schwarz iteration $k = 0, 1, 2, ...$, until convergence following (2.4)-(2.5). For work presented in this paper, we solve equations (2.8)-(2.9) using the finite element method in space and the finite difference method in time. We take the initial condition to be the interpolation of $u(x, 0) = v(x)$ into the respective finite element spaces. This specific formulation is valid for conformal spatial meshes only.

2.2. Operator Inference Preliminaries. Operator Inference is a data-driven, non-intrusive, projection-based method for model order reduction, developed by Peherstorfer and Willcox [27] as an alternative to standard Galerkin projection ROMs. Similarly to other Galerkin projection ROMs, the method begins by constructing a reduced basis from data. OpInf then diverges from the standard method in the construction of the reduced operators, which are estimated using data through a regression problem.

2.2.1. Proper Orthogonal Decomposition. To construct a reduced basis, we use proper orthogonal decomposition (POD) [12, 17, 36], though one might also use other methods, e.g., the reduced basis method with a greedy algorithm [30]. To perform POD, we require snapshots from some data source or FOM.

Continuing with the heat equation example, after the monolithic problem (2.6) has been solved in time for τ separate states inclusively between time $t = 0$ and $t = T$ (i.e., $0 = t_1 < t_2 < \cdots < t_\tau = T$), we obtain a collection of unconstrained state snapshots,

$$\mathbf{X} = [\mathbf{x}(t_1), \mathbf{x}(t_2), \ldots, \mathbf{x}(t_\tau)] \in \mathbb{R}^{N\times\tau}, \tag{2.10}$$

with $\text{rank}(\mathbf{X}) = d > 0$. $\mathbf{X}$ yields a singular value decomposition (SVD) $\mathbf{X} = \mathbf{\Psi\Sigma\Phi}^*$, where $\mathbf{\Psi} \in \mathbb{R}^{N\times N}$ is an orthogonal matrix whose first d columns form a basis for the column space of $\mathbf{X}$. The columns of $\mathbf{\Psi}$, ψ_i for $i = 1, \ldots r$ with $r \leq d$ form an optimal r-dimensional basis for the columns of $\mathbf{X}$ in an ℓ^2 sense. The restriction of $\mathbf{\Psi}$ to its first r columns is referred to as $\mathbf{\Psi}_r \in \mathbb{R}^{N\times r}$ going forwards.

2.2.2. Operator Inference. Operator inference is a non-intrusive, data-driven, projection-based model order reduction technique which learns low-dimensional operators that can be used to approximate the output of a given FOM, which, in this paper, is the heat equation described in (2.6). In a classical intrusive Galerkin ROM, the operator $\mathbf{K}_r \in \mathbb{R}^{r\times r}$ is the operator which best represents the action of $\mathbf{K}$ in the reduced space of dimension r defined by the basis $\mathbf{\Psi}_r$. That is,

$$\mathbf{K}_r = \mathbf{\Psi}_r^T \mathbf{K} \mathbf{\Psi}_r. \tag{2.11}$$

This is intrusive because it requires access to the FOM matrix $\mathbf{K}$, and, practically speaking, the code that produced $\mathbf{K}$.

The key difference between operator inference and traditional intrusive projection-based MOR is that, in the former approach, the reduced operators are not created via intrusive projection, but inferred directly using available snapshot data. OpInf is based on the observation that a projection-based ROM derived from a FOM with polynomial nonlinearities will possess the same algebraic structure as the FOM.

For the problem considered herein, the 2D heat equation, it is straightforward to see that a projection-based ROM preserves the linear algebraic structure of the FOM, (2.6). With this observation, we now set up the semi-discretized monolithic OpInf ROM problem:

$$\dot{\hat{\mathbf{x}}} = \hat{\mathbf{K}}\hat{\mathbf{x}} + \hat{\mathbf{B}}\mathbf{g}. \tag{2.12}$$

In order to obtain the reduced operators without access to FOM matrices, one solves a regression problem for the reduced matrices $\hat{\mathbf{K}} \in \mathbb{R}^{r\times r}$ and $\hat{\mathbf{B}} \in \mathbb{R}^{r\times m}$. In particular, given $j \leq \tau$ steps of FOM training data $\mathbf{x}_p = \mathbf{x}(t_p)$, $p = 1, \ldots, j$, minimize,

$$\min_{\hat{\mathbf{K}},\hat{\mathbf{B}}} \sum_{p=1}^{j} \|\dot{\hat{\mathbf{x}}}_p - \hat{\mathbf{K}}\hat{\mathbf{x}}_p - \hat{\mathbf{B}}\mathbf{g}\|_2^2. \tag{2.13}$$

Within the training data, we define the ROM representation of the state at time t_p to be $\hat{\mathbf{x}}_p = \mathbf{\Psi_r^T} \mathbf{x}_p \in \mathbb{R}^r$. The time derivative of the ROM state, $\dot{\hat{\mathbf{x}}}_p \in \mathbb{R}^r$, may not be explicitly available and can instead be estimated from available data using a difference method. The boundary information, $\mathbf{g} \in \mathbb{R}^m$, is the same as in the monolithic FOM.

We now briefly contextualize this in a classical Galerkin ROM context. A typical projection-based ROM creates matrices which are in some way representative of those constructed for the FOM, but OpInf does not start with the assumption that these FOM matrices are available. A more natural comparison is between OpInf and the standard Galerkin ROM for the choice of a particular shared basis $\mathbf{\Psi}_r$.

It is guaranteed that, under the assumptions that the time-stepping scheme is convergent as $dt \to 0$, the approximation $\dot{\hat{\mathbf{x}}}_j$ converges to the true time derivative at time t_j as $dt \to 0$, and linear independence of supplied data ($d \geq \tau$, the total number of snapshots), then, for all $0 < \epsilon \in \mathbb{R}$ there exists a timestep dt for the FOM and a $r \leq d$ such that,

$$\|\mathbf{K}_r - \hat{\mathbf{K}}\|_F < \epsilon.$$

The same result holds for $\hat{\mathbf{B}}$ [27]. In the case that a technique known as re-projection is used for the fully discretized problem, it can be shown that the learned OpInf ROM system recovers the standard Galerkin projected intrusive ROM [26].

While this overview has focused on the heat equation, we again stress that operator inference is general to low-order polynomial non-linearity, with results previously shown for a polynomial non-linearity of degree three in a one-dimensional (1D) nuclear reactor model in [27], for a 2D single injector combustion model in [21], and for a 3D rotating detonation rocket engine in [9], among others.

2.3. Opinf-Schwarz Method: FOM-ROM Coupling. We are now in a position to state the Opinf-Schwarz method for the heat equation specified in (2.1). We state the problem formulation for the case of two overlapping subdomains Ω_1 and Ω_2 with $\Omega_1 \cup \Omega_2 = \Omega$ as in Figure 2.1 where we place a FOM on Ω_1, an OpInf ROM on Ω_2, and couple the problems via the Schwarz method. The Opinf-Schwarz formulation can be extended easily to arbitrary numbers of subdomains and combinations of model couplings. For simplicity, we assume that we have access to monolithic FOM training data across the entire domain Ω. The dimensional and subdomain notation presented here follows from that defined for the FOM-FOM coupling in Section 2.1.1.

The OpInf problem on Ω_2 requires some details. Given $j \leq \tau$ steps of monolithic data $\mathbf{u}(t_p) \in \mathbb{R}^{N+m}, p = 1, \ldots, j$ extract the unconstrained state representation on Ω_2, $\mathbf{x}_2(t_p) \in \mathbb{R}^{N_2}$ as well as the the vector $\gamma_2(t_p)$, which is the state information along the Schwarz boundary Γ_2 at time t_p. The subdomain local boundary on Ω_2 is completed by defining the vector $\mathbf{y}_2(t_p) = [\mathbf{g}_2 \ \gamma_2(t_p)]^T \in \mathbb{R}^{m_2}$. We carry out POD on the collection of unconstrained snapshots on Ω_2, $\mathbf{x}_2(t_p), p = 1, \ldots, j$ to obtain the Ω_2-local basis $\mathbf{\Psi}_{r,2} \in \mathbb{R}^{N_2 \times r}$ for a choice of basis dimension r. Once the basis is obtained, we solve the regression problem,

$$\min_{\hat{\mathbf{K}}_2, \hat{\mathbf{B}}_2} \sum_{p=1}^{j} \|\dot{\hat{\mathbf{x}}}_2(t_p) - \hat{\mathbf{K}}_2 \hat{\mathbf{x}}_2(t_p) - \hat{\mathbf{B}}_2 \mathbf{y}_2(t_p)\|_2^2, \tag{2.14}$$

for the reduced operators $\hat{\mathbf{K}}_2 \in \mathbb{R}^{r \times r}$ and $\hat{\mathbf{B}}_2 \in \mathbb{R}^{r \times m_i}$ as in Section 2.2.

After setting up the FOM on Ω_1 exactly as in Section 2.1.1, the problems may be coupled using the Schwarz method as follows:

$$\begin{cases} \dot{\mathbf{x}}_1^{(k+1)} &= \mathbf{K}_1 \mathbf{x}_1^{(k+1)} + \mathbf{B}_1 \mathbf{y}_1^{(k+1)} \\ \gamma_1^{(k+1)} &= \mathbf{\Psi}_{r,2} \hat{\mathbf{x}}_2^{(k)} \big|_{\Gamma_1}, \end{cases} \tag{2.15}$$

and

$$\begin{cases} \dot{\hat{\mathbf{x}}}_2^{(k+1)} &= \hat{\mathbf{K}}_2 \hat{\mathbf{x}}_2^{(k+1)} + \hat{\mathbf{B}}_2 \mathbf{y}_2^{(k+1)} \\ \gamma_2^{(k+1)} &= \mathbf{x}_1^{(k+1)}\big|_{\Gamma_2}, \end{cases} \tag{2.16}$$

for Schwarz iteration $k = 0, 1, 2, \ldots$ until convergence following (2.4)-(2.5). $\boldsymbol{\Psi}_{r,2}\hat{\mathbf{x}}_2^{(k)}$ is the reconstruction of the ROM state $\hat{\mathbf{x}}_2^{(k)}$ on the FOM mesh originating from the data source.

The specific formulation described above is valid for conformal meshes only; some interpolation, evaluation or projection scheme would need to be employed for non-conformal meshes depending on the choice of spatial discretization for non-conformal meshes. Creating a ROM-ROM coupling in this setting only requires finding the reduced operators on Ω_1 and reconstructing the values found on the Schwarz boundary onto the FOM mesh. We extend this approach to arbitrary numbers of subdomains by traversing the domains in a round-robin fashion and use the most up-to-date Schwarz boundary information available at every Schwarz iteration.

3. Software Implementations. We now discuss specific choices made in the implementation of the Opinf-Schwarz method for this paper, in particular those which are not essential to the method itself. Our results include Schwarz coupling implementations for the FOM-FOM (Section 2.1.1) FOM-ROM (Section 2.3) and ROM-ROM (Adaptation of Section 2.3) scenarios. In our implementation, the generic FOM now specifically refers to a finite element simulation, and ROM refers to operator inference. All ordinary differential equations (ODEs), such as those presented for the Opinf-Schwarz method in equations (2.15)-(2.16), are discretized in time using backward Euler.

We use the open source FE library `FEniCSx` in `Python` for all FE simulations. `FEniCSx` is a FE problem solving environment with the ability to define variational forms close to actual mathematical notation [1], along with various tools [2, 34, 35] to enable the creation of a FE space on which computations can be performed. Boundary conditions are enforced strongly through a lifting method leading to FOM systems similar to (2.6).

Operator inference is performed with the `Python` library `OpInf`, a `PyPI Python` package available at https://pypi.org/project/opinf/. `OpInf` is a set of tools to facilitate learning polynomial reduced order models, applied in this paper as described in Section 2.2. The required data is drawn from a monolithic FE simulation on Ω, also performed in `FEniCSx`. When solving the OpInf regression problem (2.14), we estimate the required state derivative information $\dot{\hat{\mathbf{x}}}_2$ for through a first order backward difference of available state data.

The numerical stability of OpInf models is a significant concern and area of active research. OpInf models are not guaranteed to be stable, but there are a variety of strategies that can help improve stability of reduced models – independently of OpInf, data preprocessing is commonly used in ROMs to create a model for a transformed set of snapshots. In this paper, we use a centering approach where we build a model for snapshot data $\mathbf{x} - \bar{\mathbf{x}}$, the mean-centered snapshots.

Specific to OpInf, we also employ a regression regularization strategy for equation (2.13). When solving an OpInf regression problem like (2.13), we add a regularization form which converts the problem into the following form:

$$\min_{\hat{\mathbf{K}},\hat{\mathbf{B}}} \sum_{p=1}^{j} \|\dot{\hat{\mathbf{x}}}_p - \hat{\mathbf{K}}\hat{\mathbf{x}}_p - \hat{\mathbf{B}}\mathbf{g}\|_2^2 + \lambda^2 \left(\|\hat{\mathbf{K}}\|_F^2 + \|\hat{\mathbf{B}}\|_F^2 \right). \tag{3.1}$$

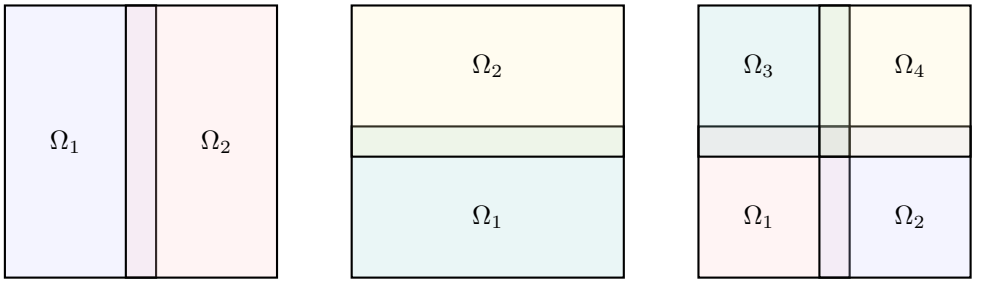
By adding this regularization, we penalize the large entries of our ROM matrices $\hat{\mathbf{K}}, \hat{\mathbf{B}}$. In our testing, this centering and regularization strategy has been sufficient to produce a

stable ROM ODE. For our results in Section 4, all tables and figures have been produced with regularization parameter $\lambda = 10^{-2}$. Regularization is a common necessity in operator inference, see for example [21, 31] for related approaches to regularization of OpInf models.

Finally, the Opinf-Schwarz method that couples these problems has been implemented by the authors of this paper in accordance with Section 2.1.1 and Section 2.3. To simplify the boundary transmission of the Schwarz process, only conformal meshes are used for the differing domains. The output of the Schwarz methods (FOM-FOM, ROM-ROM) as presented in Section 2 are unconstrained state vectors on the interiors of Ω_1 and Ω_2, when what is desired is a single approximation of the monolithic state across Ω. We have made the choice to reconcile the solutions in the following manner.

Taking the FOM-FOM coupling from Section 2.1.1 as an example, the Opinf-Schwarz method produces $\{\mathbf{x}_1(t_p)\}_{p=1}^{\tau}$ and $\{\mathbf{x}_2(t_p)\}_{p=1}^{\tau}$. We define the merged solution vector $\mathbf{u}(t_p)_{\text{SCHWARZ}} \in \mathbb{R}^N$ to be equal to the DoFs of $\mathbf{x}_i(t_p)$ on those entries associated with the interior of $\Omega_i \setminus \Omega_1 \cap \Omega_2$ and equal to the known boundary condition $\mathbf{g}$ on entries associated with $\partial\Omega$. For those entries associated with the overlap region $\Omega_1 \cap \Omega_2$, we assign the value of either the entry belonging to $\mathbf{x}_1(t_p)$ or $\mathbf{x}_2(t_p)$ depending on whether the physical node for that entry is closer to the center of Ω_1 or Ω_2.

4. Results. For the purposes of an initial test of OpInf-Schwarz, we are most interested if the model matches the standard behavior of ROMs and the Schwarz method. We expect that, as the ROM dimension r and the data allowance increases, the error should approach 0, and that as we increase the size of the Schwarz overlap, the number of Schwarz iterations to converge per timestep will decrease. Lastly, as for any ROM, we expect a significant speed-up in computation time for the price we pay in accuracy.



(a) Vertical configuration (b) Horizontal configuration (c) Four Squares configuration

Fig. 4.1: Overlapping domain decomposition configurations

We investigate each of these points in several different domain decomposition scenarios as shown to Figure 4.1. For a global domain, we choose the square $\Omega = [-1, 1] \times [-1, 1] \in \mathbb{R}^2$ on which a monolithic FE simulation is carried out. The monolithic boundaries are referred to as $\partial\Omega_L$ for the left, $\partial\Omega_R$ for the right, and $\partial\Omega_T$ and $\partial\Omega_B$ for the top and bottom of the domain respectively. We divide the domain into either two rectangles or four squares, each with an equal Schwarz overlap measured in the number of elements overlapping each neighboring subdomain. For these Schwarz overlaps, explicit communication is only between neighbors which share at least one complete edge –hence, in the Four Squares configuration, boundary information can be communicated to a horizontal or vertical neighbor, but not a diagonal neighbor (e.g. Ω_1 communicates directly with Ω_2 and Ω_3 but not Ω_4 in Figure 4.1c). The Schwarz subdomains are updated in ascending numerical order.

The specific problem we test the Opinf-Schwarz method on is the heat equation as specified in (2.1), so we fix the following quantities for the sake of comparison. Our time domain is $[0, 1]$, partitioned with a constant time step of $dt = 0.01$ leading to 101 steps, inclusively, from the initial condition at $t_1 = 0$ to the final evaluation at $t_T = 1$. The monolithic domain is be split into a 50×50 grid, and then triangular Lagrangian elements of degree 1 is formed by splitting the grid diagonally. These choices lead to 2601 FE nodes on the monolithic mesh, inclusive of the boundary. Within the Schwarz method, our relative (2.5) and absolute (2.4) tolerances for convergence are both be fixed at 10^{-10}.

Errors. Our typical measure of error for OpInf-Schwarz is the average pointwise relative error, i.e.,

$$\mathbf{E}_{\ell^2}^{\text{avg}} = \frac{1}{\tau - 1} \sum_{p=2}^{\tau} \frac{\|\mathbf{u}(t_p)_{\text{MONO}} - \mathbf{u}(t_p)_{\text{SCHWARZ}}\|_2}{\|\mathbf{u}(t_p)_{\text{MONO}}\|_2},$$

As in Section 3, $\mathbf{u}(t_p)_{\text{SCHWARZ}}$ refers to the solution via the Schwarz method merged from the component subdomains at time t_p, and $\mathbf{u}(t_p)_{\text{MONO}}$ refers to the monolithic finite element simulation on Ω at time t_p. The error determination excludes the given initial condition, but includes the boundary condition for computational convenience.

The projection error of the snapshots onto the POD basis in subdomain Ω_i (dependent on basis dimension r) is here defined as,

$$\mathbf{E}_{\text{proj}}^{\Omega_i} = \frac{\|\mathbf{X}_i - \mathbf{\Psi}_{r,i}\mathbf{\Psi}_{r,i}^T\mathbf{X}_i\|_F}{\|\mathbf{X}_i\|_F},$$

where $\mathbf{X}_j$ (chosen to recall the unconstrained state variable $\mathbf{x}$, which the POD basis is built for) are the snapshots associated with all solved for DoFs in Ω_i over the entire time domain, and $\mathbf{\Psi}_{r,i}$ is the r-dimensional basis associated with the solved for DoFs of Ω_i. This gives a measure of the approximation quality of the POD basis in each subdomain. When averaged across all subdomains, this is reported using the symbol, $\mathbf{E}_{\text{proj}}^{\text{avg}}$.

Lastly, we also use the maximum error in Section 4.2, defined as

$$\mathbf{E}_{\ell^2}^{\max} = \max_p \frac{\|\mathbf{u}(t_p)_{\text{MONO}} - \mathbf{u}(t_p)_{\text{SCHWARZ}}\|_2}{\|\mathbf{u}(t_p)_{\text{MONO}}\|_2}.$$

Quantities Reported in Tables. In addition to errors, we report the following quantities.

1. "Avg S.I." is the mean number of Schwarz iterations required to reach convergence across all time steps.
2. "Overlap" is the amount by which the 50×50 grid is shared by neighboring subdomains before being broken up into triangular elements. For example, in the Four Squares configuration of Figure 4.1c, if Overlap = 5, then the subdomain-specific grid for Ω_1 shares five rows of the original grid with Ω_3 and five columns with Ω_2. The overlap with Ω_4 is incidental.
3. "Time" is the average wall clock time taken to run the online model on identical hardware, measured in seconds. The number of runs averaged are mentioned in each table. This does not include any time spent setting up matrices, generating data, or similar offline tasks.
4. "r" is the integer dimension of the OpInf basis generated through POD. This is uniform across multiple subdomains.
5. "Data" is the number of training steps taken, with a value of 1 only including the initial condition. There are a maximum of 101 steps possible across the time interval $[0, 1]$, and any value of "Data" less than 101 results in prediction outside of training data. "Data" is uniform across all subdomains.

Lastly, computational speed-up is measured as the wall clock time to run the online portion of the OpInf-Schwarz coupled ROM models relative to that of the fully FE coupled Schwarz method and the monolithic FE solution over the same domain configuration and on the same hardware.

4.1. Static Boundary Conditions. For a first test, we consider the case where $u(\partial\Omega_T, t) = u(\partial\Omega_B, t) = 0$, $u(\partial\Omega_L, t) = 2$ and $u(\partial\Omega_R, t) = 5\ \forall t \in (0, 1]$. The discontinuity at some of the corners is resolved in favor of the top and bottom boundaries. We establish a baseline in Table 4.1 by coupling solely FE models in each of the 3 configurations of Figure 4.1. The wall time, in seconds, was averaged over 5 separate runs. Each subdomain has an overlap of 10 with its non-diagonal neighbors. The full monolithic simulation, averaged over 5 runs, takes 0.94 seconds to run the online portion.

$\mathbf{E}_{\ell^2}^{\text{avg}}$	1.27×10^{-14}
Avg S.I.	4.41
Time (s)	3.03

(a) Vertical, Fig 4.1a

$\mathbf{E}_{\ell^2}^{\text{avg}}$	1.57×10^{-14}
Avg S.I.	4.32
Time (s)	3.59

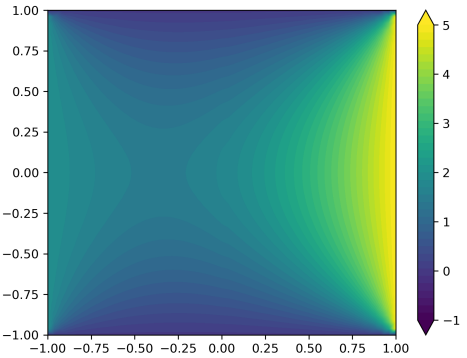
(b) Horizontal, Fig 4.1b

$\mathbf{E}_{\ell^2}^{\text{avg}}$	1.94×10^{-14}
Avg S.I.	5.9
Time (s)	7.05

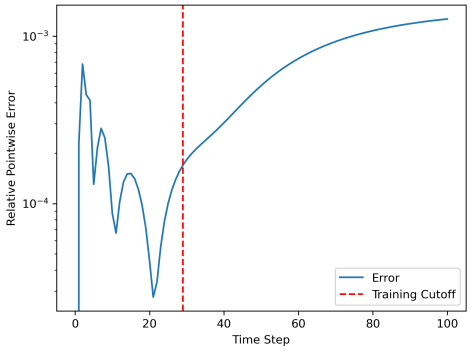
(c) Four Squares, Fig 4.1c

Table 4.1: Baseline for all-FE coupled models, Overlap = 10

Vertical Configuration. In Figure 4.2 and Table 4.2, we examine the behavior of the OpInf-Schwarz method with respect to r, the subdomain overlap, and data by coupling two OpInf models in the vertical orientation of Figure 4.1a. Wall clock time results, in seconds, are averaged over 2 individual runs.



(a) 2 coupled OpInf models in vertical configuration, $t = 0$



(b) Error between OpInf-Schwarz and monolithic FE simulation over time.

Fig. 4.2: OpInf-Schwarz vs. monolithic simulation. OpInf parameters: $r = 6$, Overlap = 10, Data = 30.

From a ROM perspective, Table 4.2c has several interesting results which deserve explanation. First, $\mathbf{E}_{\ell^2}^{\text{avg}}$ only decreases up to $r = 6$, then increases before stalling out for higher values of r. This is despite the fact that the projection error of the snapshots onto the basis drops consistently with the basis size. A basis size of $r = 3$ and $r = 4$ is necessary to capture 99.9% of the total energy of the snapshots Ω_1 and Ω_2 respectively.

Data	$\mathbf{E}_{\ell^2}^{\text{avg}}$	Avg S.I.	Time
10	1.16×10^{-01}	4.0	0.30
20	5.41×10^{-03}	4.57	0.30
30	6.02×10^{-04}	4.11	0.27
50	1.56×10^{-04}	4.52	0.31
80	9.16×10^{-05}	4.39	0.28
100	8.62×10^{-05}	4.49	0.31

(a) Fixed parameters: Overlap = 10 a $r = 6$.

Overlap	$\mathbf{E}_{\ell^2}^{\text{avg}}$	Avg S.I.	Time
1	8.68×10^{-03}	6.4	0.35
2	5.33×10^{-03}	6.34	0.34
5	2.36×10^{-03}	5.7	0.31
10	6.02×10^{-04}	4.11	0.22
20	5.96×10^{-04}	4.7	0.25
40	8.18×10^{-04}	4.26	0.24

(b) Fixed parameters: Data = 30, $r = 6$.

r	$\mathbf{E}_{\ell^2}^{\text{avg}}$	Avg S.I.	$\mathbf{E}_{\text{proj}}^{\text{avg}}$	Time
2	2.34×10^{-02}	8.04	8.98×10^{-02}	0.31
4	4.24×10^{-03}	5.0	1.02×10^{-02}	0.25
6	6.02×10^{-04}	4.11	6.14×10^{-04}	0.19
8	1.74×10^{-03}	5.0	5.85×10^{-05}	0.24
10	1.85×10^{-03}	5.0	6.02×10^{-06}	0.23
30	1.85×10^{-03}	5.0	3.54×10^{-10}	0.25

(c) Fixed parameters: Data = 30, Overlap = 10.

Table 4.2: Comparison of OpInf-OpInf coupled models in vertical orientation (Fig 4.1a).

In the row $r = 6$, $\mathbf{E}_{\ell^2}^{\text{avg}}$ and $\mathbf{E}_{\text{proj}}^{\text{avg}}$ have the same order of accuracy, while $\mathbf{E}_{\text{proj}}^{\text{avg}}$ is lower for all further values of r. This suggests that the accuracy of the OpInf-Schwarz method is limited by the basis quality up until $r = 6$, and afterwards is most limited by error in the inferred operators. This could be due to several reasons, such as data quality, our regularization strategy affecting accuracy, or perhaps the Schwarz coupling method interacting with the ROM in an unexpected way.

Another item of note in Table 4.2c is that the average run time decreases by about a factor of three as we increase r from 2 to 6, even though we are increasing the size of the linear systems we are solving. This is due to the coupling of poor quality solutions when $r = 2$, which requires additional Schwarz iterations to converge as shown by the mean Schwarz iterations. A similar phenomenon is observed in [3] when coupling subdomain-local intrusive ROMs via the Schwarz alternating method.

The remaining subtables of Table 4.2 reveal that OpInf-Schwarz performs as anticipated. In Table 4.2b, increasing the size of the overlap decreases the number of Schwarz iterations, and in Table 4.2a, increasing the training data decreases the approximation error. Since we are using the same data for both the POD basis and the OpInf training, it is fair to ask whether increasing the data allowance also allows POD to extract useful data for larger values of r—that is, further decreases to $\mathbf{E}_{\text{proj}}^{\text{avg}}$. In our testing, increasing training data had a minor effect on $\mathbf{E}_{\text{proj}}^{\text{avg}}$ compared to increasing r, though this would likely become more relevant in more challenging problems. The minimal error we were able to recover was $\mathbf{E}_{\ell^2}^{\text{avg}} = 2.5 \times 10^{-05}$ with the choices of $r = 10$, Overlap = 10, and 100 steps of training data.

While the times recorded are certainly not deterministic, a comparison between the FE-FE coupling of Table 4.1a and the minimum and maximum values of Table 4.2 suggest a speed-up of 8.6 to 15.2 times for fully OpInf-OpInf coupled models as compared to a FOM-FOM coupling on the same spatio-temporal grid, and 2.7 to 4.9 times speed-up for fully

OpInf-OpInf coupled models as compared to the monolithic FE simulation. Further speed-ups may be possible by introducing the "additive" Schwarz method [39], which computes subdomain solutions in parallel.

Comparison With Other Configurations. We now investigate whether interactions between the subdomain geometry and boundary conditions have an effect on the performance of OpInf-Schwarz. Because the non-zero boundary conditions in this simple case are on the left and right, a vertical overlap splitting the middle has a less complicated gradient than a horizontal overlap. In the scenario of $r = 6$, Overlap $= 10$, and 30 steps of training data, Table 4.3 shows that the error and Schwarz iterations are mildly worse for those configurations that impose a more complicated Schwarz boundary.

Model Configuration	$\mathbf{E}_{\ell^2}^{\text{avg}}$	Avg S.I.	Time (s)
Vertical	6.02×10^{-04}	4.11	0.26
Horizontal	8.91×10^{-04}	4.17	0.26
Four Squares	7.10×10^{-04}	5.81	0.61

Table 4.3: Vertical, horizontal, and square configurations for simple boundaries

All of the results in the simple case of Section 4.1 are in line with prior expectations for Schwarz performance in a ROM setting.

4.2. Time-Varying Boundary Conditions. We now consider a more challenging scenario for the 2D heat equation. Instead of solely constant boundary conditions, define

$$q(t, \mu) = 1 + 0.5 \sin(2\pi\mu t), \tag{4.1}$$

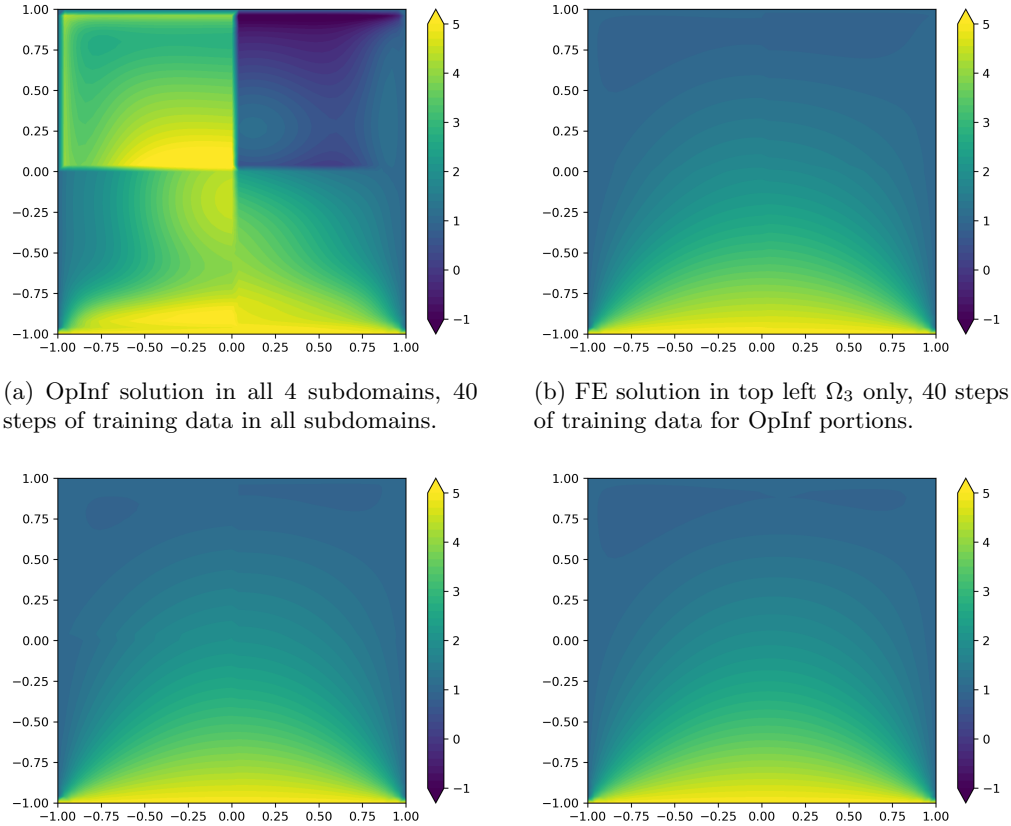
to be a time varying boundary condition parameterized by the frequency μ. We consider the monolithic heat equation of (2.1) on Ω specified by $u(\partial\Omega_L, t) = q(t, 2)$, $u(\partial\Omega_T, t) = q(t, 4)$, $u(\partial\Omega_B, t) = 5$, $u(\partial\Omega_R, t) = 1$ $\forall t \in (0, 1]$ and $u(\Omega, 0) = 0$.

The offset frequencies of the boundary conditions on the left and top boundaries create dynamics which evolve through a significant portion of the time domain, especially in the top left quadrant. These types of problems can be challenging for ROMs to resolve when given insufficient data, though they can usually be fixed by increasing the training data. This is one of the essential choices in ROMs – increasing the training range can often greatly improve accuracy, but data generation in realistic problems can be enormously expensive, especially if it must be generated from a monolithic simulation. Domain decomposition frameworks provide options beyond increasing the training range, and we now consider a comparison between purely OpInf-Schwarz with a FE-OpInf-Schwarz coupled problem.

Four Squares Configuration. Figure 4.3 shows two OpInf-Schwarz solutions at the final time step of $t = 1$ in the four square configuration of Figure 4.1c. For the OpInf portions, each solution has the same ROM basis dimension $r = 6$, 40 steps of training data (out of a possible 101), and a mean centering strategy performed on the snapshots. They additionally share the same boundary and initial conditions as described above. The only difference in their setup is that Figure 4.3a couples four OpInf models, while Figure 4.3b has replaced one of the OpInf models with a FE model in Ω_3, the top-left subdomain.

The FE assimilation in Figure 4.3b is not perfect if one looks closely along $x = 0$ or $y = 0$, but it shows a cohesive solution very similar to the monolithic solution in Figure 4.3d, while the solution in Figure 4.3a is clearly failing and is in fact about to diverge entirely. The All OpInf model with a higher data allowance in Figure 4.3c is comparable to Figure 4.3b.

One interesting feature of Figure 4.3a is that the Schwarz communication pattern is clearly evident. The solution is completely incorrect in the top left subdomain, Ω_3. The



(a) OpInf solution in all 4 subdomains, 40 steps of training data in all subdomains.

(b) FE solution in top left Ω_3 only, 40 steps of training data for OpInf portions.

(c) OpInf solution in all 4 subdomains, 65 steps of training data in all subdomains.

(d) Monolithic FOM (Finite Element Simulation).

Fig. 4.3: OpInf-Schwarz solutions and Monolithic FOM at $t = 1$.

error in Ω_3 subsequently propagates to neighboring Ω_1 and Ω_4, while the bottom right subdomain Ω_2 remains closest to an accurate solution.

From Table 4.4, it can be inferred that a purely OpInf coupled Schwarz model would require 65 total timesteps (an additional 25 steps of training data) to exceed the average relative accuracy of the OpInf-Schwarz with one FE subdomain and 40 steps of training data (as is displayed in Figure 4.3b). However, over two separate runs, the average time taken for the 4 coupled OpInf models in Table 4.4a was 0.73 seconds, while the average for the substitution of one FE model in Ω_3 in Table 4.4b is 3.15 seconds, a 4.3 times slowdown over a purely OpInf coupled model. The relative ease of model switching in the Schwarz framework lets one choose between data generation costs and monolithic FOM costs.

In comparison, a completely FE coupled model with the same settings has $\mathbf{E}_{\ell^2}^{\text{avg}} = \mathcal{O}(10^{-14})$, but, over three runs, takes an average of 7.9 seconds. A monolithic FE sim takes approximately 0.8 seconds. The "All OpInf" model is therefore about 9 times faster, and the 3 OpInf 1 FE model about 2.4 times faster than the fully FE coupled model. Comparing against the monolithic FOM, the "All OpInf" model is 1.1 times faster and the 3 OpInf 1 FE is 3.9 times slower.

Data	$\mathbf{E}_{\ell^2}^{\text{avg}}$	$\mathbf{E}_{\ell^2}^{\text{max}}$
30	1.97×10^{-01}	1.22×10^{00}
35	5.41×10^{-01}	4.07×10^{-01}
40	1.07×10^{-01}	6.40×10^{-01}
50	3.62×10^{-02}	2.12×10^{-01}
60	9.15×10^{-03}	3.13×10^{-02}
65	6.48×10^{-03}	1.91×10^{-02}

(a) All OpInf, 4 Coupled OpInf Models

Data	$\mathbf{E}_{\ell^2}^{\text{avg}}$	$\mathbf{E}_{\ell^2}^{\text{max}}$
30	3.33×10^{-02}	7.06×10^{-02}
35	9.47×10^{-03}	2.75×10^{-02}
40	7.14×10^{-03}	1.89×10^{-02}
50	4.76×10^{-03}	1.11×10^{-02}
60	1.86×10^{-03}	1.4×10^{-02}
65	1.77×10^{-03}	1.40×10^{-02}

(b) 3 OpInf 1 FE, FE model in top left subdomain Ω_3 only.

Table 4.4: OpInf-Schwarz errors for various training data ranges with $r = 6$ and Overlap = 10.

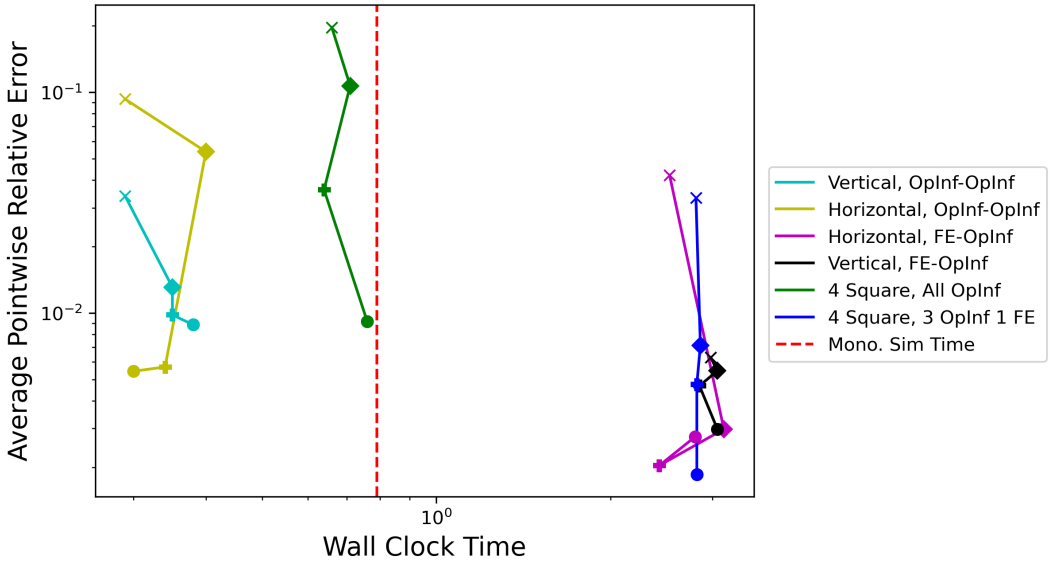


Fig. 4.4: Pareto plot for time-varying BCs, $\mathbf{E}_{\ell^2}^{\text{avg}}$ vs Time (s). Fixed parameters: $r = 6$, Overlap = 10. Plotted points are "Data" as it ranges over 30, 40, 50, and 60 marked by $\times$, $\blacklozenge$, $+$, and $\bullet$ respectively as in Table 4.4. Values in bottom left are preferred.

Efficiency Comparison With Other Configurations. We summarize our timing results for the time-varying boundary problem in the Pareto plot of Figure 4.4 using data from Table 4.4 for the "Four Squares" labelled results, and additional results for the "Vertical" labelled results produced in the vertical configuration from Figure 4.1a. In particular, "Vertical FE-OpInf" was produced with the FE model in the left subdomain, and "Horizontal FE-OpInf" was produced with the FE subdomain as the top subdomain, both advantageous DDs for resolving the more complicated dynamics in the top left portion of Ω. Fixed parameters are $r = 6$ and Overlap = 10. This chart demonstrates the significant impact that subdomain geometry can have when interacting with more complicated boundary issues or solution features. The vertical OpInf-OpInf is more accurate for smaller data allowances, but it is

overtaken in accuracy by the 3 OpInf 1 FE model for higher data allowances. All of the models with a FE component have similar time costs, but the horizontal and four square configurations make better use of additional data than the vertical configuration does.

While none of the models incorporating a FE subdomain exhibit lower runtimes than that of the monolithic FOM, we reiterate that it is possible to improve the performance of the Schwarz iteration process via the additive Schwarz formulation.

5. Conclusions and Future Work. The major impetus driving the development of the OpInf-Schwarz method is to achieve a minimally intrusive methodology for constructing ROMs that can interface modularly with existing high performance codes. A truly data-driven and modular domain decomposition-based ROM has the potential to greatly reduce computational expenses incurred by meshing and re-meshing complicated 3D objects, and therefore in performing long run-time and/or multi-query simulations.

The preliminary investigation presented herein has assessed various characteristics of the OpInf-Schwarz framework in the context of a 2D heat equation. Our results have shown that the OpInf-Schwarz method is capable of coupling OpInf and FOM subdomains to recover accurate solutions, and that a purely OpInf formulation can run faster than a monolithic simulation. We have also demonstrated that OpInf-Schwarz is very flexible, and can be tuned in several ways to the problem at hand (e.g., by choosing a different DD or ROM/FOM assignment to the subdomains).

At the same time, this work has uncovered several challenges that are worth investigating in the future. First, the calculated average relative pointwise error presented in Section 4.1 stagnates at around $\mathcal{O}(10^{-5})$ and does not decrease further despite a increase to the POD basis size and training data. While this level of error is acceptable for many applications, it would be ideal for the method to be truly convergent to the data source. While it seems likely that the issue lies with the error of the inferred operators from the regression problem (2.14) given that the projection errors reported in Table 4.2c continue to decrease, a comparison with a standard intrusive projection ROM would confirm whether the cause is error in the inferred operators or errors induced by the coupling process. It may be possible to reduce optimization errors by employing a technique called re-projection [26].

There are also potential improvements to the implementation of OpInf presented in this paper, as the stability of the learned operators is not enforced at the inference step. At present, this is resolved with a simple Tikhonov-based regularization strategy, but a more rigorous approach may be needed in more complicated problems, e.g., by using a more sophisticated regularization strategy such as those presented in [21, 31]. There are additional opportunities to improve the implementation of the Schwarz boundary conditions within our OpInf-Schwarz formulation in a way that both reduces computational cost and improves accuracy.

Currently, the size of the learned boundary operator $\hat{\mathbf{B}}$ scales with the size of the boundary in our formulation, but considering that even in this simple 2D case the number of boundary nodes is already 20 or more times larger than the ROM dimension, it is worth questioning whether all of this boundary information is being used in a meaningful way. Reducing the computational complexity of our method requires reducing the number of columns of $\hat{\mathbf{B}}$, which can be large for multi-dimensional problems with many boundary nodes. To mitigate the cost associated with evaluating $\hat{\mathbf{B}}\mathbf{g}$, it may be possible to perform a separate dimension reduction of this term. Finally, as noted earlier, the overall computational complexity of OpInf-Schwarz can be improved by introducing "additive" Schwarz, characterized by parallel subdomain solves with asynchronous boundary condition communication [39].

The last major point is on extending the Opinf-Schwarz method to more challenging

and realistic problems, as we expect the greatest utility and time savings of this model to be shown in more complicated situations. The heat equation is a standard prototype for model reduction methods, but extending the model to 3D, non-linear, convection dominated, and parametric problems [22, 38, 40] is a priority, as are implementing strategies to choose optimal domain decomposition and model assignment to enable on-the-fly ROM-FOM switching for maximal performance.

Acknowledgements. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Multifaceted Integrated Capability Centers (MMICCs) program, under Field Work Proposal 22-025291 (Multifaceted Mathematics for Predictive Digital Twins (M2dt)), Field Work Proposal 20-020467, and the Laboratory Directed Research and Development program at Sandia National Laboratories. The writing of this manuscript was funded in part by the fourth author's (Irina Tezaur's) Presidential Early Career Award for Scientists and Engineers (PECASE). This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan `https://www.energy.gov/downloads/doe-public-access-plan`.

The authors would like to thank Dr. Shane McQuarrie for numerous helpful technical discussions in the area of operator inference, and for providing support on the `OpInf` library.

REFERENCES

[1] M. S. Alnaes, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells, *Unified form language: A domain-specific language for weak formulations of partial differential equations*, ACM Transactions on Mathematical Software, 40 (2014).

[2] I. A. Baratta, J. P. Dean, J. S. Dokken, M. Habera, J. S. Hale, C. N. Richardson, M. E. Rognes, M. W. Scroggs, N. Sime, and G. N. Wells, *DOLFINx: the next generation FEniCS problem solving environment.* Pre-print, 2023. https://zenodo.org/records/10447666.

[3] J. Barnett, I. Tezaur, and A. Mota, *The Schwarz alternating method for the seamless coupling of nonlinear reduced order models and full order models.* ArXiv pre-print, 2022.

[4] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations*, International Journal for Numerical Methods in Engineering, 86 (2011), pp. 155–181.

[5] D. Cinquegrana, A. Viviani, and R. Donelli, *A hybrid method based on POD and domain decomposition to compute the 2-D aerodynamic flow field - incompressible validation*, 2011, pp. AIMETA 2011– XX Congresso dell'Associazione Italiana di Meccanica Teorica e Applicata, Bologna, ITA.

[6] A. Corigliano, M. Dossi, and S. Mariani, *Model order reduction and domain decomposition strategies for the solution of the dynamic elastic–plastic structural problem*, Computer Methods in Applied Mechanics and Engineering, 290 (2015), pp. 127–155.

[7] A. de Castro, P. Bochev, P. Kuberry, and I. Tezaur, *Explicit synchronous partitioned scheme for coupled reduced order models based on composite reduced bases*, Computer Methods in Applied Mechanics and Engineering, (2023), p. 116398.

[8] A. de Castro, P. Kuberry, I. Tezaur, and P. Bochev, *A novel partitioned approach for reduced order model – finite element model (ROM-FEM) and ROM-ROM coupling.* ArXiv pre-print.

[9] I.-G. Farcas, R. P. Gundevia, R. Munipalli, and K. E. Willcox, *Improving the accuracy and scalability of large-scale physics-based data-driven reduced modeling via domain decomposition.* ArXiv pre-print, 2023.

[10] R. Geelen, S. Wright, and K. Willcox, *Operator inference for non-intrusive model reduction with quadratic manifolds*, Computer Methods in Applied Mechanics and Engineering, 403 (2023), p. 115717.
[11] O. Ghattas and K. Willcox, *Learning physics-based models from data: perspectives from inverse problems and model reduction*, Acta Numerica, 30 (2021), p. 445–554.
[12] P. Holmes, J. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge University Press, 1996.
[13] A. Iollo, G. Sambataro, and T. Taddei, *A one-shot overlapping Schwarz method for component-based model reduction: application to nonlinear elasticity*, Computer Methods in Applied Mechanics and Engineering, 404 (2023), p. 115786.
[14] P. Kerfriden, O. Goury, T. Rabczuk, and S. Bordas, *A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics*, Computer Methods in Applied Mechanics and Engineering, 256 (2013), pp. 169–188.
[15] P. Kerfriden, J. C. Passieux, and S. P. A. Bordas, *Local/global model order reduction strategy for the simulation of quasi-brittle fracture*, International Journal for Numerical Methods in Engineering, 89 (2012), pp. 154–179.
[16] B. Kramer, B. Peherstorfer, and K. E. Willcox, *Learning nonlinear reduced models from data with operator inference*, Annual Review of Fluid Mechanics, 56 (2024), pp. 521–548.
[17] K. Kunisch and S. Volkwein, *Galerkin proper orthogonal decomposition methods for parabolic problems*, Numerische Mathematik, 90 (2001), pp. 117–148.
[18] K. Li, K. Tang, T. Wu, and Q. Liao, *D3m: A deep domain decomposition method for partial differential equations*, IEEE Access, 8 (2020), pp. 5283–5294.
[19] W. Li, X. Xiang, and Y. Xu, *Deep domain decomposition method: Elliptic problems*, Proceedings of Machine Learning Research, 107 (2020), pp. 269–286.
[20] I. Maier and B. Haasdonk, *A Dirichlet-Neumann reduced basis method for homogeneous domain decomposition problems*, Applied Numerical Mathematics, 78 (2014), pp. 31–48.
[21] S. A. McQuarrie, C. Huang, and K. E. Willcox, *Data-driven reduced-order models via regularised operator inference for a single-injector combustion process*, Journal of the Royal Society of New Zealand, 51 (2021), pp. 194–211.
[22] S. A. McQuarrie, P. Khodabakhshi, and K. E. Willcox, *Nonintrusive reduced-order models for parametric partial differential equations via data-driven operator inference*, SIAM Journal on Scientific Computing, 45 (2023), pp. A1917–A1946.
[23] A. Mota, D. Koliesnikova, I. Tezaur, and J. Hoy, *A fundamentally new coupled approach to contact mechanics via the Dirichlet-Neumann Schwarz alternating method*, 11 2023.
[24] A. Mota, I. Tezaur, and C. Alleman, *The Schwarz alternating method in solid mechanics*, Computer Methods in Applied Mechanics and Engineering, 319 (2017), pp. 19–51.
[25] A. Mota, I. Tezaur, and G. Phlipot, *The Schwarz alternating method for dynamic solid mechanics*, Int. J. Numer. Meth. Engng, (2022), pp. 1–36.
[26] B. Peherstorfer, *Sampling low-dimensional Markovian dynamics for preasymptotically recovering reduced models from data with operator inference*, SIAM Journal on Scientific Computing, 42 (2020), pp. A3489–A3515.
[27] B. Peherstorfer and K. Willcox, *Data-driven operator inference for nonintrusive projection-based model reduction*, Computer Methods in Applied Mechanics and Engineering, 306 (2016), pp. 196–215.
[28] I. Prusak, M. Nonino, D. Torlo, F. Ballarin, and G. Rozza, *An optimisation–based domain–decomposition reduced order model for the incompressible Navier-Stokes equations*, Computers & Mathematics with Applications, 151 (2023), pp. 172–189.
[29] A. Radermacher and S. Reese, *Model reduction in elastoplasticity: proper orthogonal decomposition combined with adaptive sub-structuring*, Computational Mechanics, 54 (2014), pp. 677–687.
[30] G. Rozza, *Fundamentals of reduced basis method for problems governed by parametrized PDEs and applications*, Springer, Vienna, 2014, pp. 153–227.
[31] N. Sawant, B. Kramer, and B. Peherstorfer, *Physics-informed regularization and structure preservation for learning stable reduced models from data with operator inference*, Computer Methods in Applied Mechanics and Engineering, 404 (2023), p. 115836.
[32] P. Schmid, *Dynamic mode decomposition of numerical and experimental data*, Journal of Fluid Mechanics, 656 (2010), p. 5–28.
[33] H. A. Schwarz, *Ueber einen Grenzübergang durch alternirendes Verfahren*, Zürcher u. Furrer, 1870.
[34] M. W. Scroggs, I. A. Baratta, C. N. Richardson, and G. N. Wells, *Basix: a runtime finite element basis evaluation library*, Journal of Open Source Software, 7 (2022), p. 3982.
[35] M. W. Scroggs, J. S. Dokken, C. N. Richardson, and G. N. Wells, *Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes*, ACM Transactions on Mathematical Software, 48 (2022), pp. 18:1–18:23.
[36] L. Sirovich, *Turbulence and the dynamics of coherent structures, part iii: dynamics and scaling*, Q.

Appl. Math., 45 (1987), pp. 583–590.

[37] W. Snyder, I. Tezaur, and C. Wentland, *Domain decomposition-based coupling of physics-informed neural networks via the Schwarz alternating method.* ArXiv pre-print, 2023.

[38] A. Vijaywargiya and A. Gruber, *Tensor parametric operator inference with Hamiltonian structure.* Computer Science Research Institute Summer Proceedings 2024, 2024.

[39] C. R. Wentland, F. Rizzi, J. Barnett, and I. Tezaur, *The role of interface boundary conditions and sampling strategies for Schwarz-based coupling of projection-based reduced order models.* ArXiv pre-print, 2024.

[40] S. Yildiz, P. Goyal, P. Benner, and B. Karasözen, *Learning reduced-order dynamics for parametrized shallow water equations from data*, International Journal for Numerical Methods in Fluids, 93 (2021), pp. 2803–2821.

OPERATOR INFERENCE BASED FLUX SURROGATE ALGORITHM FOR COUPLED TRANSMISSION PROBLEMS

RISHI PAWAR* AND PAVEL BOCHEV†

Abstract.
In this paper we develop and demonstrate a new surrogate-based partitioned scheme for a model advection-diffusion transmission problem. The work is motivated by the Implicit Value Recovery (IVR) partitioned method [4] in which the interface flux is approximated by the dual Schur complement of a discrete monolithic formulation of the coupled problem. Unless the discretized equations employ lumped mass matrices, forming and solving the Schur complement equation for the flux adds nontrivial computational cost at every time step. To reduce the computational cost we replace the Schur complement equation by an efficient flux surrogate. The latter is based on the Operator Inference algorithm (OpInf) which learns the dynamic behavior of the interface flux from a suitable training set during an offline training phase. The inferred operator is then used during the online stage to provide accurate flux approximations for each subdomain problem. Numerical examples illustrate the potential of the approach.

1. Introduction. Explicit partition methods for coupled problems enable independent solution of subdomain equations by codes that can be optimized for each specific physics model. Such schemes improve computational efficiency and enable reuse of existing simulation codes.

Loosely coupled partitioned schemes exchange interface states and/or fluxes to define suitable boundary conditions for the subdomain equations, thereby enabling their independent solution. Such schemes are minimally intrusive and computationally efficient. However, loosely coupled schemes are equivalent to a single step of an iterative solution procedure for the underlying coupled system and, as a result, they can experience instabilities and loss of accuracy. An alternative approach is to develop a partitioned scheme starting from a monolithic formulation of the coupled problem in which the continuity of the states is enforced by a Lagrange multiplier. The interface flux is then estimated by solving a dual Schur complement system. This approach leads to provably stable and accurate solutions [4] and is second order accurate provided the monolithic problem employs consistent mass matrices. However, in this case forming the Schur complement involves inversion of the subdomain mass matrices and leads to a small but dense linear system. As a result, such an approach can incur significant computational costs, especially in three dimensions.

The main goal of this work is to demonstrate that a data-driven surrogate flux approach can potentially combine the computational efficiency of loosely coupled schemes with the accuracy of methods based on monolithic formulations. Specifically, we replace the Schur complement step of the IVR algorithm by a flux surrogate based on the Operator Inference algorithm (OpInf). We choose OpInf because it provides an effective mechanism to learn the dynamics of the interface flux from time series data. In so doing we shift the main computational burden to an offline training phase where the OpInf flux surrogate is inferred from a suitable training set. During the online phase, the trained OpInf surrogate is used to compute interface fluxes at a fraction of the cost of solving the Schur complement.

We have organized this paper as follows. In Section 2 we review the Implicit Value Recovery Algorithm, which provides the basis for the OpInf surrogate-based partitioned scheme. Section 3 summarizes the basics of OpInf and formulates the OpInf flux surrogate. This section also describes the modified IVR scheme in which the Schur complement solve is replaced by the OpInf flux surrogate. Section 4 describes the offline training stage for the OpInf flux surrogate. In Section 5 we present numerical results with the OpInf surrogate-

*University of Arizona, rpawar@arizona.edu
†Sandia National Laboratories, pbboche@sandia.gov

based scheme for a model advection-diffusion transmission problem. Finally, in Section 6 we offer some conclusions and comment on the efficacy of the OpInf surrogate-based partitioned scheme.

2. Implicit Value Recovery. In this paper we restrict attention to a rectangular region $\Omega \in \mathbb{R}^2$ partitioned into two non-intersecting subdomains Ω_1 and Ω_2 by an interface γ; see Fig. 2.1. We orient the interface by a unit normal vector $\boldsymbol{n}_\gamma$ and set $\Gamma_i = \partial\Omega_i \setminus \gamma$.

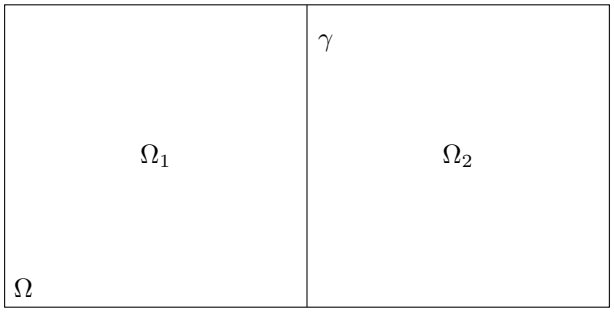


Fig. 2.1: The domain Ω is divided into subdomains Ω_1 and Ω_2.

We assume that each subdomain is endowed by an independently defined finite-element partition Ω_i^h with mesh parameter h_i, vertices $x_{i,r}$, and elements $K_{i,s}$. The subdomain partitions induce finite element partitions γ_1^h and γ_2^h of the interface that are spatially coincident but not required to have matching nodes.

The model advection-diffusion transmission problem is defined by a pair of governing equations on each subdomain

$$\begin{cases} \dot{\varphi}_i - \nabla \cdot F_i(\varphi_i) = f_i & \text{in } \Omega_i \times [0,T] \\ \varphi_i = g_i & \text{in } \Gamma_i \times [0,T] \end{cases} \tag{2.1}$$

augmented with initial conditions,

$$\varphi_i(\boldsymbol{x}, 0) = \varphi_{i,0}(\boldsymbol{x}) \text{ in } \Omega_i \quad i = 1, 2; \tag{2.2}$$

and interface conditions,

$$\begin{aligned} 0 &= \varphi_1(\boldsymbol{x}, t) - \varphi_2(\boldsymbol{x}, t), \\ 0 &= F_1(\boldsymbol{x}, t) \cdot \boldsymbol{n}_\gamma - F_2(\boldsymbol{x}, t) \cdot \boldsymbol{n}_\gamma \end{aligned} \tag{2.3}$$

on $\gamma \times [0, T]$. The total flux F_i is assumed to have the form

$$F_i(\varphi_i) = \epsilon_i \nabla(\varphi_i) - \boldsymbol{u}\varphi_i \tag{2.4}$$

where $\epsilon_i > 0$ is a diffusion coefficient in Ω_i and $\boldsymbol{u}$ is a given velocity field that can be zero.

Partitioned solution of (2.1), (2.2), and (2.3) is based on the observation that the transmission problem can be written in an equivalent form as

$$\begin{cases} \dot{\varphi}_i - \nabla \cdot F_i(\varphi_i) & = f_i \quad in \quad \Omega_i \times [0,T] \\ \varphi_i & = g_i \quad in \quad \Gamma_i \times [0,T] \qquad i = 1, 2\,, \\ F_i(\varphi_i) \cdot \boldsymbol{n}_i & = (-1)^i \lambda \quad on \quad \gamma \times [0,T] \end{cases} \tag{2.5}$$

where λ denotes the interface flux. Formally, each equation in (2.5) is a mixed boundary value problem on its respective domain with Neumann data provided by the interface flux. These equations remain coupled because λ is one of the unknowns in this system. However, if one can obtain an approximation of the interface flux these equations can be solved independently, thereby decoupling the problem.

The IVR scheme [4] computes an approximation of the interface flux by forming and solving the Schur complement of a discrete monolithic system obtained by enforcing the alternative coupling condition

$$\dot{\varphi_1}(\boldsymbol{x},t) - \dot{\varphi_2}(\boldsymbol{x},t) = 0 \tag{2.6}$$

by a Lagrange multiplier. Assuming that that the initial data is continuous along the interface, i.e. $\varphi_0(\boldsymbol{x}^-) = \varphi_0(\boldsymbol{x}^+)\ \forall \boldsymbol{x} \in \gamma$ condition (2.6) is equivalent to the original one.

The resulting semi-discrete monolithic system is equivalent to the following system of differential algebraic equations (DAEs); see [4]:

$$\begin{bmatrix} M_1 & 0 & G_1^T \\ 0 & M_2 & -G_2^T \\ G_1 & -G_2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\varphi}}_1 \\ \dot{\boldsymbol{\varphi}}_2 \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1(\boldsymbol{\varphi}_1) \\ \boldsymbol{f}_2(\boldsymbol{\varphi}_2) \\ 0 \end{bmatrix}, \tag{2.7}$$

for finite element coefficient vectors $\boldsymbol{\varphi_1}, \boldsymbol{\varphi_2}$, and $\boldsymbol{\lambda}$. Here, M_1, M_2 are mass matrices, and G_1, G_2 are matrices of inner products between the finite element bases functions and the test functions along boundary γ; full definitions for these matrices can be found in [4].

The IVR scheme then proceeds to eliminate the interface flux from (2.7) by solving the system:

$$S\boldsymbol{\lambda} = G_1 M_1^{-1} \boldsymbol{f}_1(\boldsymbol{\varphi}_1) + G_2 M_2^{-1} \boldsymbol{f}_2(\boldsymbol{\varphi}_2), \tag{2.8}$$

where

$$S = G_1 M_1^{-1} G_1^T + G_2 M_2^{-1} G_2^T \tag{2.9}$$

is the Schur complement of (2.7). It is straightforward to see that application of explicit time integration to the resulting coupled system of ODEs decouples this system and enables the independent solution of each subdomain ODE problem for $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$. Algorithm 1 summarizes the IVR scheme.

Algorithm 1 Implicit Value Recovery (IVR) Algorithm Summary

Require: Evaluation of $G_1 M_1^{-1} G_1^T + G_2 M_2^{-1} G_2^T$, $\boldsymbol{\varphi}_1^1$, and $\boldsymbol{\varphi}_2^1$

while $k < N_{max}$ **do**

1. **Compute** $\boldsymbol{f}_i(\boldsymbol{\varphi}_i^k)$ for $i = 1, 2$,
2. **Solve:**

$$\left(G_1 M_1^{-1} G_1^T + G_2 M_2^{-1} G_2^T\right) \boldsymbol{\lambda}_k = G_1 M_1^{-1} \boldsymbol{f}_1(\boldsymbol{\varphi}_1^k) + G_2 M_2^{-1} \boldsymbol{f}_2(\boldsymbol{\varphi}_2^k) \tag{2.10}$$

for $\boldsymbol{\lambda}_k$.

3. **Solve**

$$M_i D^{k+1} \boldsymbol{\varphi}_i = \boldsymbol{f}_i(\boldsymbol{\varphi}_i^k) + (-1)^i G_i^T \boldsymbol{\lambda}_k \quad i = 1, 2\,, \tag{2.11}$$

for $\boldsymbol{\varphi}_i^{k+1}$, where D^{k+1} denotes an explicit approximation of the time derivative at t_{k+1}.

end while

Although the IVR scheme can be implemented using lumped mass matrices, in order to achieve optimal accuracy one has to employ consistent mass matrices. In this case, forming the Schur complement in Step 2 involves inversion of the sparse matrices M_i, which results in a dense Schur complement matrix. In principle, one could precompute and store the Schur complement in factored form, however, the storage burden may be unacceptable in three dimensions and even in some two-dimensional configurations employing very fine meshes.

In this paper we investigate a data-driven alternative in which the Schur complement system for the interface flux is replaced by a data-driven surrogate for the dynamics of this variable. To that end we shall use Operator Inference which can be used to infer systems of ODEs comprised of polynomial terms. The operators are inferred offline from a suitable time series data of the solutions of the transmission problem. Once the OpInf surrogate is trained we propose to use it in Step 2 of the IVR algorithm as a cost and storage effective replacement for the Schur complement.

3. Operator Inference Surrogate. Operator Inference attempts to use trajectory data sampled from a system of ODEs, parametrized by $\boldsymbol{\mu} \in \mathbb{R}^d$, to infer a lower dimensional representation of the system by fitting operators of polynomial terms in the least squares sense. Once these inferred operators are learned, one can simulate the system in the reduced space for one time step, then project the reduced state into the full state to forecast state data. The lower dimensional representation of the system will come from the SVD of snapshot matrix $\boldsymbol{X}$, i.e. $\boldsymbol{X} = U\Sigma V^T$. We extract matrices U and Σ, where U will be used to generate our projection matrix U_r from dimension n into lower dimension r, and Σ will be used to determine the rank r based off a desired energy percentage. By taking the SVD of these simulated trajectories $\boldsymbol{X}$ in phase space, we attempt to recover a statistical lower dimensional representation of the manifold that our data is generated from.

Consider a dynamical system with the following form:

$$\dot{\boldsymbol{x}} = A(\boldsymbol{\mu})\boldsymbol{x} + H(\boldsymbol{\mu})\boldsymbol{x}^2 + \boldsymbol{c}(\boldsymbol{\mu}), \tag{3.1}$$

where $A \in \mathbb{R}^{n\times n}$, $H \in \mathbb{R}^{n\times s}$, with $s = \binom{n}{2} + n = \frac{n(n+1)}{2}$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{x} \in \mathbb{R}^n$, and $\boldsymbol{x}^2$ refers to quadratic terms generated by $\boldsymbol{x}$, e.g. if $\boldsymbol{x} = (x_1, x_2)$, then $\boldsymbol{x}^2 = (x_1^2, x_1x_2, x_2^2)$. The term $\boldsymbol{x}^2$ can be obtained from Kronecker tensor product $\boldsymbol{x} \otimes \boldsymbol{x}$. Operator H is then written as $H = QI_1$, where I_1 is a reduced row identity matrix that removes redundant elements from $\boldsymbol{x} \otimes \boldsymbol{x}$, since some elements will be repeated in the Kronecker tensor product, e.g. $x_1x_2 = x_2x_1$. Matrix Q represents the linear operator that acts on the quadratic terms, now uniquely represented.

A uniform time discretization of time domain $[0, T]$ into K time steps of size δ_K, $\delta_K = \frac{T}{K}$, is chosen. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ be discrete states of the system of ODEs at times $t_1, \ldots, t_K$ computed with some time stepping scheme.

Including, $\boldsymbol{x}_0$ at t_0, assemble the $K + 1$ snapshot vectors into the following matrix $\boldsymbol{X}$:

$$\boldsymbol{X} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_K] \in \mathbb{R}^{n\times(K+1)}. \tag{3.2}$$

Take the SVD of matrix $\boldsymbol{X}$

$$\boldsymbol{X} = U\Sigma V^T, \tag{3.3}$$

and collect matrices U and Σ.

Extract the first r columns of matrix U for desired rank r such that,

$$U_r = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{n \times r}. \tag{3.4}$$

The rank r is chosen as some function of the singular values of matrix Σ.

The data-driven reduced order operators will approximate Galerkin projections, i.e. for Galerkin projections:

- $\tilde{A}(\boldsymbol{\mu}) = U_r^T A(\boldsymbol{\mu}) U_r \in \mathbb{R}^{r \times r}$
- $\tilde{H}(\boldsymbol{\mu}) = U_r^T H(\boldsymbol{\mu})(U_r \otimes U_r) \in \mathbb{R}^{r \times r^2}$
- $\tilde{c}(\boldsymbol{\mu}) = U_r^T \vec{c}(\boldsymbol{\mu}) \in \mathbb{R}^r$,

where for $\tilde{H}(\boldsymbol{\mu})$, a vector $\boldsymbol{x} \in \mathbb{R}^n$ is projected into $\boldsymbol{x}_r \in \mathbb{R}^r$ and its Kronecker product is taken with itself, i.e. $\boldsymbol{x}_r \otimes \boldsymbol{x}_r \in \mathbb{R}^{r^2}$, we have a reduced order system with the following dynamics:

$$\begin{aligned} \dot{\boldsymbol{x}}_r(t, \boldsymbol{\mu}) &= \tilde{A}(\boldsymbol{\mu}) \boldsymbol{x}_r(t; \boldsymbol{\mu}) + \tilde{H}(\boldsymbol{\mu})(\boldsymbol{x}_r \otimes \boldsymbol{x}_r) + \tilde{c}(\boldsymbol{\mu}) \\ \boldsymbol{x}_{r,0}(\boldsymbol{\mu}) &= U_r^T \boldsymbol{x}_0(\boldsymbol{\mu}) \in \mathbb{R}^r \end{aligned} \tag{3.5}$$

The OpInf algorithm will attempt to approximate the Galerkin projections $\tilde{A}, \tilde{H}, \tilde{c}$ with learned operators $\hat{A}, \hat{H}, \hat{c}$ respectively.

The projected vectors are calculated in the following manner

$$\hat{\boldsymbol{x}}_j = U_r^T \boldsymbol{x}_j \quad j = 0, \ldots, K \tag{3.6}$$

The following reduced order data matrices are calculated:

$$\begin{aligned} \hat{\boldsymbol{X}} &= [\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}_K}]^T \in \mathbb{R}^{K \times r} \\ \hat{\boldsymbol{X}}^2 &= [I_2(\hat{\boldsymbol{x}}_1 \otimes \hat{\boldsymbol{x}}_1), \ldots, I_2(\hat{\boldsymbol{x}}_K \otimes \hat{\boldsymbol{x}}_K)]^T \in \mathbb{R}^{K \times r'}, \end{aligned} \tag{3.7}$$

where $r' = \binom{r}{2} + r = r(r+1)/2$ and I_2 is another reduced identity matrix that removes repeated elements from the Kronecker tensor product.

For $j = 1, \ldots, K$, let $\dot{\hat{\boldsymbol{x}}}_j \in \mathbb{R}^n$ be an approximation to the derivative $\frac{d}{dt}\hat{\boldsymbol{x}}(t_j)$ of the projected state $\hat{\boldsymbol{x}}(t_j)$ at time t_j that converges to $\frac{d}{dt}\hat{\boldsymbol{x}}(t_j)$ in the L^2 norm as $\delta t \to 0$. One can use any discrete derivative approximation scheme to calculate the corresponding derivatives at each corresponding $\hat{\boldsymbol{x}}_j$, e.g. forward/backward Euler methods, RK-4 methods, etc. In this report, a forward Euler difference scheme is used to calculate the derivatives at corresponding points $\hat{\boldsymbol{x}}_j$.

The matrices $\hat{A}, \hat{H}, \hat{c}$ are the solution to the optimization problem

$$\min_{\hat{A}, \hat{H}, \hat{c}} \sum_{j=1}^{K} \left\| \dot{\hat{\boldsymbol{x}}}_j - \hat{A}\hat{\boldsymbol{x}}_j - \hat{H}(\hat{\boldsymbol{x}}_j \otimes \hat{\boldsymbol{x}}_j) - \hat{c} \right\|_2^2, \tag{3.8}$$

which implies that

$$\dot{\hat{\boldsymbol{x}}}_j = \hat{A}\hat{\boldsymbol{x}}_j + \hat{H}(\hat{\boldsymbol{x}}_j \otimes \hat{\boldsymbol{x}}_j) + \hat{c} \quad j = 1, \ldots, K. \tag{3.9}$$

We now construct data matrix D defined in the following manner

$$D = [\hat{\boldsymbol{X}}, \hat{\boldsymbol{X}}^2, \mathbf{1}] \in \mathbb{R}^{K\times(r+r'+1)}, \tag{3.10}$$

where $\mathbf{1}$ is a vector of ones. We also define derivative matrix R as

$$R = [\dot{\hat{\boldsymbol{x}}}_1, \ldots, \dot{\hat{\boldsymbol{x}}}_K]^T \in \mathbb{R}^{K\times r}. \tag{3.11}$$

The optimization problem of interest becomes:

$$\min_{\hat{A},\hat{H},\hat{c}} \left\| R - \hat{\boldsymbol{X}}\hat{A}^T - \hat{\boldsymbol{X}}^2\hat{H}^T - \hat{c} \right\|_F^2, \tag{3.12}$$

with respect to the Frobenius norm. Problem (3.12) can be written equivalently as

$$\min_{O\in\mathbb{R}^{r\times(r+r'+1)}} \left\| DO^T - R \right\|_F^2, \tag{3.13}$$

where operator matrix $O = [\hat{A}, \hat{H}, \hat{c}] \in \mathbb{R}^{r\times(r+r'+1)}$ is of interest. Iteratively, line (3.13) can be expressed as

$$\min_{\boldsymbol{o}_i\in\mathbb{R}^{r+r'+1}} \left\| D\boldsymbol{o}_i - \dot{\hat{\boldsymbol{x}}}_i \right\|_2^2 \quad i = 1, \ldots, r \tag{3.14}$$

which yield n least squares problems that solve each row of O^T.

3.0.1. Convergence Guarantees. If the time stepping scheme used in the full model has the following properties:

1. is convergent as $\delta t \to 0$, i.e. $\dot{\hat{\boldsymbol{x}}}_j$ converges in the L^2 norm to $\frac{d}{dt}\hat{\boldsymbol{x}}(t_j)$ as $\delta t \to 0$ for $j = 1, \ldots, K$,
2. has data matrices $\tilde{D}$ and D that have full column rank $r + r' + 1$ for all $\delta t > 0$ and for all $r \leq n$,

then there exists for $\epsilon > 0$ and $r \leq n$, a time step δt such that the Frobenius norm of the difference of the inferred $\hat{A}$ and the intrusive reduced operator $\tilde{A}$ is below ϵ, i.e.

$$\|\hat{A} - \tilde{A}\|_F < \epsilon.$$

Similar results hold for $\hat{H}$ and $\hat{c}$ [3].

3.1. OpInf Surrogate-Based Partition Scheme. To develop the OpInf flux surrogate we consider inputs defined by the concatenation of a patch of states and the flux:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\lambda}_{k-1} \\ \boldsymbol{u}_{1,k} \\ \boldsymbol{u}_{2,k} \end{bmatrix}, \tag{3.15}$$

where interface patches $\boldsymbol{u}_{1,k}$ and $\boldsymbol{u}_{2,k}$ refer to patches sampled left and right of the interface respectively, Figure 3.1 shows the sampling scheme. This patch scheme is used because the interface flux $\boldsymbol{\lambda}$ should maintain solution continuity across $\boldsymbol{\gamma}$, i.e. state data is needed from both subdomains to enforce continuity, and the solution dynamics should not be strongly

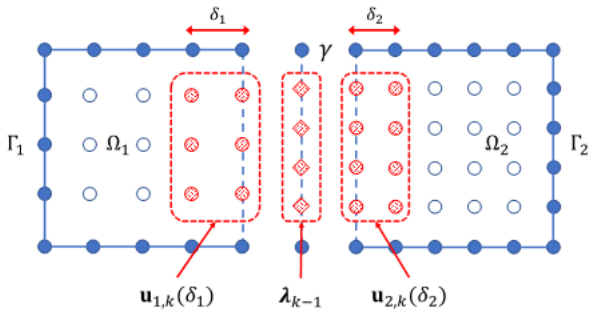


Fig. 3.1: Interface patches of width 2 for each side, image from [1].

influenced by solution values away from the interface, explaining why patches around the interface are used [1].

We provide details about the training of the OpInf surrogate in the next section. Assuming that such a surrogate is available, we use it to compute an approximation of the interface flux at t_k by forming the state $\boldsymbol{x}_k$ and solving numerically the OpInf surrogate for one time step. The flux component $\boldsymbol{\lambda}_k$ is then extracted from the integrated state and used to decouple the equations for $\boldsymbol{\varphi}_i$. Algorithm 2 summarizes the resulting OpInf surrogate-based partitioned scheme.

Algorithm 2 OpInf Surrogate-based partitioned scheme

Require: $\hat{A}, U_r$, $\boldsymbol{x}_0$, Δt

while $k < N_{max}$ **do**

1. **Compute** $\boldsymbol{f}_i(\boldsymbol{\varphi}_i^k)$ for $i = 1, 2$,
2. **Assemble** $\boldsymbol{x}_k$ from $\boldsymbol{\lambda}_{k-1}$, $\boldsymbol{u}_{1,k}$, and $\boldsymbol{u}_{2,k}$.
3. **Project into** $\mathbb{R}^r$

$$\hat{\boldsymbol{x}}_k = U_r^T \boldsymbol{x}_k \tag{3.16}$$

4. **Calculate**

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_k + \Delta t \hat{A} \hat{\boldsymbol{x}}_k \tag{3.17}$$

5. **Project into** $\mathbb{R}^n$

$$\tilde{\boldsymbol{x}}_{k+1} = U_r \hat{\boldsymbol{x}}_{k+1} \tag{3.18}$$

6. **Extract** $\boldsymbol{\lambda}_k$ from $\tilde{\boldsymbol{x}}_{k+1}$.
7. **Solve**

$$M_i D^{k+1} \boldsymbol{\varphi}_i = \boldsymbol{f}_i(\boldsymbol{\varphi}_i^k) + (-1)^i G_i^T \boldsymbol{\lambda}_k \,, i = 1, 2 \tag{3.19}$$

for $\boldsymbol{\varphi}_i^{k+1}$.

end while

3.2. FLOP Analysis. To quantify the computational benefit of the OpInf approach, FLOP counts are taken for the implementation of the OpInf flux surrogate into the IVR

scheme. We recall that $U_r \in \mathbb{R}^{n\times r}, \boldsymbol{x}_k \in \mathbb{R}^n, \hat{\boldsymbol{x}}_k \in \mathbb{R}^r, \hat{A} \in \mathbb{R}^{r\times r}$. The approach written in Algorithm 2 consists of the following calculations:

$$\begin{aligned} \hat{\boldsymbol{x}}_k &= U_r^T \boldsymbol{x}_k, \\ \hat{\boldsymbol{x}}_{k+1} &= \hat{\boldsymbol{x}}_k + \Delta t \hat{A} \hat{\boldsymbol{x}}_k, \\ \tilde{\boldsymbol{x}}_{k+1} &= U_r \hat{\boldsymbol{x}}_{k+1}, \end{aligned} \tag{3.20}$$

which yields a FLOP count of $4nr + 2r^2 + 2r$. This approach implies usage of a forward Euler scheme, but can be modified depending on the desired integrator. Combining several steps together, one can write the following equivalent formulation:

$$\begin{aligned} \tilde{\boldsymbol{x}}_{k+1} &= U_r U_r^T \boldsymbol{x}_k + \Delta t \overset{\circ}{A} \boldsymbol{x}_k \\ &= P \boldsymbol{x}_k + \Delta t \overset{\circ}{A} \boldsymbol{x}_k \\ &= (P + \Delta t \overset{\circ}{A}) \boldsymbol{x}_k \\ &= M \boldsymbol{x}_k, \end{aligned} \tag{3.21}$$

where $P = U_r U_r^T \in \mathbb{R}^{n\times n}$ and $\overset{\circ}{A} = U_r \hat{A} U_r^T \in \mathbb{R}^{n\times n}$, and $M \in \mathbb{R}^{n\times n}$. For maximum efficiency, $P, \overset{\circ}{A}$, and M are precomputed before running the OpInf modified IVR algorithm. The FLOP count for the last line of (3.21) is $2n^2$. Since we are interested in solely the flux vector $\boldsymbol{\lambda}$ from $\tilde{\boldsymbol{x}}_{k+1}$, we can focus on the portion of M that outputs $\boldsymbol{\lambda}$ and evaluate

$$\boldsymbol{\lambda}_k = M_f \boldsymbol{x}_k, \tag{3.22}$$

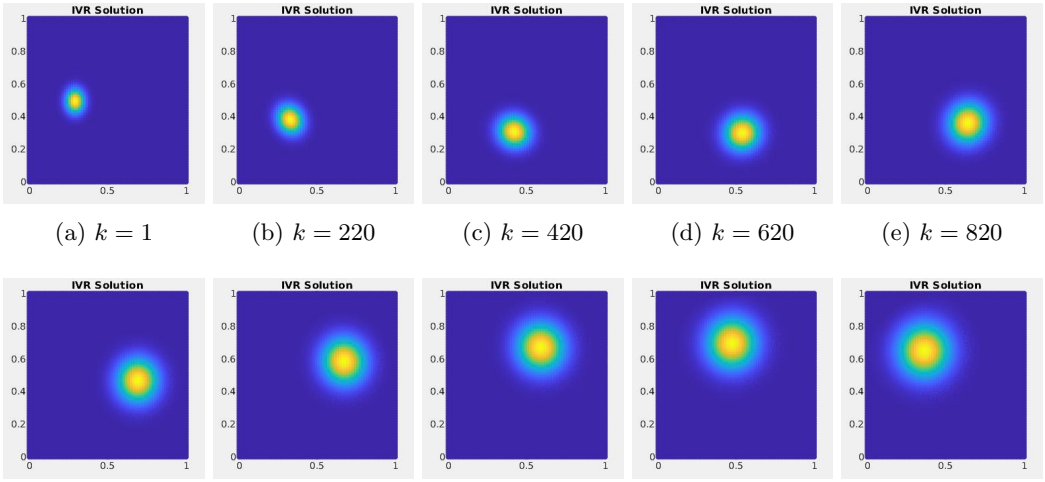
where $M_f = M(1 : f_d, :)$, where f_d is the dimension of the interface flux $\boldsymbol{\lambda}$, and $M_f \in \mathbb{R}^{f_d \times n}$ for a FLOP count of $2n \times f_d$, with $f_d \leq r \leq n$. (Assuming that the dimension of our flux interface f_d is sufficient to accurately model the flux crossing across the interface, we expect the rank r used in our scheme to be at least f_d. If $r < f_d$, then there exists some degeneracies/redundancies in the data of the flux interface). Approach (3.22) is the cheapest with regards to the number of FLOPs required for each iteration.

4. Training the Model. We now describe training of the inferred operators for the OpInf flux surrogate.

4.1. Training Data. To generate training data for the OpInf flux surrogate we use the IVR scheme to solve the model transmission problem for an initial condition given by a Gaussian distribution centered at a point $(x, y) \in \Omega$ and a rotating velocity field; see Figure 4.1 for a sample solution trajectory. To compute the solution we use a uniform mesh with a total of $2145 = 65 \times 33$ degrees of freedom in each subdomain and 63 degrees of freedom for the interior interface flux.

The training set comprises time series data for solution trajectories corresponding to 9 Gaussian initial conditions centered along the line $y = 0.5$. The x-coordinates of the initial conditions are given by $x = \{0.05, \ldots, 0.45\}$ with increments of 0.05. The transmission problem is simulated for one full revolution using k=1866 time steps. Figure 4.1 shows a few snapshots of a representative solution trajectory.

This training set was chosen because the spacing between the centers is on the same order as the mesh size of the FEM partition. This spacing combined with the choices of Gaussian sizes allows the interface γ to experience crossings everywhere when the entire



(a) $k = 1$ (b) $k = 220$ (c) $k = 420$ (d) $k = 620$ (e) $k = 820$

(f) $k = 1020$ (g) $k = 1220$ (h) $k = 1420$ (i) $k = 1620$ (j) $k = 1820$

Fig. 4.1: A representative solution trajectory corresponding to a Gaussian initial condition centered at $(0.3, 0.5)$.

training set is used; consequently, the training set contains information about the dynamical response of the interface flux along the entire interface [1].

The raw training data consists of solution states $\boldsymbol{\varphi}_i^k \in \mathbb{R}^{2145}$ and interface fluxes $\boldsymbol{\lambda}_i^{k-1} \in \mathbb{R}^{63}$ for $k = \{1, \ldots, 1866\}$.

To define the training data of the OpInf surrogate, data patches $\boldsymbol{u}_{i,k}$ of equal depth, i.e. $\delta_1 = \delta_2$, and $\boldsymbol{\lambda}_{k-1}$ are stacked on top of each other:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\lambda}_{k-1} \\ \boldsymbol{u}_{1,k} \\ \boldsymbol{u}_{2,k} \end{bmatrix} \in \mathbb{R}^{63+65\times 2\times(\delta_1+1)}, \tag{4.1}$$

where δ_1 can vary between 1 to 32. As we are interested in computational speed, we desire to use small $\delta_1 < 32$ while providing sufficient accuracy for various testing conditions. In this report, we found $\delta_1 = 5$ to be satisfactory for our tests.

For each of our 9 initial conditions, we assemble a snapshot matrix $\boldsymbol{X}_i$ for $i = 1, \ldots, 9$ whose columns are the above $\boldsymbol{x}_k$ for $k = 0, \ldots, 1866$. We concatenate these 9 trajectories into matrix $\boldsymbol{X}$ in the following manner

$$\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_9]. \tag{4.2}$$

Upon forming $\boldsymbol{X}$, we take the SVD of this matrix and obtain its constituent factors:

$$\boldsymbol{X} = U\Sigma V^T. \tag{4.3}$$

We obtain the singular values σ_i from Σ and use energy tolerance for rank r selection. If we wish p percentage of the energy of our system, we select the first r such that

$$p \leq \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_i \sigma_i^2}. \tag{4.4}$$

Once rank r is chosen, we extract the first r columns of U and form U_r. For our report, the energy percentage used was 99.99999999999%. Despite the high percentage of energy p, significant dimension reduction is still observed when this amount is used. In our testing, reduction from 843 dimensions, from padded sampling, to 132 dimensions was observed; recall that the original dimension of our system was $4353 = 2 \times 2145 + 63$. Since we are interested in an accurate surrogate, it is beneficial to use as much energy as possible without trading off low dimensionality. From numerical testing, this is the largest p that can be used before matrix U_r starts to possess linearly dependent columns. One could use a lower percentage of energy p, but the trade off would be a larger rank for r in matrix U_r.

Since the advection-diffusion problem is linear, we assume that the ODEs of our dynamics have the following form:

$$\dot{\boldsymbol{x}}_k = A\boldsymbol{x}_k, \tag{4.5}$$

where $A \in \mathbb{R}^{n\times n}$ and $\boldsymbol{x}_k \in \mathbb{R}^n$.

Therefore, we will be interested in obtaining reduced order operator $\hat{A} \in \mathbb{R}^{r\times r}$ that satisfies the following reduced order equation:

$$\dot{\hat{\boldsymbol{x}}}_k = \hat{A}\hat{\boldsymbol{x}}_k, \tag{4.6}$$

where $\hat{\boldsymbol{x}}_k = U_r^T \boldsymbol{x}_k$.

First, we form $\hat{\boldsymbol{X}} = U_r^T \boldsymbol{X}$, where the columns of $\hat{\boldsymbol{X}}$ are $\hat{\boldsymbol{x}}_k$. Next we construct a matrix R of reduced order derivative approximations.

Using the framework of (3.10), (3.11), and (3.12), we formulate our problem of interest with $D = \hat{\boldsymbol{X}}$ to solve the problem:

$$\min_{\hat{A}} \left\| R - \hat{\boldsymbol{X}}\hat{A}^T \right\|_F^2. \tag{4.7}$$

We can solve (4.7) for $\hat{A}$ using a QR decomposition on matrix $\hat{\boldsymbol{X}}$ in the following manner:

$$\begin{aligned} \hat{\boldsymbol{X}}\hat{A}^T &= R \\ QR_2\hat{A}^T &= R \\ R_2\hat{A}^T &= Q^T R \\ \hat{A}^T &= R_2 \backslash (Q^T R), \end{aligned} \tag{4.8}$$

where the last line uses Matlab's \ command. Using MATLAB's function allows one to compute operator $\hat{A}$ very quickly in comparison to the iterative approach written in (3.14). This approach can be applied to general operator O to calculate the matrices for polynomial terms.

5. Results. In this section we provide numerical results illustrating the OpInf flux surrogate-based partitioned scheme. The experiments were performed using the 9 equally distributed Gaussians with equal width in both the x and y directions as training data. For reference, the relative error at timestep t_k was calculated using the following formula:

$$Error(t_k) = \frac{\|\boldsymbol{\lambda}_{true}(t_k) - \boldsymbol{\lambda}_{approx}(t_k)\|_2}{\|\boldsymbol{\lambda}_{true}(t_k)\|_2} \quad \text{for } t_k = 0, 1, \ldots, 1865, \tag{5.1}$$

where $\|\cdot\|_2$ refers to the Euclidean 2-norm.

5.1. Reproductive Tests of the OpInf Surrogate. As an initial test, we attempt to reproduce 1 training trajectory associated with the initial condition position $(0.25, 0.5)$. The error plot for the approximated $\boldsymbol{\lambda}$ and IVR generated $\boldsymbol{\lambda}$ values is shown in Figure 5.1.

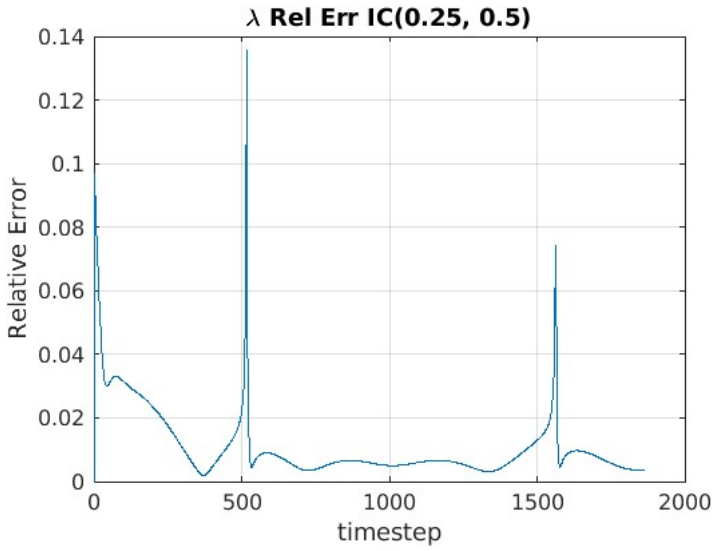
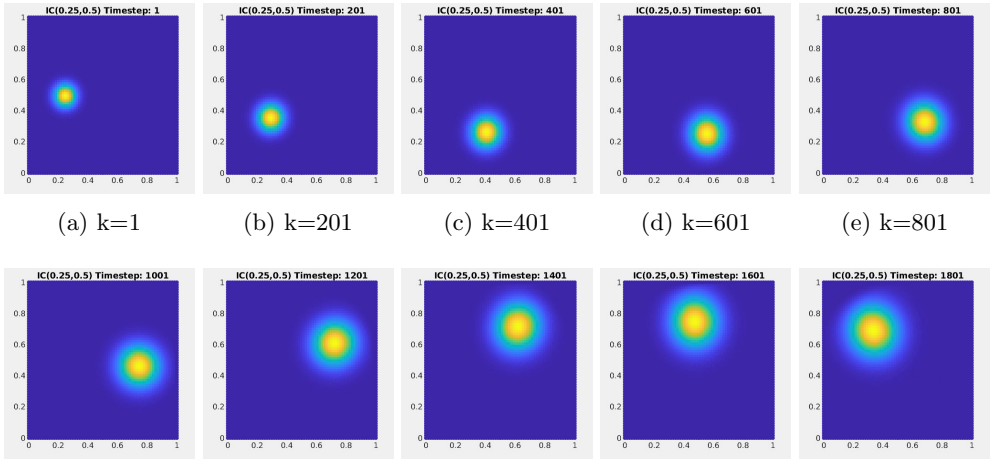


Fig. 5.1: The relative error spikes when the Gaussian is crossing the interface.

The associated trajectory is shown in Figure 5.2. As one can see, the trajectory is visually reproduced without any significant artifacts and the relative error to approximate $\boldsymbol{\lambda}$ is at worst on the order of $O(10^{-2})$.



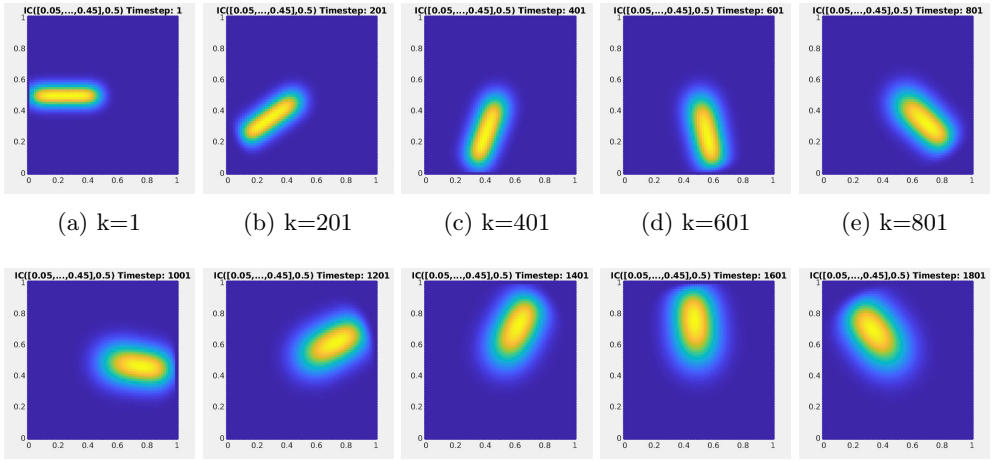
(a) k=1 (b) k=201 (c) k=401 (d) k=601 (e) k=801

(f) k=1001 (g) k=1201 (h) k=1401 (i) k=1601 (j) k=1801

Fig. 5.2: Advection and diffusion of Gaussian with initial condition position $(0.25, 0.5)$.

For a secondary test, combinations of the training data Gaussians were tested.

The first combination used all 9 training Gaussians; the plot can be seen in Figure 5.3. The associated error plot is shown in Figure 5.4.



(a) k=1 (b) k=201 (c) k=401 (d) k=601 (e) k=801

(f) k=1001 (g) k=1201 (h) k=1401 (i) k=1601 (j) k=1801

Fig. 5.3: Advection and diffusion of Gaussian with initial condition positions $([0.05, \ldots, 0.45], 0.5)$.

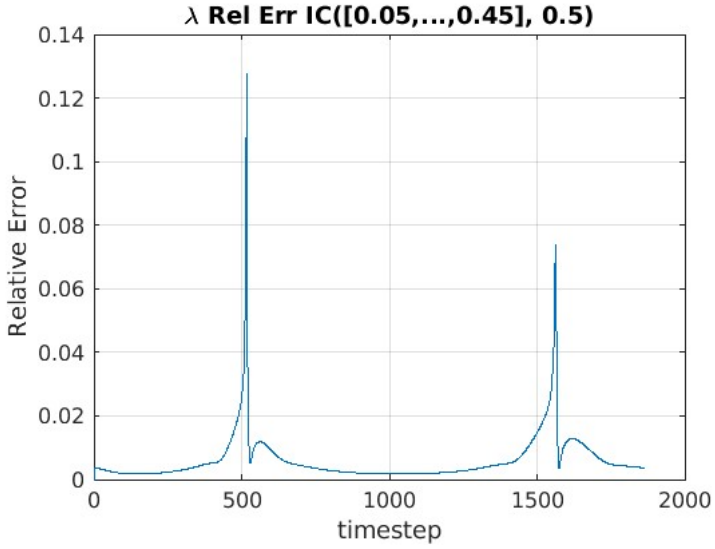
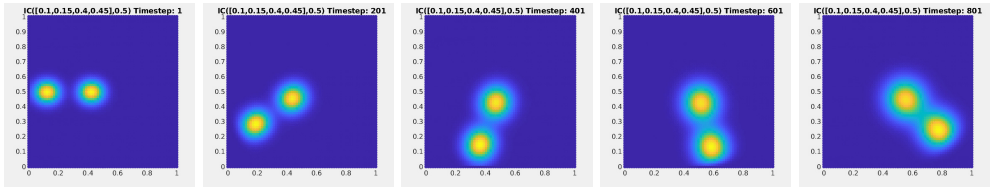
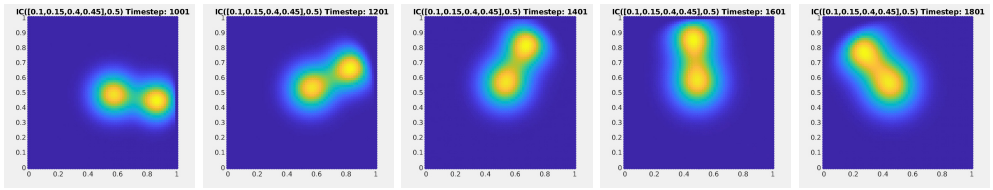


Fig. 5.4: The relative error spikes when the Gaussians are crossing the interface.

The second combination used Gaussians centered at $(0.1, 0.5), (0.15, 0.5), (0.4, 0.5), (0.45, 0.5)$, seen in Figure 5.5. The associated error plot can be seen in Figure 5.6.



(a) k=1 (b) k=201 (c) k=401 (d) k=601 (e) k=801



(f) k=1001 (g) k=1201 (h) k=1401 (i) k=1601 (j) k=1801

Fig. 5.5: Advection and diffusion of Gaussian with initial condition positions $([0.1, 0.15, 0.4, 0.45], 0.5)$.

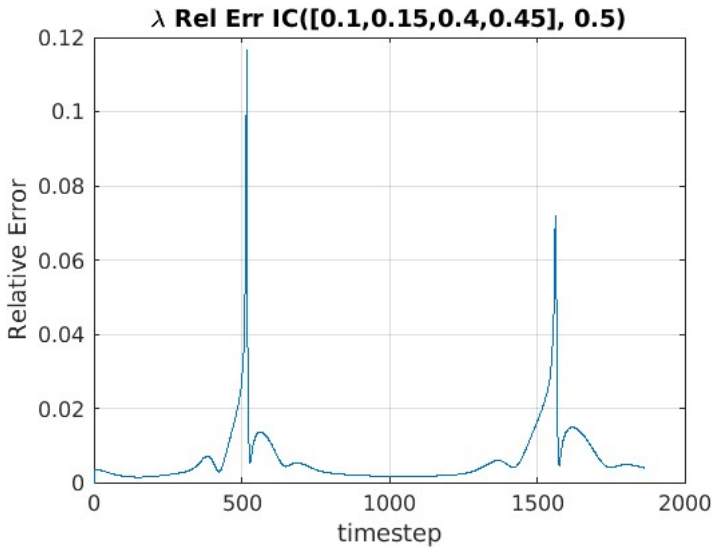
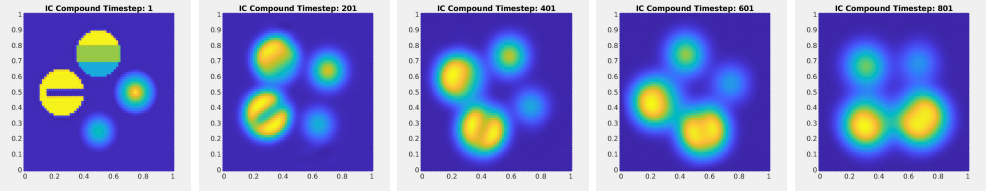


Fig. 5.6: The relative error spikes when the Gaussians are crossing the interface.

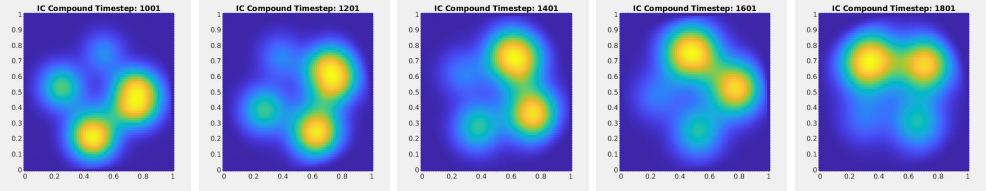
In the reproduction tests, the maximum relative error is on the order of $O(10^{-2})$. Visually, the plotted trajectories using the OpInf flux surrogate are indistinguishable from those generated by the IVR scheme that solves the Schur Complement.

5.2. Generalization to Arbitrary Initial Conditions. One test of interest is a compound test involving 4 elements: a slotted cylinder, a stepped cylinder, a Gaussian, and a cone [2]. The initial condition and IVR incorporated trajectory can be seen in Figure 5.7.

The corresponding error plot for the whole time interval is shown in Figure 5.8.



(a) k=1 (b) k=201 (c) k=401 (d) k=601 (e) k=801



(f) k=1001 (g) k=1201 (h) k=1401 (i) k=1601 (j) k=1801

Fig. 5.7: Advection and diffusion of Gaussian with compound initial condition positions.

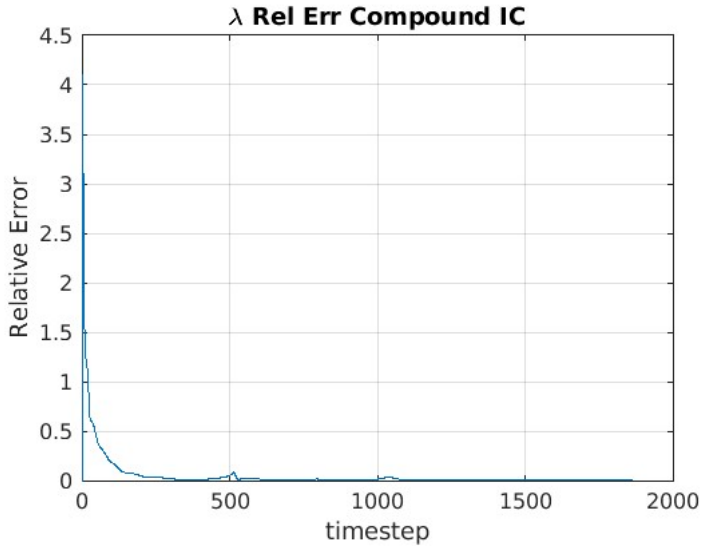


Fig. 5.8: The error drops off very quickly after the first few time steps.

Since the behavior of the relative error is distinctly different between the first few time steps and subsequent time steps, two error plots will be given for analysis. The first error plot can be considered as the relative error observed when the initial condition is discontinuous in nature; see Figure 5.9.

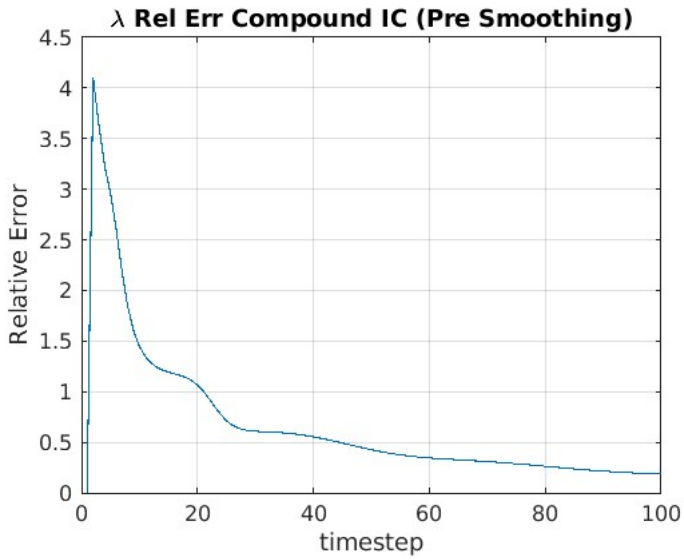


Fig. 5.9: The error during the first 100 time steps, when the solution is discontinuous, is maximal, but reduces as the solution smooths out via diffusion.

The second error plot shows the relative error after the solution has been smoothed via diffusion over time, see Figure 5.10.

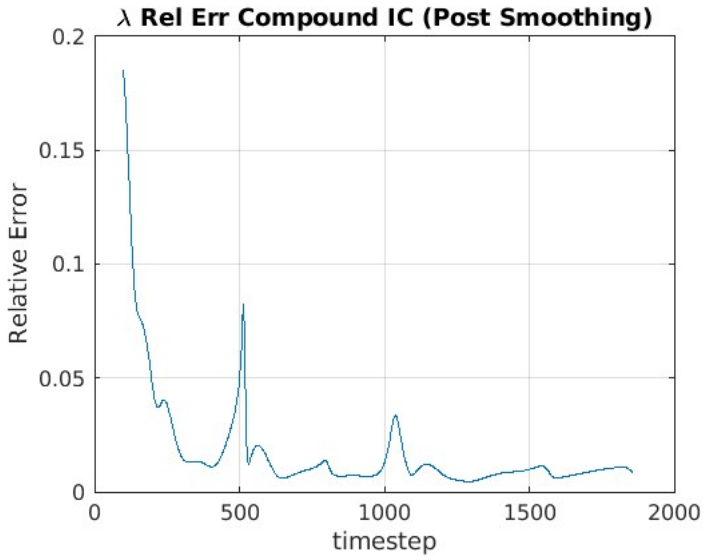
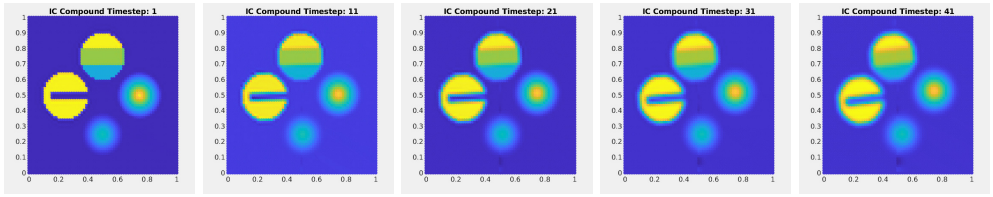
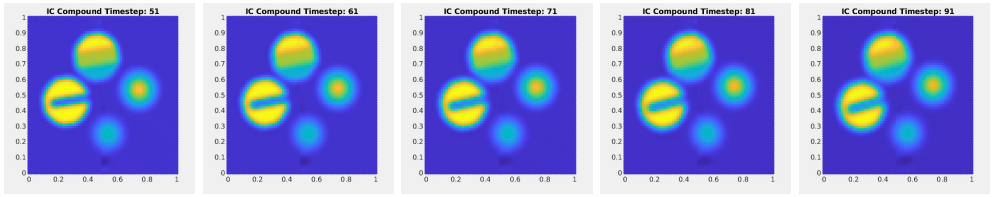


Fig. 5.10: The error after 100 time steps has order $O(10^{-2})$.

Examining the trajectory plot for the first time steps, an artifact is present that eventually smoothens out and disappears as shown in Figure 5.11. This behavior is reflected in the relative error plot Figure 5.9.



(a) k=1 (b) k=11 (c) k=21 (d) k=31 (e) k=41



(f) k=51 (g) k=61 (h) k=71 (i) k=81 (j) k=91

Fig. 5.11: During the first few times steps, the relative error spikes, but reduces within a few time steps. An artifact along the interface, more noticeable near the bottom, can be observed during these time steps.

6. Conclusion. In this work we developed and demonstrated a surrogate-based partitioned scheme in which the interface flux is approximated by an OpInf flux surrogate trained on suitable solution data. Reproductive tests with the OpInf surrogate trained on one set of 9 carefully chosen solution trajectories show acceptable accuracy and potential for the surrogate flux to serve as a more cost-effective substitute for the Schur complement flux in the IVR scheme.

The flux dynamics along the interface γ have been learned to a satisfactory degree as arbitrary initial conditions can be applied to the subdomains, and the dynamics across the interface transmit data between the two subdomains with acceptably low relative errors on the order of $O(10^{-2})$. Additionally, this reduced order operator can be generated very quickly, which speaks to the efficiency of this approach.

Further investigations on the diffusion smoothing effect observed for the compound test should be done. The relationship between the learned flux surrogate operator and smoothness of the initial conditions that reside on the flux boundary should be explored to better determine if more accurate surrogates can be learned. Recall that the discontinuity of the compound test yielded high relative error, 400%, for the first 100 time steps. After those 100 time steps, the simulation was sufficiently smoothed through diffusion to a relative error on order $O(10^{-2})$. It might be the case that two different surrogates one for pre-smoothing and one for post-smoothing may need to be trained and combined for the most accurate surrogate. This could be done by enriching the training data or by using OpInf with higher order terms.

Additional future work will involve applying the OpInf technique to more challenging nonlinear problems and changing the advection/diffusion parameters governing each subdomain.

REFERENCES

[1] P. Bochev, J. Owen, and P. Kuberry, *Dynamic flux surrogate-based partitioned methods for interface problems*, (2024).

[2] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM Journal on Numerical Analysis, 33 (1996), pp. 627–665.

[3] B. Peherstorfer and K. Willcox, *Data-driven operator inference for nonintrusive projection-based model reduction*, Computer Methods in Applied Mechanics and Engineering, 306 (2016), pp. 196–215.

[4] K. Peterson, P. Bochev, and P. Kuberry, *Explicit synchronous partitioned algorithms for interface problems based on lagrange multipliers*, Computers & Mathematics with Applications, 78 (2019), pp. 459–482. Proceedings of the Eight International Conference on Numerical Methods for Multi-Material Fluid Flows (MULTIMAT 2017).

TUSQH: TOPOLOGICAL CONTROL OF VOLUME-FRACTION MESHES NEAR SMALL FEATURES AND UGLY GEOMETRY

BRIAN SHAWCROFT*, KENDRICK M. SHEPHERD†, AND SCOTT A. MITCHELL‡

Abstract. This work develops a framework through which meshes with user-specified homology can be created from potentially ugly geometry by coupling background grids, persistent homology, and a generalization of volume fractions. For a mesh with fixed grid size, the topology of the output mesh changes predictably and monotonically as its volume fraction threshold decreases. Topological anti-aliasing methods are introduced to resolve pinch points and disconnected regions that are artifacts of user choice of grid size and orientation, making the output meshes suitable for downstream processes including analysis. The methodology is demonstrated on geographical and mechanical models in 2D and 3D using a custom-made software called Tusqh. The work demonstrates that the proposed framework is viable both for generating meshes on topologically invalid geometries and for automatic defeaturing of small geometric artifacts. Finally, the work shows that though subdividing the background grid for volume fraction-based meshing algorithms frequently improves topological and geometrical fidelity of the output mesh, there are simple 2D examples for which topology does not converge under refinement.

1. Introduction. Getting geometry that is suitable for mesh generation is often more difficult and time consuming than creating a mesh from that geometry. "Ugly" geometry is ubiquitous "in the wild." Industrial and commercial CAD data are often hand-designed "blueprints" to guide assembly, and do not represent the as-built part. Data from imaging and segmentation may have topological inconsistencies. Even in cases with valid geometry and topology, analysts must defeature models. This is because typical techniques generate meshes whose topology and geometry matches the input model. Thus the analyst must carefully review and modify the input based on the intended purpose of the mesh. Common issues in "ugly" geometry are gaps and overlaps, unneeded features smaller than the desired mesh size, topological complexities such as small holes, and small angles and thin regions that would produce poor-quality elements.

However, the mesh is a discrete approximation to the geometry. Why require a higher fidelity in the input than is aspired to in the output? Indeed, the community is developing tools to mesh ugly geometry, robustly producing meshes that are topologically correct and have high-quality elements despite topological and geometrical defects and small features in the input.

Sculpt [19, 21, 22] is one such tool, achieving a hexahedral mesh of reasonable quality, but reconstructing an approximation of the input geometry and topology. Inexact reconstruction is a *benefit* in the case of gaps, overlaps, and small features. The Sculpt algorithm starts with a background grid overlaying the input geometry. The fraction of each grid cell that lies inside the geometry of an input material is its *volume fraction*. Cells with volume fractions above a threshold (e.g., one-half) are retained; the rest are discarded. Heuristics remove undesirable topology such as pinch points and connected components consisting of only a few cells. Retained cells are then snapped to the geometry, and mesh quality is achieved through pillowing [15, 25, 26], smoothing [13], and other changes to mesh topology and node positions. Similarly, Morph [17, 24] is a parallel tet mesher using a background grid that snaps nodes to geometry based on dimension, proximity, and how other nodes are snapped. When no suitable node snap is found, Morph adds new nodes at the intersections with the geometry to produce nodes on the geometry boundary. In both Sculpt and Morph,

*Brigham Young University and Sandia National Laboratories Center for Computing Research, bshaw23@byu.edu

†Brigham Young University, kendrick_shepherd@byu.edu

‡Sandia National Laboratories Center for Computing Research, samitch@sandia.gov

the size of the background grid indirectly determines the geometric fidelity of the output to the input. TetWild [8, 9] does not use a background grid, but instead uses a Delaunay triangulation. The triangles representing the geometry are incrementally inserted and the mesh is refined. Edge length and geometric proximity parameters control the mesh resolution and geometric fidelity. Even for these tools that always produce meshes with valid topology, there is no a priori knowledge of what the homology of the output will be, nor how it will compare to the input topology or the desired topology.

Herein, we explore how to predict and achieve the desired mesh topology algorithms based on background grids and volume fractions, including Sculpt.

To accomplish this goal, Tusqh–a prototype mesher–was developed in Rhinoceros 3D. It is a testbed for research and demonstrates that our techniques are effective. It mimics the initial steps of Sculpt, using a background grid and volume fractions to decide which grid cells to retain. As with TetWild, it uses generalized winding numbers [1, 10] to define the "interior" for valid and invalid geometries. Subsequently, we explore the topological structure of potential meshes under different volume-fraction thresholds using persistent homology [6, 18]. We make local connectivity decisions based on sub-sampling volume-fractions, to mitigate the effects of the arbitrary orientation and offset of the background grid. This enables the analyst to measure and select the desired mesh topology, which then informs which volume-fraction threshold to select. The user may also visualize any mesh by adjusting the volume fraction threshold using a sliding scale. These meshes can serve as input for subsequent steps to improve geometric fidelity, such as Sculpt's snapping, pillowing, and smoothing. Finally, concluding theoretical results demonstrate that applications of grid-based volume fraction methods cannot guarantee consistent topological output.

2. Background Material. The proposed method, which selects which cells of a background grid to retain, is related to the computer graphics problem of rasterization [12]. Consequently, we first introduce background information about rasterization and anti-aliasing, after which fundamentals about tools employed in this work are introduced, including homology, persistent homology, and winding numbers.

Rasterization. Rasterization is the process of converting an arbitrary geometry into a grid-based representation of the shape. In traditional computer graphics, the background grid is screen pixels, the objects are represented by vector data or triangles embedded in floating point $\mathbb{R}^2$, and the problem is to select which color and intensity to display in each pixel. *Aliasing* [4, 14] is a significant problem in rasterization: pixel values are sensitive to the offset, rotation, and distance of the grid relative to the objects, as well as which locations within a pixel are sampled. For example, consider a non axis-aligned edge shared by a blue and a red triangle. For a given pixel, if we choose red or blue we get increased contrast but also stair-step patterns called "jaggies." If we choose a purple mixture the image appears more smooth but shading can produce Moiré patterns. Both are glaring to human eyes. For small triangles, the pixel topology may not match the triangle topology; see fig. 2.1. Such topological errors may be visually insignificant, but they can lead to significant errors in simulation results.

Topological Anti-aliasing. For volume-fraction meshing, deciding whether to retain a cell is sensitive to the offset and rotation of the objects (as in graphics), and grid size (analogous to pixel density in graphics). A parallel axis-aligned gap is closed or open depending on its size and position relative to the grid; see fig. 2.2. A gap smaller than half the grid size is always closed. A gap larger than the grid size is always open. Between these, shifting the grid left will cause the mesh to alternate between closed and open.

In fig. 2.3 we see the aliasing effects of rotations, where the feature is not aligned with the grid. In fig. 2.3a the gap is a fixed width, but about the size of the grid cells, leading

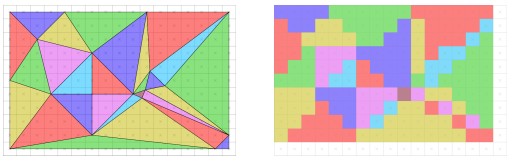


Fig. 2.1: Rasterization of triangles into pixels for computer graphics is shown. Note the pinches from the two left cyan triangles, the archipelago from the lower right pink triangle, and the multitude of additional topological errors in the lower right. Image courtesy Wikipedia `https://en.wikipedia.org/wiki/Rasterisation`

to the gap being inconsistently resolved as open or closed, with separate sides connected by pinch points. In fig. 2.3b we see a similar inconsistency, but exacerbated because the gap width varies. The boundary lines meet at a sharp angle, so fewer cells are retained as the apex is approached, leading to a chain of small disjoint mesh islands which we call an *archipelago.*

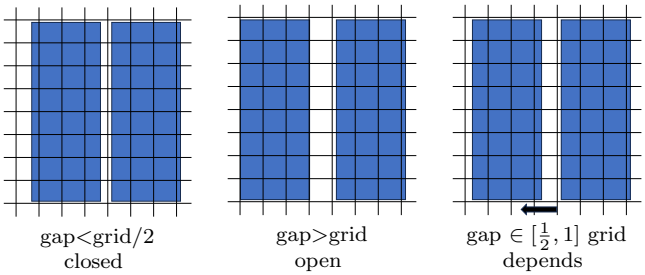


Fig. 2.2: Small grid-aligned gaps are closed, large are open, and for intermediates it depends on their offset.

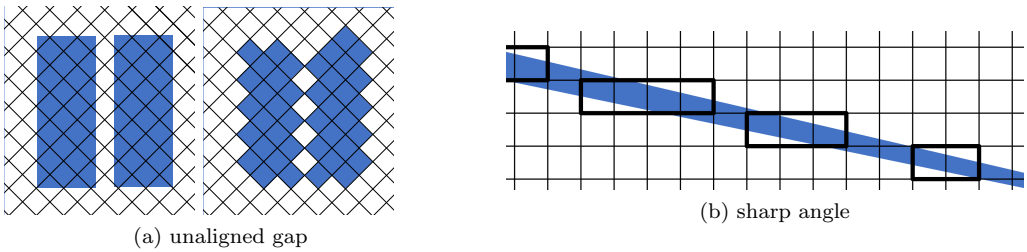


Fig. 2.3: In (a) the unaligned gap is resolved inconsistently. In (b) it is deomonstrated that rotational aliasing may cause stair-step patterns and archipelagos of islands near where two lines meet at a sharp angle (bold-outlined cells are filled, thin are open).

To address these undesirable local topological features, a topological anti-aliasing method is defined and coupled with introducing/removing various templated cells, as described in section 3. The first undesirable feature is pinches, where exactly two grid cells meet at a vertex with no shared edge, or exactly two 3D grid cells meet at an edge with no shared faces. These must be removed because the mesh is required to be locally connected face-to-face or disjoint. (The complement is also connected face-to-face or disjoint.) All other ways in which a mesh can be non-manifold do not occur, because we form the mesh from the union of some cells of a structured grid. Pinches are either separated into different components by removing small elements, or thickened into meeting face-to-face by adding

small elements using data from the anti-aliasing framework. These small elements come from templates that split background grid cells and perform swaps. The second undesirable feature is archipelagos. Our anti-aliasing technique joins some islands with template elements around connecting edges and quads, and removes any small islands that remain. For comparison, Sculpt resolves pinches by adding or removing entire grid cells, and resolves small components by removing them [20].

For simplicity we only discuss domains with a single material and only discuss the retained cells. In principle our anti-aliasing could be extended and applied to multi-material volume-fraction meshing [26, 20].

Homology. The topology of a mesh should contain the significant features of a domain for its intended computational analysis. Herein, we shall study mesh topology using a cellular complex: nodes are zero-cells, edges are one-cells, faces are two-cells, and volumes are three-cells. Specifically, we will make use of simplicial and cubical complexes, in which two-cells are triangles and quadrilaterals, and three-cells are tetrahedra and hexahedra, respectively. Homology [7] is a mathematical tool that distinguishes cell complexes using certain algebraic quotient groups. The *Betti numbers* B_i count the rank of these groups. Specifically, B_0 equals the number of connected components, B_1 is the number of holes, and B_2 is the number of cavities or voids. For planar domains B_2 will always be zero.

Persistent homology [6, 18] describes homology changes as objects are added and connections are made. A *filtration* has a "persistence parameter" which defines when a cell enters the complex. A filtration is monotonic, so no cell may ever leave the complex after entering. However, the homology has both additions and removals because adding a cell could, e.g., create a new connected component, or combine two components into one. The parameter value at which a group generator is created is called its "birth," while the value it disappears is called its "death." Birth and death coordinates are plotted in a persistence diagram such as fig. 3.2d. This not only counts Betti numbers, but tracks individual components and holes.

This work studies the persistent homology of cubical background grids using volume fractions as the persistence parameter. (In other work the signed distance to a domain boundary was the parameter [16].) Alternatively, zigzag persistence, which does not require a monotonic filtration [3, 5], could be used, but doing so would increase complexity and computational expense.

Winding Numbers. Volume fractions may be estimated by sampling points and counting the fraction of them inside the geometry. However, for "ugly" geometry, what is "inside" may be poorly defined. The generalized winding number [1, 10] overcomes this obstacle; see fig. 2.4. It gives answers identical to ray shooting for watertight domains, and gives answers that humans find both reasonable and intuitive for other domains. In its traditional form, the winding number at a point with respect to a closed curve describes the net number of times the curve encircles the point in the counter-clockwise direction, with negative numbers indicating clockwise encirclement. The winding number is the integral of the angle of the ray from the point to the curve as it is traversed. The generalized winding number extends this definition to sets of open curves. It yields a continuous value where 0 indicates outside

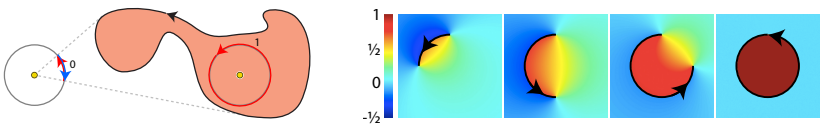


Fig. 2.4: Winding number point and field values, courtesy Figures 4 and 6 from Jacobson et al. [10].

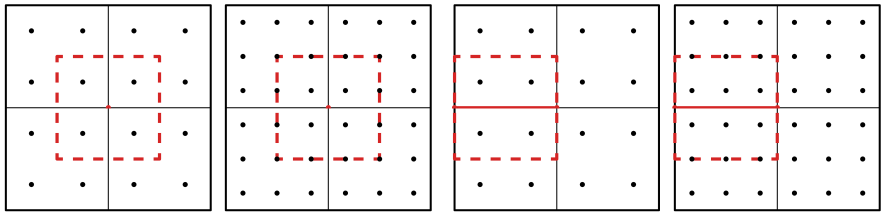
and 1 is inside. For geometry with gaps, the winding number near a gap is typically between 0 and 1. In extreme cases, such as overlapping domain boundaries, invalid geometries may give values beyond $[0, 1]$. In 3D, the winding number integrates the solid angles seen from a point, e.g., for a volume defined by a triangle soup. Which normal direction is outward-facing determines the sign of the solid-angle contribution.

3. Methodology.

Volume Fractions. Herein we study both 2D and 3D domains. We define a regular background grid, e.g. by subdividing an axis-aligned bounding box. This grid is a cubical quadrilateral or hexahedral complex, depending on the domain dimension. The volume fraction of each maximally-dimensioned cell is computed as the average of the winding numbers of its sample points. Sample points lie in an s^d array, as shown in fig. 3.1. (Recall we calculate persistent homology based on volume fractions, with the goal that the user may select the volume fraction that achieves their desired mesh topology.)

Volume-Fraction Persistence Parameter. The persistence parameter used herein is the volume-fraction threshold, ordered from 1 down to 0 (i.e. by decreasing value). By defining volume fractions only for cells of maximal dimension, a mesh is defined by including all cells with volume fraction greater than or equal to an input volume fraction and removing all others. The order that cells are added to the mesh as a function of the persistence parameter is demonstrated on a topologically invalid representation of the Chesapeake Bay in fig. 3.2. The persistent homology diagram is displayed in fig. 3.2d. These images illustrate significant topological aliasing, which limits the utility of these meshes. In what follows, we aim to mitigate these rasterization effects.

Sub-cell Volume Fractions. To assist in topological anti-aliasing, we also define volume fractions for all lower-dimensional grid cells. For any such n-cell, its sample points are those in a (fictitious) grid cell centered at that n-cell; see fig. 3.1. We use only even numbers s of sample points because these samples do not lie on cell boundaries. Volume fractions for these lower-dimensional cells are computed as averages of winding numbers associated with sample points contained in the fictitious grid cell, in a manner analogous to cells of maximal dimension. As before, only cells with volume fraction greater than or equal to a prescribed threshold will remain in the cell complex. All others are omitted. Omitted cells are called "exterior" cells, while remaining cells are called "interior" cells. We call this sampling process "*subgrid sampling.*"

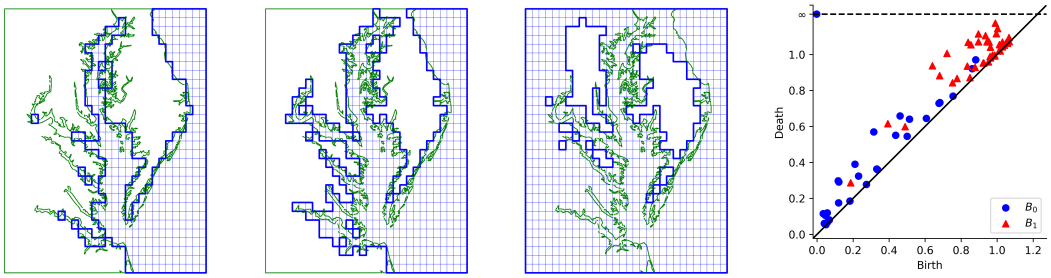


(a) Vertex, even (b) Vertex, odd (c) Edge, even (d) Edge, odd

Fig. 3.1: Sample $s \times s$ arrays are shown. Samples contained by th red dashed lines define vertex and edge volume fractions in 2D. We use only even sample arrays.

Anti-aliasing. To address the undesirable rasterization effects introduced by our background grid, we employ subgrid sampling as an anti-aliasing method.

In 2D, the only possible pinch is two quads meeting at a pinch vertex, whereas in 3D there are 11 possible configurations of pinched edges and vertices. These are shown in



(a) Vol. Frac.: 50%; $B_0 = 4; B_1 = 2$ (b) Vol. Frac.: 15%; $B_0 = 1; B_1 = 7$ (c) Vol. Frac.: 1%; $B_0 = 1; B_1 = 12$ (d) Persistence Diagram

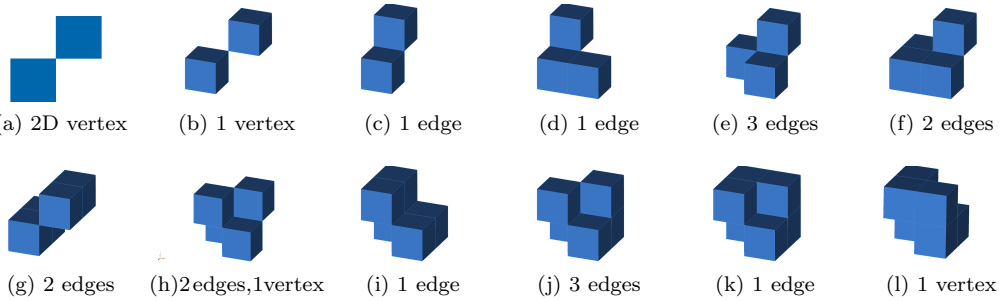
Fig. 3.2: Chesapeake Bay meshes and their Betti numbers change as the volume fraction threshold is lowered. The model contains deliberate errors: gaps, overlaps, and offsets. As a result, the winding numbers of sampled points may be positive in regions that are clearly inland, as in the lower-left region of (c). Also note the effects of rasterization when incorporating volume fractions of elements in the background grid without any anti-aliasing. In (d) the persistence diagram is shown, with parameter 1 minus the volume fraction.

fig. 3.3. To find pinches, we consider each vertex and the neighborhood of cells containing it, i.e., 2×2 quads in 2D and $2 \times 2 \times 2$ hexes in 3D. If the neighborhood corresponds to a pinch case, the pinch vertices and edges are queued. Each pinch in the queue is processed in a way that is compatible with processing nearby pinches. The pinches are connected if the subcells (vertices or edges) are interior, and disconnected if they are exterior. Each pinch is repaired by splitting cells (either mesh cells or their complement) using predefined splits, and discarding or adding some of the split cells. The splits are shown in fig. 3.4. In 2D, the template is a one-to-five split. In 3D, pinch edges are repaired before vertices. The edge-repair template is a one-to-seven split. For pinch vertices, we follow with a two-to-six split of any pairs of hexes from two different one-to-seven splits that share a face; see fig. 3.4c.

To separate cells, splits are performed on the cells of the mesh itself, and child cells that contain pinch vertices or edges are removed, as shown in fig. 3.5. To connect cells, splits are performed on the complement of the mesh, and child cells that contain pinches are added to the mesh, as illustrated in fig. 3.6. A single mesh can use both separations and connections in different regions. However, we require that all adjacent pinches must be resolved in the same way to ensure validity of the resulting mesh. Two sets of pinches separated by cells without pinches can be resolved in opposite ways. Our rules occasionally indicate that adjacent pinches should be resolved in opposite ways. We pre-select whether we connect or separate these cases.

When separating pinches, the configuration in fig. 3.3d is the only exception to the rule of removing all the child cells that share the pinch edge. The one-to-seven split is performed on the hexahedra sharing the pinch edge, and all the child hexahedra that contain that edge are removed. This turns the central vertex into a pinch. To prevent that, one additional child hexahedron is removed. In the orientation of fig. 3.3d, the removed hex is the rightmost child of the top hex, which contains the central vertex; see fig. 3.5d.

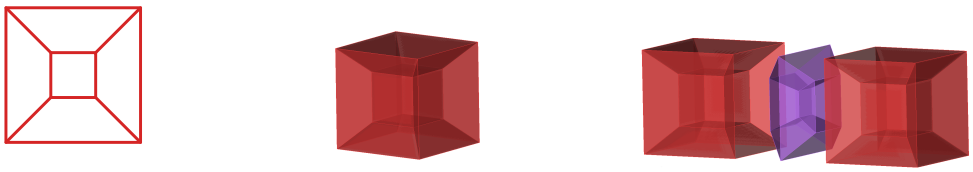
For resolving archipelagos, all the connected components of the mesh are identified. For any pair of connected components, if the edges that connect them are interior to the geometry, the components are joined using templates along those edges. The remaining connected components that contain fewer than a user-defined number of highest-dimensional cells are removed. In fig. 3.7 the utility of the anti-aliasing algorithm is demonstrated on



(a) 2D vertex (b) 1 vertex (c) 1 edge (d) 1 edge (e) 3 edges (f) 2 edges

(g) 2 edges (h)2edges,1vertex (i) 1 edge (j) 3 edges (k) 1 edge (l) 1 vertex

Fig. 3.3: All possible pinches in 2D and 3D are shown.



(a) 2D 1–5 split (b) 3D 1–7 split (c) 3D 2–6 split

Fig. 3.4: Templates are displayed for fixing pinches.

the unaligned gap and sharp angle of fig. 2.3.

Transferring Persistent Parameters to Simplices. Having a framework by which topological anti-aliasing can be performed on the mesh by use of subgrid sampling and templates, we now turn our attention to ensuring that the topological anti-aliasing defined above is accurately represented in persistent homology calculations. Because most open-source persistent homology software employs simplicial complexes, we first transform the cubical filtration into a topologically-equivalent simplicial filtration. In what follows, we focus on consistency with pinch resolution, rather than also on archipelago resolution. As a result, we make the assumption in 2D that an edge shared by two interior quadrilaterals will also be interior, and an edge shared by two interior vertices must also be interior. Similarly, in 3D, a face shared by two interior hexahedra will be interior, as will a face bounded by four interior edges and vertices. We first focus on the 2D framework, then the 3D framework.

In 2D, a primal vertex induces a dual 2-cell, and a primal edge induces a dual edge, and a primal quad induces a dual vertex. Each dual vertex is assigned a filtration value of its corresponding primal face's volume fraction. The dual mesh is then further subdivied into a simplicial mesh. To create the simplicial mesh, an additional simplicial vertex is introduced at the centroid of each dual 2-cell. Simplices are then formed as the join of each dual face's edge with the simplicial centroid vertex. This simplicial vertex corresponds to a vertex on the primal mesh, and takes the filtration value of the corresponding primal vertex's volume fraction. However, to preclude the introduction of spurious topological artifacts (and consistent with previous computations), we also require that this volume fraction be between the maximal and minimal volume fraction of the surrounding four vertices. The filtration value of this simplicial vertex then informs whether two primal faces connected with a pinch should be topologically separated or connected. Having thus defined filtration values for all vertices of the induced simplicial complex, we then use a Vietoris–Rips complex to calculate persistent homology, meaning that at persistence value $k \in \mathbb{R}$, all vertices with persistence parameter value greater than or equal to k are added to the filtration, then all edges between already-added vertices, then all triangles formed by already-added edges.

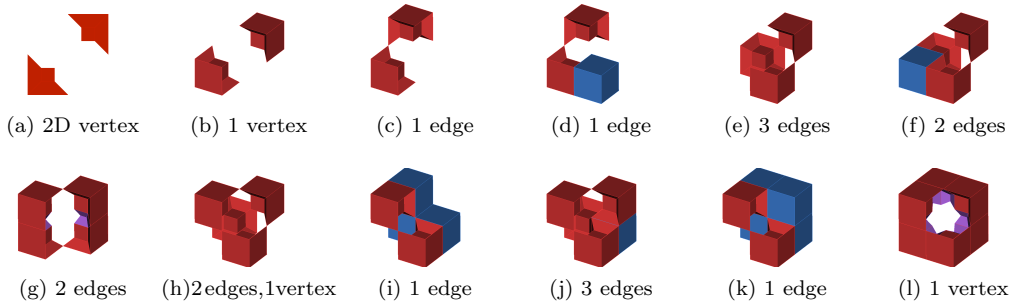


Fig. 3.5: Templates for shrinking pinches about a central vertex are shown. Pinched vertices on the boundary of the depicted neighborhoods are resolved when the template window is shifted onto the pinched vertex.

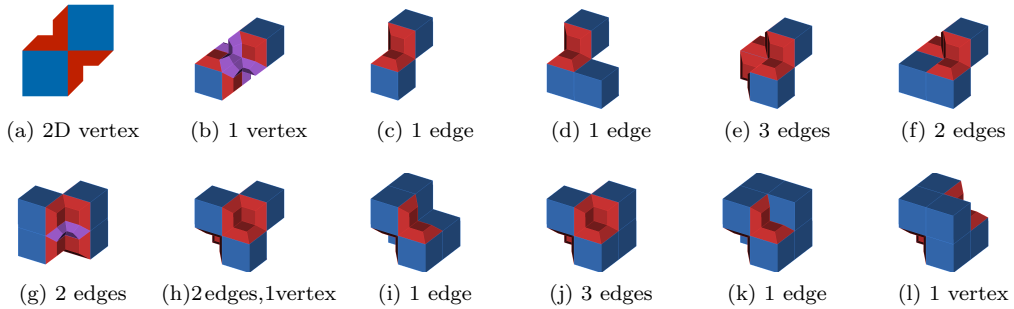


Fig. 3.6: Templates for growing pinches are shown.

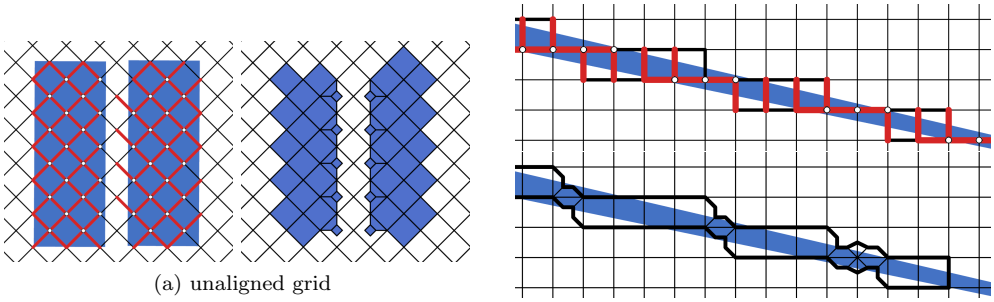
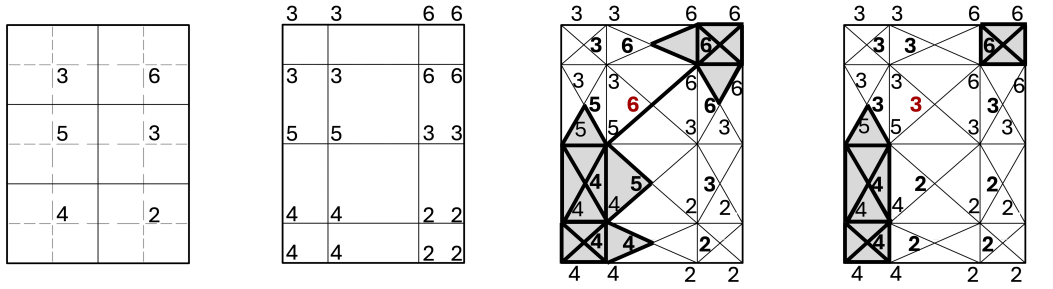


Fig. 3.7: Subgrid sampling and anti-aliasing performed on fig. 2.3.

In 3D, a primal vertex induces a dual volume cell, and a primal edge induces a dual face, a primal face induces a dual edge, and a primal volume induces a dual vertex. As in 2D, each dual vertex is assigned the filtration value of its primal volume. To generate a simplicial mesh from the dual mesh, each dual face is subdivided into four triangles with a new vertex introduced, as in 2D. Here, the additional vertex corresponds to a primal edge and will take filtration value of the volume fraction of this primal edge, subject to constraints keeping the filtration value between the maximal and minimal values of the surrounding four simplices of the dual face. Each dual volume is then subdivided into 24 tetrahedra by introducing a single simplicial vertex at the centroid of the dual volume and



(a) Primal volume fractions

(b) Dual volume fractions

(c) Connected simplices

(d) Disconnected simplices

Fig. 3.8: An example of converting 2D cell volume fractions into a filtration of a simplicial complex is shown. Numbers indicated are numerators, and are to be divided by six to arrive at volume fractions. Herein, we consider cells with volume fraction strictly greater than 0.5. The leftmost figure indicates the ordering of grid cells. The next shows the dual grid with vertex values transferred from grid cells. Finally, the dual complex is subdivided into a simplicial complex. The choice of volume fraction for the introduced vertices (in bold) will lead to varying connectivity of non-manifold regions.

taking the join of this vertex with each of the 24 triangles defined on the (subdivided) faces of the dual volume. Again, this new simplicial vertex will correspond to a vertex on the primal mesh, and consequently takes a filtration value of the volume fraction of this primal mesh vertex (again subject to the constraint that the filtration value must be between the maximal and minimal values of the 26 surrounding vertices). The filtration values of the simplicial vertices corresponding to primal vertices and primal edges is then locally consistent with the procedures performed resolving pinch points, and will have identical persistent homology. This conversion process, from a cubical primal cell complex into a simplicial one with an identical filtration is illustrated in figs. 3.8 and 3.9.

For the sake of completeness, we also note that similar primal to dual to simplicial operations could be performed on unstructured background meshes. In the 2D case, each dual cell of maximal dimension and with k sides would be subdivided into k triangles. In 3D, each dual cell of maximal dimension and with ℓ faces, with the ith face having k_i sides, would subdivide into $\sum_{i=1}^{\ell} k_i$ tetrahedra.

Finally, we note that for a truly general framework, a vertex in the above-defined simplicial complexes would need to be defined for each cell in the primal complex. Particularly, in 2D we currently introduce vertices corresponding to primal faces and vertices, but not for primal edges. This would require splitting every dual face into 8 simplices, rather than 4. In 3D, we introduce vertices for all cells except for primal faces, and would require splitting every dual volume into 48 tetrahedra, rather than 24. Given the challenges of navigating this topological space in a meaningful way (as well as the additional computational expense incurred by such a navigation), we leave this for future work.

4. Results.

4.1. Computational Results. Tusqh was developed and evaluated using custom plugins to Rhinoceros 3D and Grasshopper. Winding numbers were computed using libigl [11]. Persistent homology was computed using Aleph [23], which is based on PHAT [2].

The framework is tested on an oriented planar representation of the Chesapeake Bay[1]

[1] Model derived from `https://vecta.io/symbols/281/ecosystems-maps/93/usa-md-va-chesapeake-bay-line-map`

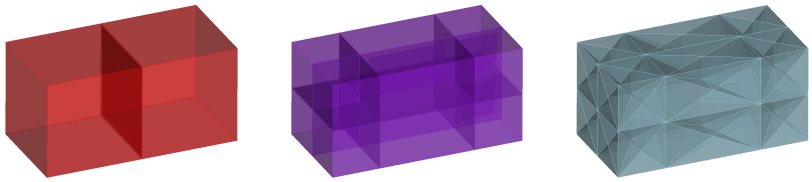
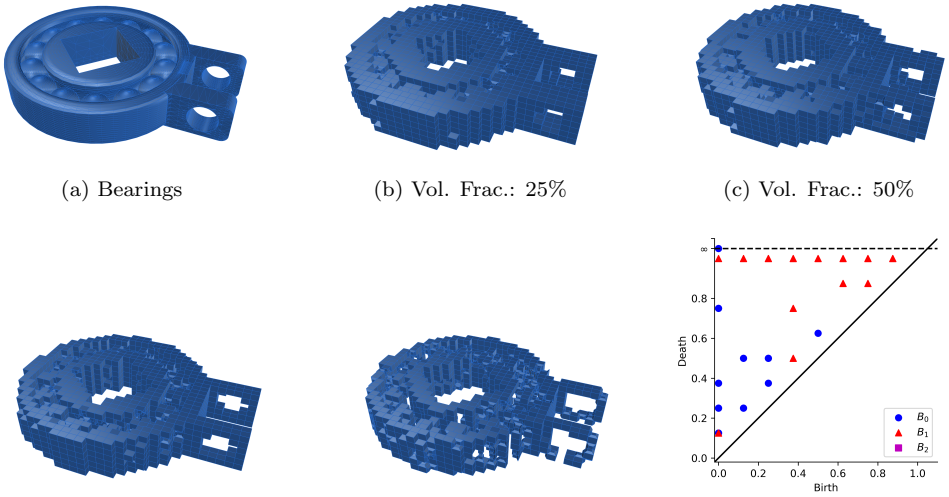


Fig. 3.9: All faces of two adjacent 3D hexes are converted into a filtration of a tetrahedral simplicial complex.



(a) Bearings (b) Vol. Frac.: 25% (c) Vol. Frac.: 50%

(d) Vol. Frac.: 75% (e) Vol. Frac.: 100% (f) Persistence Diagram

Fig. 4.1: Meshes of bearings for various volume fraction thresholds are shown. In (f) the persistence diagram is shown, with paramerter 1 minus the volume fraction.

and mechanical bearings [2]. Snapshots of given computed volume fractions are shown in fig. 3.2 and fig. 4.1. Model errors include overlapping edges, repeated/offset edges, and numerous gaps. Despite the "interior" of the bay being ill-defined, the proposed method still captures the intended geographic domain with respect to both the continent and to islands. A complete view of the homological structure based on varying the volume fraction is shown in the persistence diagram of figs. 3.2d and 4.1f. Results demonstrate that a mesh with the desired homological structure could be extracted from the background grid by selecting the correct threshold. These figures are primarily for illustrative purposes. We purposely chose a coarse grid size to generate the topological issues we are addressing. In practice, a finer grid would better capture local behavior.

Anti-aliasing. Figure 4.2 demonstrates the anti-aliasing technique on a mesh of the Chesapeake Bay to resolve both pinches and archipelagos. The anti-aliased mesh is analysis suitable, although in practice a refined mesh would be used as input for analysis.

4.2. Anti-aliasing Guarantees and Limitations. As noted in Section 2, one of the primary difficulties with volume fraction-based meshing methods is mitigating the effects of rasterization (i.e. choice of orientation and sample size) through a topological anti-aliasing. The following theoretical result holds regarding the success of the proposed anti-aliasing

[2]Model provided at `https://ten-thousand-models.appspot.com/detail.html?file_id=1716283`

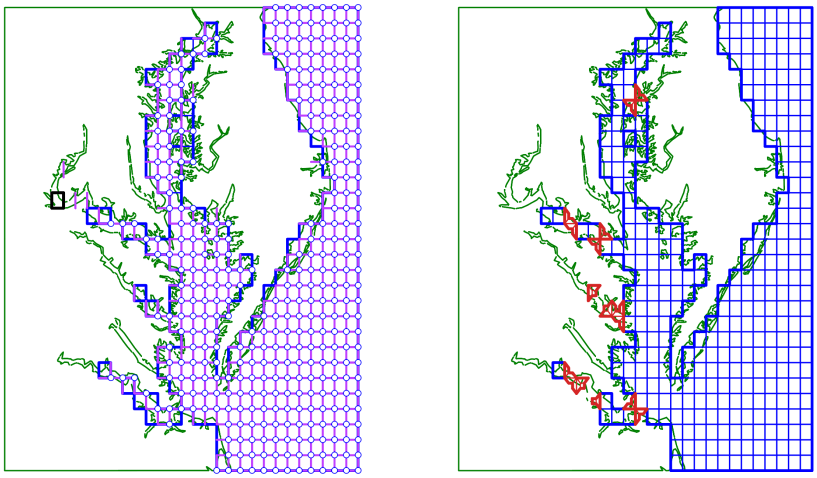


Fig. 4.2: Meshes of Chesapeake bay with subcells and anti-aliasing are shown. Interior vertices and edges are blue outlined circles and purple lines respectively, while removed faces are shown in black (left). Connecting and separating templates are in red (right).

methods in mitigating topological rasterization. A proof of the result can be found in the appendix of the preprint version of this document on arXiv.

THEOREM 4.1. *Given a rectangular lattice in* $\mathbb{R}^2$ *with characteristic length* ℓ *overlaying two parallel half-spaces separated by a length of* L*, topological rasterization may occur when* $\ell(\sqrt{2}-1) < L \leq \ell$*. For the subgrid sampling scheme proposed in this text, topological rasterization may only occur when* $\ell(\sqrt{2}-1) < L < \frac{\sqrt{2}\ell}{2}$*. Finally, topological rasterization due to changes in orientation cannot be resolved for* L *such that* $\ell(\sqrt{2}-1) < L < \frac{\ell}{2}$*.*

4.3. Topological Effects of Grid Refinement. At the beginning of our project we conjectured that persistent homology can measure the necessary grid size to achieve a desired topology, but this turned out not to be true in some cases. When features are isolated or globally the same scale, grid refinement has intuitive and predictable topological effects. However, we discovered that this is *not* true for general inputs. Counterexamples show non-monotonic filtration behavior by grid size. Discretization by grid cells and their alignment with input features strongly effects topological behavior. Thus algorithm parameters of when to refine the background grid may have unexpected effects on mesh topology.

Convergence. For some inputs, as the grid is refined, topological features of the input are resolved and the output mesh topology becomes stable. However, for some other inputs, the topology never converges and no filtration is possible. For some inputs it may be possible to define a filtration, with simplices only appearing, never disappearing. If simplices appear and disappear, then zig-zag persistence could computationally predict topology.

In fig. 2.3 a feature is inconsistently resolved due to aliasing effects of unaligned grids. For the constant-width gap in fig. 2.3a, refining or coarsening the grid makes the gap resolved consistently as open or closed. However, for the variable-sized gap in fig. 2.3b, global uniform refinement merely moves where the problem occurs. The example is a wedge of material bounded by two lines meeting at a small angle α at an apex. In locations where the grid size is about the same as the local width, whether a cell is included or excluded can change every few grid cells, leading to many separate connected grid components. For any small grid size, there will be some portion of the wedge where the lines are about that size apart, specifically in the range $[\frac{1}{2}, 1] \cot \alpha$ squares away from the apex. The geometry is undesirable

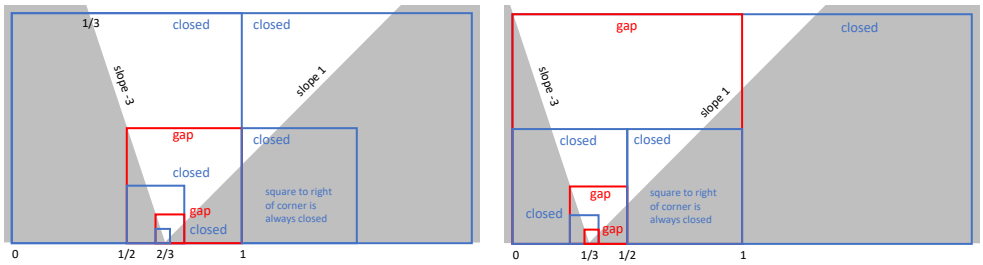


Fig. 4.3: In these counterexamples to convergence, the grid topology alternates between one and two connected components ad infinitum under refinement. The alternations of the left and right examples are opposite.

because the islands move location. The grid topology may be constant over refinement, but that topology is undesirable.

Non-convergence. In each of the examples in fig. 4.3 the output mesh topology does not converge under refinement. That is, there is no grid size below which the output mesh topology does not change. The background grid is uniform, but we only draw some of the relevant cells at each level of refinement. Blue (closure) cells are mostly material and thus included in the output mesh. Red (gap) cells are unfilled and excluded. Under refinement, the meshes alternate between one and two connected components ad infinitum. The grid squares containing the corner alternate between filled and open, because of the corner's relative position inside its square. The descriptions of the geometries are finite, just two triangular blocks. It is simple, plausible, and without sharp angles. The only unusual feature is the blocks touch at a single pinch point.

The left and right examples in fig. 4.3 have alternate sizes of when they are open and closed. If an input has both of these pinch features between two material blocks, then exactly one of the pinches will be closed, giving a mesh with the homology of a disk. It is converged in the sense that the homology does not change under refinement, but the local connectivity does change. Hence, even if we were to use zigzag homology it would not distinguish between this case and a single solid block of material.

The analytic description of the geometries in fig. 4.3 is two triangular blocks of material with slopes -3 and 1 meeting at a corner. In the upper example, the corner's coordinate is $(\frac{2}{3}, 0)$, and in the lower it is $(\frac{1}{3}, 0)$. If we start with a unit grid with a vertex at the origin, then under refinement the grid square containing the corner alternates between having the corner $2/3$ of the way along the bottom edge (blue), and $1/3$ of the way (red). Such blue squares have volume fraction $10/18$ and the red squares $7/18$. This construction is not tight. The slopes may be different. The corner may lie at some other coordinate, and a grid vertex will never lie on it if its x-coordinate is not $k/2^m$ for some $\{k, m\} \in \mathbb{Z}$. Thus more complicated sequences than alternating may be constructed.

5. Conclusion. In this work, a mesh generation framework is developed to facilitate the creation of meshes on potentially ugly geometry based on user-specified needs through the use of persistent homology. The framework is built on a volume-fraction based meshing method, similar to Sculpt; a desired mesh can be selected based on the appropriate homological structure induced by this volume fraction. The software, Tusqh, demonstrates the potential of the meshing framework in both two and three dimensions, and is planned for open source release. As a counterpoint we have theoretical analysis showing that for any volume-fraction threshold we choose, there are cases where aliasing artifacts will still arise.

REFERENCES

[1] G. Barill, N. G. Dickson, R. Schmidt, D. I. Levin, and A. Jacobson, *Fast winding numbers for soups and clouds*, ACM Transactions on Graphics (TOG), 37 (2018), pp. 1–12.

[2] U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner, *PHAT – persistent homology algorithms toolbox*, Journal of Symbolic Computation, 78 (2017), pp. 76–90. Algorithms and Software for Computational Topology.

[3] G. Carlsson and V. De Silva, *Zigzag persistence*, Foundations of Computational Mathematics, 10 (2010), pp. 367–405.

[4] F. C. Crow, *The aliasing problem in computer-generated shaded images*, Communications of the ACM, 20 (1977), pp. 799–805.

[5] T. K. Dey and T. Hou, *Computing zigzag vineyard efficiently including expansions and contractions*, in 40th International Symposium on Computational Geometry (SoCG 2024), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[6] H. Edelsbrunner, D. Letscher, and A. Zomorodian, *Topological persistence and simplification*, Discrete Computational Geometry, 28 (2002), pp. 511–533.

[7] A. Hatcher, *Algebraic Topology*, Cambridge University Press, 2001.

[8] Y. Hu, T. Schneider, B. Wang, D. Zorin, and D. Panozzo, *Fast tetrahedral meshing in the wild*, ACM Trans. Graph., 39 (2020).

[9] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo, *Tetrahedral meshing in the wild*, ACM Trans. Graph., 37 (2018).

[10] A. Jacobson, L. Kavan, and O. Sorkine-Hornung, *Robust inside-outside segmentation using generalized winding numbers*, ACM Trans. Graph., 32 (2013), p. 33.

[11] A. Jacobson, D. Panozzo, et al., *libigl: A simple C++ geometry processing library*, 2018. https://libigl.github.io/.

[12] J. Jon Hasselgren, T. Akenine-M oller, and L. Ohlsson, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Addison-Wesley Professional, 2005, ch. 42, Conservative Rasterization. `https://developer.nvidia.com/gpugems/gpugems2/part-v-image-oriented-computing/chapter-42-conservative-rasterization`.

[13] P. Knupp, *Introducing the target-matrix paradigm for mesh optimization by node movement*, Engr. with Comptr., 28 (2012), pp. 419–429.

[14] D. P. Mitchell, *The antialiasing problem in ray tracing*, Advanced Topics in Ray Tracing, SIGGRAPH 1990 Course Notes, (1990).

[15] S. A. Mitchell and T. J. Tautges, *Pillowing doublets: Refining a mesh to ensure that faces share at most one edge*, Proc. 4th Int. Meshing Roundtable, 1995, (1995). Sandia National Laboratories technical report SAND-95-2356C.

[16] C. Moon, S. A. Mitchell, J. E. Heath, and M. Andrew, *Statistical inference over persistent homology predicts fluid flow in porous media*, Water Resources Research, 55 (2019).

[17] D. Noble, M. Staten, and C. R. Wilson, *Using a faceted geometry representation to improve the performance of overlay grid meshing*, Tech. Rep. SAND2024-08942A, Sandia National Laboratories, Albuquerque, NM U.S.A., 2024. Research abstract, SIAM IMR24.

[18] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, *A roadmap for the computation of persistent homology*, EPJ Data Science, 6 (2017), p. 17.

[19] S. J. Owen, *Parallel smoothing for grid-based methods*, International Meshing Roundtable, Research Note, (2012), pp. 161–178.

[20] S. J. Owen, J. A. Brown, C. D. Ernst, H. Lim, and K. N. Long, *Hexahedral mesh generation for computational materials modeling*, Procedia Engineering, 203 (2017), pp. 167–179.

[21] S. J. Owen and T. R. Shelton, *Evaluation of grid-based hex meshes for solid mechanics*, Engineering with Computers, 31 (2015), pp. 529–543.

[22] S. J. Owen, M. L. Staten, and M. C. Sorensen, *Parallel hex meshing from volume fractions*, in International Meshing Roundtable, W. R. Quadros, ed., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 161–178.

[23] B. Rieck et al., *Aleph — a library for exploring persistent homology*, 2016. https://github.com/Pseudomanifold/Aleph.

[24] M. Staten, D. Noble, and C. R. Wilson, *Constructing tetrahedral meshes no matter how ugly the CAD*, Tech. Rep. SAND2024-03643C, Sandia National Laboratories, Albuquerque, NM U.S.A., 2024. Research abstract, SIAM IMR24.

[25] M. L. Staten, J. F. Shepherd, and K. Shimada, *Mesh matching — creating conforming interfaces between hexahedral meshes*, in Proceedings of the 17th International Meshing Roundtable, R. V. Garimella, ed., Berlin, Heidelberg, 2008, Springer Berlin Heidelberg, pp. 467–484.

[26] Y. Zhang, T. J. Hughes, and C. L. Bajaj, *An automatic 3D mesh generation method for domains with multiple materials*, Computer Methods in Applied Mechanics and Engineering, 199 (2010),

pp. 405–415. Computational Geometry and Analysis.

PARALLEL INCOMPLETE LU FACTORIZATIONS BASED ON ALTERNATING TRIANGULAR SOLVES

MARC A. TUNNELL* AND ERIK G. BOMAN†

Abstract.
Incomplete factorizations are popular preconditioners and are well known to be effective for a wide range of problems. Additionally, these preconditioners can be used as a "black box" and do not rely on any *a priori* knowledge of the problem. However, traditional algorithms for computing these incomplete factorizations are based on Gaussian elimination and do not parallelize well. Recently, a more parallel incomplete factorization algorithm was proposed by Chow and Patel [5], where the factors are computed iteratively. In this paper, we propose a new iterative approach that is based on alternating triangular solves of L and U. We develop two versions: ATS-ILU for a static sparsity pattern, and ATS-ILUT for a dynamic pattern (using thresholding). We show that this new method is similar to the fine-grained iterative ILU method by Chow but has the added advantage that it allows greater reuse of memory and is fully deterministic in parallel, meaning the results do not depend on scheduling. We evaluate the new method on several test matrices from the SuiteSparse collection and show that is competitive with current ILU methods.

Key words. preconditioner, incomplete factorization, ILU, parallel, least squares

1. Introduction. Preconditioning is well known to be essential for improving the speed of convergence of Krylov methods such as Conjugate Gradient (CG) and Generalized Minimal Residual (GMRES) [3, 11, 19, 20]. Incomplete Lower-Upper (ILU) factorizations are a popular class of preconditioners as they can be used as a "black box" on a wide range of problems. There are two main types of ILU factorizations, level-based ILU(k) [3, 16, 19] and threshold-based ILUT [18]. However, these methods are inherently sequential and do not parallelize well.

There has been interest in the parallelization of these more classical interpretations of ILU, largely through graph partitioning schemes. These graph partition-based methods, such as [8, 12, 13, 15], offer a promising approach to parallelizing classical ILU methods. By decomposing the graph corresponding to the matrix and determining variables that can be eliminated in parallel, these methods aim to distribute the computational load more evenly across processors. While these strategies have shown effectiveness for certain types of problems [3], their implementation can be highly complex. Additionally, their performance can be problem-dependent, requiring consideration of the underlying graph structure when choosing a parallelization strategy.

More recently, there have been strides into methods of computing ILU factors iteratively, potentially giving up some of the robustness of the classical methods for better parallel properties [5]. Iterative ILU methods, such as those introduced by Chow [5], offer significant advantages in terms of scalability on modern parallel architectures. For the remainder of this paper, we refer to the method introduced by Chow as ParILU and its thresholded counterpart as ParILUT [1, 5]. These methods approximate the ILU factors through a series of iterative updates, which can be more easily distributed across multiple processors or offloaded to accelerators.

One of the primary benefits of iterative ILU methods is their ability to handle larger and more complex systems efficiently and be utilized as a true "black box" preconditioner by domain scientists. By breaking down each iterative update into smaller approximate subproblems and solving them independently, different parts of the factorization can be computed in parallel without the need for complex graph-partitioning algorithms. This

*Purdue University, mtunnell@purdue.edu
†Sandia National Labs, egboman@sandia.gov

approach allows for the use of iterative ILU methods on a wide range of problems, including those with complex or irregular graph structures that may preclude high levels of parallelism in the graph-partitioned classical ILU methods.

Furthermore, iterative ILU methods are adaptable to various hardware accelerators such as graphics processing units (GPUs) [2], which are increasingly important for high-performance computing. By leveraging the parallel processing capability of these accelerators, iterative ILU methods can significantly reduce the real-world time required to compute the ILU factors for large-scale problems, thereby speeding up the overall solution process.

In this paper, we propose a new class of iteratively-computed ILU preconditioners, which we call Alternating triangular Solves ILU (ATS-ILU). This method builds upon the strengths of existing iterative ILU approaches while leveraging improved memory reuse and determinism in parallel. We provide an analysis of the method and evaluate its performance compared to the state of the art on a variety of test matrices. We show that our method is competitive with current ILU methods and has the potential to be a powerful tool for solving large-scale problems on modern parallel architectures. We implement our algorithm in Julia [4, 17], as well as Kokkos [9, 22], and utilize the Kokkos ecosystem [21] for their implementations of ParILUT and GMRES.

This paper is organized as follows. First, we gave an introduction to classical ILU methods as well as their iterative counterparts in section 1. Next, we present needed background information in section 2, including an introduction of our notation, a review of a classical ILU method, and the fine-grained ILU method by Chow [5]. In section 3, we introduce our new class of iteratively computed ILU preconditioners based on alternating triangular solves and discuss its relation to ParILU. The algorithms for ATS-ILU and ATS-ILUT are presented in subsection 3.3 and subsection 3.5, respectively. In section 4, we provide implementation details for our new method. In section 5, we discuss the parallelization of our method. We evaluate the performance of our method on a variety of test matrices in section 6. Finally, we discuss future work and concluding remarks in section 7.

2. Background Information. We start this section by describing the notation used in this paper. We use largely standard notation but introduce some new notation for the sake of clarity. We use $\boldsymbol{A}$ to denote a matrix, $\boldsymbol{A}^T$ to denote its transpose, $\boldsymbol{a}$ to denote a vector, $\boldsymbol{I}$ to refer to the identity matrix where the size is determined based on context, and $\boldsymbol{L}$ and $\boldsymbol{U}$ to reference sparse matrices that are lower and upper triangular in structure. We subscript these matrices with the row and column indices separated by a comma. The notation $a_{i,j}$ refers to a scalar entry in the matrix $\boldsymbol{A}$ at row i and column j, while $\boldsymbol{a}_{i,:}$ refers to a row vector of $\boldsymbol{A}$ at row i. We utilize MATLAB [14] style slicing notation, where $\boldsymbol{a}_{i,j:k}$ refers to the elements of $\boldsymbol{a}_{i,:}$ at the column indices j through k. We additionally use non-contiguous slicing, where $\boldsymbol{a}_{i,\text{idx}}$ refers to the elements of $\boldsymbol{a}_{i,:}$ at the column indices in idx. Non-contiguous submatrices may also be referenced in a similar manner, where $\boldsymbol{A}_{\text{idx}_1,\text{idx}_2}$ refers to the submatrix of $\boldsymbol{A}$ whose entries correspond to the cartesian product of idx_1 and idx_2. We assume that slicing with a set of indices is performed in the natural order of the indices in the set. In the case of the cartesian product, a topological ordering is used where it is first ordered by the row index and then by the column index.

We use $\mathbb{N}$ to denote the set of natural numbers, which we define to start at 1, and $\mathbb{R}$ to denote the set of real numbers. The notation $\mathbb{N}^2$ refers to the set of pairs of natural numbers, and $\mathbb{R}^{n\times m}$ refers to the set of $n \times m$ matrices with real entries. We use $\boldsymbol{A}^k$ to refer to the k^{th} power of the matrix $\boldsymbol{A}$, whereas $\boldsymbol{L}^{(k)}$ refers to the k^{th} iteration of a method applied to $\boldsymbol{L}$. We are consistent with our use of 1-based indexing throughout this paper and assume all loops are inclusive of their endpoints unless otherwise stated.

We use notation like $\mathcal{L}$ to denote a graph of the sparsity pattern of $\boldsymbol{L}$ and are consistent

with our use of $\mathcal{S}$ to denote some arbitrary initial sparsity pattern. We use standard notation to define operations on a graph, where $\mathcal{L}\backslash\mathcal{U}$ refers to the graph $\mathcal{L}$ with the edges that exist in $\mathcal{U}$ removed. Similarly, $\mathcal{L}\cup\mathcal{U}$ refers to the graph $\mathcal{L}$ with the edges that exist in $\mathcal{U}$ added. We assume that the edges in a graph are uniquely defined by a set and often use set notation to define the entries of an adjacency matrix that fully describes the edges of a graph.

Next, we review a classical implementation of ILU, the sparse triangular solve algorithm, and the fine-grained ILU method by Chow [5].

2.1. Classical ILU Method. The classical ILU(k) method is based on Gaussian elimination, but with entries dropped to avoid or reduce the amount of fill-in, and given a pattern, $\mathcal{S}$, of non-zero entries in the factorization. These non-zero locations, in the case of an ILU(0) factorization, are often chosen to be the same as the non-zero pattern of the matrix, $\boldsymbol{A}$, but this is not required. For higher levels of fill, a common choice is to use the sparsity pattern of $\boldsymbol{A}^{k+1}$. This can be seen as an approximation to the true LU factors of $\boldsymbol{A}$, where the equation $\boldsymbol{LU} = \boldsymbol{A}$ is satisfied exactly on the pattern of $\boldsymbol{A}$ and may be violated elsewhere.

We show this algorithm given a fixed non-zero pattern, $\mathcal{S}$, in Algorithm 1. This algorithm computes the ILU factors by iterating through the rows and columns of the input matrix $\boldsymbol{A}$, updating the non-zero entries according to the sparsity pattern $\mathcal{S}$. For each row i, the algorithm computes the entries in the k^{th} column of $\boldsymbol{L}$ by scaling the entries in the k^{th} column of $\boldsymbol{U}$ by the diagonal entry $u_{k,k}$. This algorithm is inherently sequential as the updates to the factors are dependent on the previous row or column.

Algorithm 1 Classic ILU

Input: Sparse matrix $\boldsymbol{A}$, sparsity pattern $\mathcal{S}$.
Output: Sparse ILU factors $\boldsymbol{L}$ and $\boldsymbol{U}$, where $\boldsymbol{LU} = \boldsymbol{A}$ along the sparsity pattern $\mathcal{S}$.
$\boldsymbol{L} \leftarrow \boldsymbol{I}$
$\boldsymbol{U} \leftarrow \boldsymbol{A}$ where $u_{i,j} = 0$ if $(i,j) \notin \mathcal{S}$
for $j = 1$ **to** $n-1$ **do**
  for $i = j+1$ **to** n and $(i,j) \in \mathcal{S}$ **do**
    $\ell_{i,j} \leftarrow u_{i,j}/u_{j,j}$
    for $k = i$ **to** n and $(i,k) \in \mathcal{S}$ **do**
      $u_{i,k} \leftarrow u_{i,k} - \ell_{i,j} \cdot u_{j,k}$
    end for
  end for
end for

2.2. Fine-Grained Iterative ILU Method by Chow. Like the classical ILU method, ParILU computes an approximation to the true LU factors of $\boldsymbol{A}$ but does so iteratively using a series of smaller, approximate subproblems. Along the given sparsity pattern $\mathcal{S}$, the ParILU method updates the non-zero entries of the factors $\boldsymbol{L}$ and $\boldsymbol{U}$ by relaxing each variable independently. This independent relaxation is performed in parallel, allowing for better handling of large-scale and complex systems, with the potential for significant speedup on modern parallel architectures. This algorithm is given in Algorithm 2.

Each update can be performed in parallel, as the updates to each row or column are independent. In application, the algorithm can either be implemented in a manner that is deterministic in parallel or a non-deterministic manner. In implementing the non-deterministic version, the updates are performed in an atomic fashion directly to the factors $\boldsymbol{L}$ and $\boldsymbol{U}$. In the deterministic version, the updates are performed directly to the matric $\boldsymbol{L}$ but are stored in a temporary matrix for $\boldsymbol{U}$.

Algorithm 2 ParILU [5]

Input: Sparse matrix $\boldsymbol{A}$, sparsity pattern $\mathcal{S}$, starting factors $\boldsymbol{L}$ and $\boldsymbol{U}$
Output: Updated factors $\boldsymbol{L}$ and $\boldsymbol{U}$
while not converged **do**
  for $(i, j) \in \mathcal{S}$ **do**
    if $i > j$ **then**
      $\ell_{ij} \leftarrow a_{ij} - \sum_{k=1}^{j-1} \ell_{ik} u_{kj}$
    else
      $u_{ij} \leftarrow \left(a_{ij} - \sum_{k=1}^{i-1} \ell_{ik} u_{kj}\right) / \ell_{ii}$
    end if
  end for
end while

3. Alternating Triangular Solves Method. In this section, we introduce our new method for computing ILU factors, ATS-ILU. This method is based on the idea of alternating iterative updates to the $\boldsymbol{L}$ and $\boldsymbol{U}$ factors of the matrix $\boldsymbol{A}$. The basic idea is the same as before, where we iteratively update the factors $\boldsymbol{L}$ and $\boldsymbol{U}$ until convergence, but where the updates are performed in an alternating manner. This general process is a common method for solving bilinear systems and is outlined in Algorithm 3.

Algorithm 3 Alternating ILU

$\boldsymbol{U}^{(0)} \leftarrow \texttt{triu}(\boldsymbol{A})$
$k \leftarrow 0$
while not converged **do**
  Solve $\boldsymbol{L}^{(k)}\boldsymbol{U}^{(k)} \approx \boldsymbol{A}$ for $\boldsymbol{L}^{(k)}$
  Solve $\boldsymbol{L}^{(k)}\boldsymbol{U}^{(k+1)} \approx \boldsymbol{A}$ for $\boldsymbol{U}^{(k+1)}$
  Check convergence
  $k \leftarrow k + 1$
end while

One way to perform this procedure would be to perform Algorithm 4 with the entirety of $\boldsymbol{U}^{(k)}$ and let $\boldsymbol{A}$ be the right-hand side vector to solve for $\boldsymbol{L}^{(k+1)}$, and similar to solve for $\boldsymbol{U}^{(k+1)}$. This entire process can largely be performed in parallel as each row of $\boldsymbol{L}$ and column of $\boldsymbol{U}$ can be solved independently. Despite the potential for high levels of parallelism, it is still extremely computationally expensive and likely suffers from significant levels of fill-in during intermediate steps. The computational cost could be reduced by using an approximation. A simple approximation is the Neumann series: $(\boldsymbol{I}-\boldsymbol{L})^{-1} = \boldsymbol{I}+\boldsymbol{L}+\boldsymbol{L}^2+\ldots$ This has been explored [2] in the context of solving for dense vectors. Sparsity may be preserved to some degree, but the fill will increase with summing higher powers. Therefore, we do not consider this option any further.

Additionally, the algorithm as stated above does not guarantee that $\boldsymbol{L}$ and $\boldsymbol{U}$ remain lower and upper triangular, respectively. One method to address this issue would be to solve for $\boldsymbol{L}$ only in the lower triangular part of $\boldsymbol{A}$ and for $\boldsymbol{U}$ only in the upper triangular part of $\boldsymbol{A}$. This would ensure that the factors remain lower and upper triangular, respectively, but would still leave the problem of significant levels of fill-in. Instead, we suggest a more practical approach where we impose a sparsity pattern on $\boldsymbol{L}$ and $\boldsymbol{U}$, namely $\mathcal{L}$ and $\mathcal{U}$, respectively. This sparsity pattern can be chosen to be the same as the sparsity pattern of $\boldsymbol{A}$, which is the choice we make in this paper.

In order to get around the issue of fill-in, we propose a method where we approximately

solve for $\boldsymbol{L}$ and $\boldsymbol{U}$ along their given sparsity patterns, which we discuss next.

3.1. Sparse Triangular Solve. Because we introduce an approximate triangular solve in the following section, we provide a brief overview of the classical sparse triangular solve algorithm for reference in Algorithm 4. This algorithm is used to solve a lower triangular system of equations, where the solution vector $\boldsymbol{x}$ is updated in a forward sweep. The algorithm performs an exact inversion of the sparse lower triangular matrix $\boldsymbol{L}$ that has sparsity pattern $\mathcal{L}$. Although it is accurate, it is computationally expensive and does not parallelize particularly well. Therefore, it is only suitable for fairly small matrices.

Algorithm 4 Sparse Triangular Solve: Column-by-Column Elimination (Lower Triangular)

procedure TRIANGULARSOLVE($\boldsymbol{L}, \mathcal{S}, \boldsymbol{b}$)
  Input: Lower triangular matrix $\boldsymbol{L}$ of size $n \times n$, sparsity pattern $\mathcal{S}$, vector b of size n
  Output: Solution vector $\boldsymbol{x}$ of size n
  $\boldsymbol{x} \leftarrow$ **copy**($\boldsymbol{b}$)
  for $j = 1$ **to** n **do**
    $x_j \leftarrow x_j / \ell_{j,j}$
    for $i = j + 1$ **to** n and $(i, j) \in \mathcal{S}$ **do**
      $x_i \leftarrow x_i - x_j \cdot \ell_{i,j}$
    end for
  end for
  return x
end procedure

3.2. Approximate Sparse Triangular Solve. The exact sparse triangular solve may introduce fill. We want an approximate solve with no fill. Our approximate sparse triangular solve algorithm is given in Algorithm 5.

Algorithm 5 Sparse Triangular Solve: Column-by-Column Elimination (Approximate)

procedure APPROXIMATETRIANGULARSOLVE($\boldsymbol{L}, \mathcal{L}, \boldsymbol{b}, \mathcal{B}$)
  Input: Lower triangular matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$, sparsity pattern $\mathcal{L}$, vector $\boldsymbol{b} \in \mathbb{R}^n$, and pattern $\mathcal{B}$ along which to approximate the application of the inverse to $\boldsymbol{b}$
  Output: Solution vector $\boldsymbol{x}$ of size n
  $\boldsymbol{x} \leftarrow$ **copy**($\boldsymbol{b}$)
  for $j = 1$ **to** n and $j \in \mathcal{B}$ **do**
    $x_j \leftarrow x_j / \ell_{j,j}$
    for $i = j + 1$ **to** n and $i \in \mathcal{B}$ and $(i, j) \in \mathcal{L}$ **do**
      $x_i \leftarrow x_i - x_j \cdot \ell_{i,j}$
    end for
  end for
  return x
end procedure

Typically, the right hand side $\boldsymbol{b}$ is sparse and $\mathcal{B}$ is the sparsity pattern of $\boldsymbol{b}$. Our use case is when $\boldsymbol{b}$ is a column of $\boldsymbol{U}$. The special case of a sparse solve with a sparse right hand side has been well studied [6, 10], and the sparsity in $\boldsymbol{x}$ is determined by the reachability set. However, as we wish to preserve sparsity, we instead impose that the sparsity pattern of $\boldsymbol{x}$ shall be the same as that of $\boldsymbol{b}$. The intuition behind our method is to extract the nonzero

parts of $\boldsymbol{b}$ and solve for the corresponding submatrix of $\boldsymbol{L}$, though the implementation is slightly different to avoid extracting such a submatrix. This results in an approximate solve that preserves sparsity.

We describe the ATS-ILU algorithm next.

3.3. Alternating Triangular Solves ILU(k) Algorithm. The ATS-ILU(k) algorithm is based on the idea of approximately solving for $\boldsymbol{L}$ and $\boldsymbol{U}$ in an alternating fashion along only their given sparsity patterns. Again, the rows of $\boldsymbol{L}$ and the columns of $\boldsymbol{U}$ can be solved independently, allowing for a high level of parallelism. The algorithm is shown in Algorithm 6. We present the algorithm for a general pattern $\mathcal{S}$ but in practice, this will correspond to the pattern of $\boldsymbol{A}^k$ for some small power k.

Algorithm 6 ATS-ILU(k)

Input: Sparse matrix $\boldsymbol{A}$, sparsity pattern $\mathcal{S}$, starting factors $\boldsymbol{L}$ and $\boldsymbol{U}$
while not converged **do**
  for $i \in \{1\ 2\ \ldots\ n\}$ **do**
    $\text{idx} \leftarrow \{j \in \mathbb{N} \mid (i,j) \in \mathcal{S}, j \leq i\}$
    $\boldsymbol{\ell}_{i,\text{idx}} \leftarrow \boldsymbol{a}_{i,idx} \left(\boldsymbol{U}_{\text{idx,idx}}\right)^{-1}$
  end for
  for $j \in \{1\ 2\ \ldots\ n\}$ **do**
    $\text{idx} \leftarrow \{i \in \mathbb{N} \mid (i,j) \in \mathcal{S}, i \geq j\}$
    $\boldsymbol{u}_{\text{idx},j} \leftarrow \left(\boldsymbol{L}_{\text{idx,idx}}\right)^{-1} \boldsymbol{a}_{\text{idx},j}$
  end for
end while

In this algorithm, we solve for each row of $\boldsymbol{L}$ and each column of $\boldsymbol{U}$ independently. Recall from section 2 that the notation $\boldsymbol{a}_{i,idx}$ refers to the i^{th} row of $\boldsymbol{A}$ restricted to the indices in idx, and similarly for $\boldsymbol{U}_{\text{idx,idx}}$ and $\boldsymbol{L}_{\text{idx,idx}}$. These submatrices can be viewed as the (dense) non-contiguous submatrices of $\boldsymbol{L}$ and $\boldsymbol{U}$ that correspond to the sparsity pattern $\mathcal{S}$ along the given row or column. We show an example of this in Figure 3.1.

We note that solving for the solution of these non-contiguous submatrices and vectors can either be viewed as an exact solution of an approximate problem or an approximate solution of an exact problem. The former applies the inverse of the dense submatrix to the right-hand side vector, while the latter utilizes the approximate solve algorithm shown in Algorithm 5. These two views are equivalent in the sense that they both result in the same solution vector.

3.4. Relation between ATS-ILU and ParILU. We again note that the ATS-ILU(k) algorithm has some similarities to the ParILU algorithm by Chow, but there are some key differences. Both algorithms perform iterative updates to the $\boldsymbol{L}$ and $\boldsymbol{U}$ factors of the matrix $\boldsymbol{A}$ until convergence, both algorithms seek to solve the equation $\boldsymbol{LU} = \boldsymbol{A}$ approximately, and both algorithms can be implemented in a parallel fashion. However, in Chow's method, a single inner product is used to relax one value at a time, while our method is coarser-grained, as we operate on an entire row (column) at a time.

A second difference is that ParILU is asynchronous and updates $\boldsymbol{L}$ and $\boldsymbol{U}$ simultaneously, while we alternate between $\boldsymbol{L}$ and $\boldsymbol{U}$ updates. However, there is a close relationship. Suppose the execution order in Chow's method is to update first the lower triangular part, then the upper part. In this case, ParILU becomes synchronous and very similar to ATS-ILU. Conversely, one could modify ATS-ILU to update $\boldsymbol{L}$ and $\boldsymbol{U}$ simultaneously (asynchronously) but we do not explore that here.

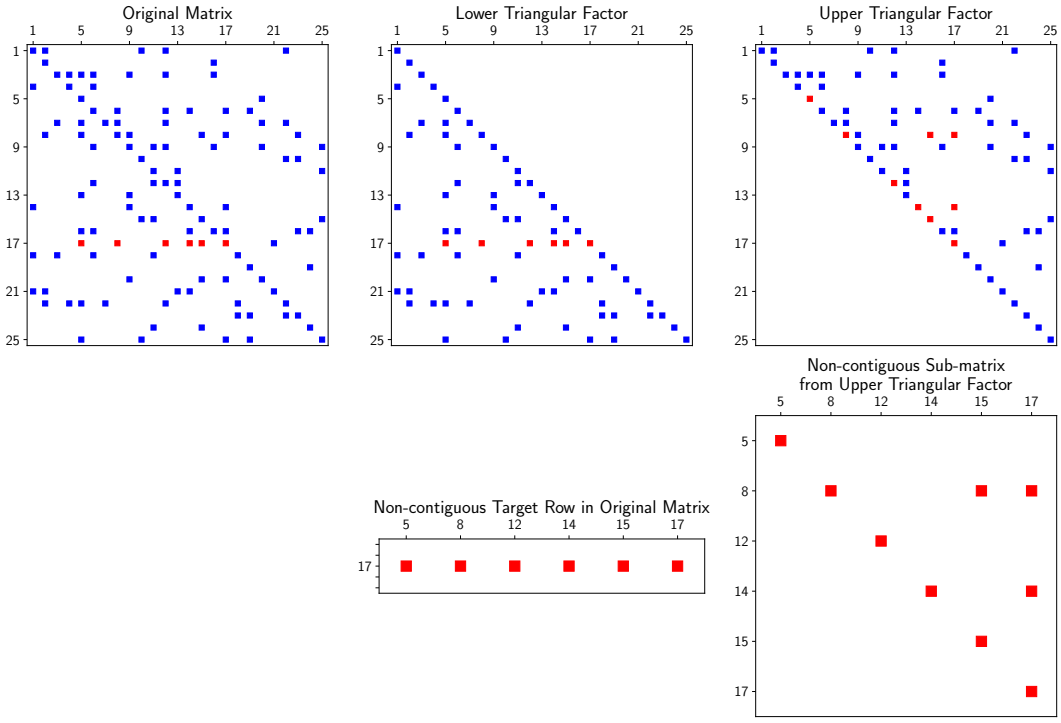


FIG. 3.1. *This figure illustrates the extraction of the small non-contiguous submatrix from* $\boldsymbol{U}$ *given row* 17 *from* $\boldsymbol{L}$*. Given the non-zero index locations of the given row, we extract the non-contiguous submatrix from* $\boldsymbol{U}$ *that corresponds to these columns and rows. Similarly, we extract the right-hand side vector from* $\boldsymbol{A}$ *that corresponds to these columns at the given row.*

We claim ATS-ILU uses exactly the same number of flops as ParILU per sweep (or update iteration). This can be seen from Figure 3.1. Consider the flops needed to update row 17 in $\boldsymbol{L}$. In ATS-ILU, this is given by the number of nonzeros in the small submatrix shown in the bottom right. These correspond to the red nonzeros in $\boldsymbol{U}$. On the other hand, ParILU requires sparse inner products between row 17 of $\boldsymbol{L}$ and columns 1 to 17 of $\boldsymbol{U}$. Interestingly, this corresponds exactly to the same red nonzeros. Therefore, the number of flops is the same.

Note that even if the flop count is the same, the methods are different and will typically produce different factors. For example, ATS-ILU updates the diagonal of $\boldsymbol{L}$ so it will typically not be unit while ParILU strictly enforces unit diagonal.

We believe our method is likely more memory efficient as the memory access pattern is more regular. Also, we avoid the sparse inner products, which are difficult to implement efficiently.

3.5. Alternating Triangular Solves Algorithm with Thresholding (ILUT). In this subsection, we describe the ATS-ILUT algorithm, which is a thresholded variant of the ATS-ILU algorithm that allows for a variable sparsity pattern based on the level of fill-in.

The variable sparsity pattern utilized in algorithm is based on the one used in the ParILUT method [1, 2]. After testing, we found that the "candidate fill-in" method used in ParILUT was highly competitive and we utilize something similar. This method is based on creating a new sparsity pattern based on the sparsity pattern of the residual matrix,

$\boldsymbol{R} = \boldsymbol{A} - \boldsymbol{LU}$ [1]. Like in [1], candidate locations for fill-in are chosen from

$$\mathcal{C} \leftarrow \mathcal{R} \backslash (\mathcal{L} \cup \mathcal{U}),$$

where $\mathcal{R}$ is the sparsity pattern of $\boldsymbol{R} = \boldsymbol{A} - \boldsymbol{LU}$, and $\mathcal{L}$ and $\mathcal{U}$ are the sparsity patterns of $\boldsymbol{L}$ and $\boldsymbol{U}$, respectively. The lower and upper triangular candidate locations are added to the sparsity pattern of $\boldsymbol{L}$ and $\boldsymbol{U}$, respectively, prior to updating the factors in each iteration and the original locations of $\boldsymbol{L}$ and $\boldsymbol{U}$ are retained. This method allows for a variable sparsity pattern that can be adjusted based on the level of fill-in in the factors $\boldsymbol{L}$ and $\boldsymbol{U}$.

At the end of each iteration, the factors $\boldsymbol{L}$ and $\boldsymbol{U}$ are thresholded to remove elements with a magnitude below a certain threshold. This threshold is chosen to be the k^{th} largest element in the factors $\boldsymbol{L}$ and $\boldsymbol{U}$, where k is the maximum number of elements allowed in $\boldsymbol{L}$ and $\boldsymbol{U}$. This method of thresholding is both simple and effective, and it reduces the number of variables the end user needs to tune in order to get a good approximation of the ILU factors. We note that this thresholding strategy is also used in the ParILUT method [1]. The full algorithm is given in Algorithm 7.

Algorithm 7 ATS-ILUT

1: **Input:** Sparse matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, starting factors $\boldsymbol{L}$ and $\boldsymbol{U}$, starting sparsity pattern $\mathcal{S}$, maximum number of elements $\boldsymbol{L}_{\max}$ and $\boldsymbol{U}_{\max}$ in $\boldsymbol{L}$ and $\boldsymbol{U}$, respectively
2: $\boldsymbol{R} \leftarrow (\boldsymbol{A} - \boldsymbol{LU})$
3: $\mathcal{L} \leftarrow \left\{(i,j) \in \mathbb{N}^2 \mid j \leq i, r_{i,j} \neq 0\right\}$
4: **for** $i \in \{1\ 2\ \ldots\ n\}$ **do**
5: idx $\leftarrow \{j \in \mathbb{N} \mid j \leq i, (i,j) \in \mathcal{L} \cup \mathcal{S}\}$
6: $\boldsymbol{\ell}_{i,\text{idx}} \leftarrow \boldsymbol{a}_{i,idx} \left(\boldsymbol{U}_{\text{idx,idx}}\right)^{-1}$
7: **end for**
8: $\boldsymbol{R} \leftarrow (\boldsymbol{A} - \boldsymbol{LU})$
9: $\mathcal{U} \leftarrow \left\{(i,j) \in \mathbb{N}^2 \mid j \geq i, r_{i,j} \neq 0\right\}$
10: **for** $j \in \{1\ 2\ \ldots\ n\}$ **do**
11: idx $\leftarrow \{i \in \mathbb{N} \mid j \geq i, (i,j) \in \mathcal{U} \cup \mathcal{S}\}$
12: $\boldsymbol{u}_{\text{idx},j} \leftarrow \left(\boldsymbol{L}_{\text{idx,idx}}\right)^{-1} \boldsymbol{a}_{\text{idx},j}$
13: **end for**
14: $\tau \leftarrow \boldsymbol{L}_{\max}$ rank element in $\{\left|\ell_{i,j} \mid (i,j) \in \mathbb{N}^2, \ell_{i,j} \neq 0\right|\}$
15: Threshold $\boldsymbol{L}$ to elements with larger magnitude than τ
16: $\tau \leftarrow \boldsymbol{U}_{\max}$ rank element in $\{\left|u_{i,j} \mid (i,j) \in \mathbb{N}^2, u_{i,j} \neq 0\right|\}$
17: Threshold $\boldsymbol{U}$ to elements with larger magnitude than τ

Note that on lines 2 and 8 of Algorithm 7, only the lower and upper portions of the residual are needed. As an implementation detail, one could compute half of the residual at each half step to reduce the total expended work by a significant margin. Next we briefly discuss options for the starting sparsity pattern.

3.6. Initial Guess and Sparsity Pattern. The initial guess for $\boldsymbol{L}$ and $\boldsymbol{U}$ make a difference. In fact, there is no guarantee that the alternating method converges to the global solution (though empirically it usually works).

There are several options for the starting sparsity pattern $\mathcal{S}$. One option is to use the sparsity pattern of the matrix $\boldsymbol{A}$, which was used in the ParILUT method [1]. Alternatively, one could use the sparsity pattern of the matrix $\boldsymbol{A}^k$ for some $k > 1$ or $k = 0$. We find it hard to justify using the sparsity pattern of $\boldsymbol{A}^k$ for $k > 1$. The sparsity pattern of $\boldsymbol{A}^0 = \boldsymbol{I}$ is the diagonal of $\boldsymbol{A}$, which is a reasonable choice. Later in section 6, we test the sparsity patterns of $\boldsymbol{A}^k$ for $k \in \{0, 1, 2\}$ and report our findings.

Next, we discuss important implementation details of the two algorithms.

4. Implementation details. This section is dedicated to the implementation details of the ATS-ILU and ATS-ILUT algorithms. For simplicity, we focus on how to solve for $\boldsymbol{L}$ given $\boldsymbol{U}$. How to compute $\boldsymbol{U}$ given $\boldsymbol{L}$ is analogous.

For reasons relating to the manner in which we iterate over $\boldsymbol{L}$ and $\boldsymbol{U}$, we store $\boldsymbol{L}$ in a compressed sparse row (CSR) format and $\boldsymbol{U}$ in a compressed sparse column (CSC) format. For this reason, we can treat $\boldsymbol{U}$ as the transpose of itself in CSR format, which transforms the solve in lines 6 and 20 of Algorithm 7 and line 5 of Algorithm 6 from

$$\boldsymbol{\ell}_{i,\mathrm{idx}} \leftarrow \boldsymbol{a}_{i,idx} \left(\boldsymbol{U}_{\mathrm{idx},\mathrm{idx}}\right)^{-1}$$

to

$$\boldsymbol{\ell}_{i,\mathrm{idx}} \leftarrow \left(\left(\boldsymbol{U}_{\mathrm{idx},\mathrm{idx}}\right)^{-T} \boldsymbol{a}_{idx,i}\right)^{T},$$

which allows us to use nearly the same algorithm for the update of both $\boldsymbol{L}$ and $\boldsymbol{U}$. In implementation, the only difference between the two algorithms is the manner in which we lookup the elements of $\boldsymbol{A}$ that we are solving for. If we were to pass $\boldsymbol{A}^T$ instead of $\boldsymbol{A}$ to the update routine when solving for $\boldsymbol{U}$, the implementation would be identical.

We implement the approximate sparse triangular solve algorithm shown in Algorithm 5 using a custom kernel that looks up values in $\boldsymbol{U}$ and $\boldsymbol{A}$ given the input indices from $\boldsymbol{L}$ via binary search. We call this kernel `binary_search_triangular_solve`. This implementation performs a lookup of the values in the non-contiguous submatrix of $\boldsymbol{U}$ and the right-hand side vector of $\boldsymbol{A}$. In the first pass of the triangular solve, the row values of $\boldsymbol{L}$ are repopulated with their corresponding elements in $\boldsymbol{A}$, and are treated as the non-contiguous right-hand side vector.

We implemented the ATS-ILUT algorithm with a large number of different variations. These different variations are controlled by options that conditionally compile different parts of the code using `constexpr` variables and `C`-style preprocessing. The options we implemented are as follows:

1. `USE_NEW`: This option utilizes the newest factor in the update step. i.e $\boldsymbol{L}^{(k+1)}$ is used to update $\boldsymbol{U}^{(k)}$ and similar.
2. `UPDATE_RESIDUAL_BETWEEN`: This option updates the residual matrix $\boldsymbol{R}$ between the two update steps.
3. `USE_RESIDUAL_LHS`: This option augments the left-hand side of the solve with the residual matrix. i.e for some modified matrix m $\boldsymbol{M}$, we let $m_{i,j} = u_{i,j}$ if $u_{i,j} \neq 0$ and $r_{i,j}$ otherwise.
4. `USE_RESIDUAL_RHS`: This option augments the right-hand side of the solve with the residual matrix. i.e given a solve for some row i of $\boldsymbol{L}$ we have $\boldsymbol{b}$, where $b_j = a_{i,j}$ if $a_{i,j} \neq 0$ and $r_{i,j}$ otherwise.
5. `ADDITIONAL_SWEEPS_BEFORE`: This option performs additional sweeps prior to the beginning of the main loop.
6. `ADDITIONAL_SWEEPS_AFTER`: This option performs additional sweeps after the thresholding step of the main loop.

We have exhaustively tested all combinations of these options (with options 5 and 6 limited to $\{0, 1\}$) on several matrices in the SuiteSparse collection. In section 6, we will discuss a selection of the results from these tests.

5. Parallel Aspects. A key advantage of ATS-ILU compared to the traditional ILU algorithm is that it is highly parallel. Thus, it is well suited for modern architectures, such as multicore CPU and accelerators such as GPU.

The ParILU method is also parallel, but not deterministic. The results will vary depending on the thread execution order. It is possible to impose a certain execution order, but this will slow the method down. In contrast, ATS-ILU is naturally deterministic. This is easy to see as $\boldsymbol{L}$ is used to update $\boldsymbol{U}$ and vice versa. It is also possible to update $\boldsymbol{L}$ and $\boldsymbol{U}$ simultaneously (asynchronously) but we did not explore that as we view the deterministic behavior as an advantage.

In ParILU, there is one thread per nonzero in the matrix. In ATS-ILU, the parallelism is over rows and columns. In our current implementation, one thread is assigned one row (or column), giving a coarser parallelism than in ParILU. However, another option is to assign a thread team for the sparse triangular solve in each row (column). This would require a team-level sparse triangular solve, which is not widely available. Thus, we consider this option future work.

6. Experiments and Results. We implement both the ATS-ILU and ATS-ILUT algorithms in C++ using the Kokkos library [22]. We test the algorithms on a variety of matrices from the SuiteSparse collection [7] and compare the results to the ParILUT algorithm. We note that we use the Kokkos implementation of ParILUT, which is a shared-memory parallel implementation of the ParILUT algorithm [1]. This implementation performs thresholding using the same method as the [1] implementation and not the fast approximate approach described in [2]. This is a valid comparison because we use the same thresholding strategy in our ATS-ILUT algorithm and the approximate one would be similarly applicable to our algorithm. All of the experiments given below were run on a single node processor with 24 threads and 128 GB of memory.

We compare the ATS-ILUT algorithm implementation against the ParILUT algorithm. In [2], they found their thresholding variant outperformed both their non-thresholded level-based variant but also often outperformed the classical level-based method. Because the results they obtained in their paper largely carry over to ours, we do not repeat that experiment in particular. Instead, we focus on the improved thresholding-based variants of ATS and ParILU.

Throughout this section, our measure of goodness is the number of GMRES iterations to converge to a relative residual of 10^{-10}. When referring to something as worse in terms of this measure, we mean that it required more iterations to converge. Conversely, when referring to something as better, we mean that it required fewer iterations to converge. We set the maximum subspace size of GMRES to 500 and the maximum number of iterations to 1500. We scale each matrix to have unit diagonal with Jacobi scaling. We use the GMRES implementation from the KokkosKernels library [22].

We tested 36 combinations of the options described in section 4 on all sparsity patterns stated above. We limited the two sweep-related options to either 0 or 1 additional sweeps. We share the results for the best two configurations below. Both of these configurations have `USE_NEW` and `UPDATE_RESIDUAL_BETWEEN` enabled, and `USE_RESIDUAL_LHS` and `USE_RESIDUAL_RHS` disabled. We found the performance of the algorithm to be largely unaffected by the `ADDITIONAL_SWEEPS_BEFORE` option but do notice a difference with the `ADDITIONAL_SWEEPS_AFTER` option. Because there was little difference in performing an initial sweep before the first iteration, we share the results with it disabled. The results for the two remaining configurations are given next in section 6. Note that "No Extra" refers to the configuration with `ADDITIONAL_SWEEPS_AFTER` disabled and "One Extra" refers to the configuration with `ADDITIONAL_SWEEPS_AFTER` set to 1. These configurations were tested on `atmosmodd`, `torso2`, `majorbasis`, and `venkat01` from the SuiteSparse collection [7].

Additionally, we tested with starting sparsity patterns of $\boldsymbol{A}^k$ for $k \in \{0, 1, 2\}$. The behavior with the starting sparsity pattern of $\boldsymbol{A}^0$ was similar, sometimes marginally better

but often marginally worse, than the behavior with the starting sparsity pattern of $\boldsymbol{A}^1$. Because the behavior is similar between these two patterns and because $\boldsymbol{A}^1$ was the starting pattern used in the ParILUT method, we do not share the results with the starting sparsity pattern of $\boldsymbol{A}^0$ in this paper.

The behavior with the starting sparsity pattern $\boldsymbol{A}^2$ was inconsistent and was largely uncompetitive with the other starting sparsity patterns. With this starting sparsity pattern, the preconditioner after the first update cycle was always worse than with the other starting patterns. In some cases, the resulting factors were better after the fifth update cycle, but this was not consistent across all matrices or even levels of fill for a given matrix. We discuss potential reasons for this behavior in section 7 but do not share the results in this paper due to space.

TABLE 6.1
Comparison of ATS-ILU Variants with PAR-ILUT across Different Matrices, Fill Levels, and Iterations with an $\boldsymbol{A}^1$ Starting Sparsity Pattern. The best at each iteration is bolded.

Matrices:		atmosmodd					torso2					majorbasis					venkat01				
Method	Fill	Iterations																			
		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
ATS-ILUT No Extra	1.0	166	150	147	146	146	9	8	7	8	7	**12**	**9**	10	**9**	10	**23**	**23**	**23**	**24**	25
	2.0	132	101	**96**	**95**	98	9	7	6	8	7	12	**8**	9	9	9	**17**	**14**	**13**	**12**	**12**
	3.0	132	95	**86**	**83**	**82**	9	7	6	8	7	12	**8**	9	9	9	**17**	**13**	**11**	**10**	**10**
ATS-ILUT One Extra	1.0	**147**	**145**	**145**	**145**	**145**	**7**	8	8	9	9	**12**	10	10	10	10	**23**	**23**	24	25	25
	2.0	**114**	**97**	97	97	98	**7**	8	8	9	9	**11**	9	9	10	10	**17**	**14**	**13**	**12**	**12**
	3.0	**114**	**89**	**86**	85	86	**7**	8	8	9	9	**11**	9	9	10	10	**17**	**13**	**11**	**10**	**10**
ParILUT	1.0	154	**145**	**145**	**145**	**145**	8	**7**	**6**	**6**	**6**	**12**	10	**9**	**9**	**9**	**23**	**23**	24	**24**	**24**
	2.0	120	98	97	97	**97**	8	**5**	**4**	**4**	**3**	12	**8**	**6**	**6**	**5**	19	15	**13**	**12**	**12**
	3.0	120	91	**86**	85	85	8	**5**	**4**	**4**	**3**	12	**8**	**6**	**5**	**5**	19	15	12	11	**10**

The results in section 6 show that the ATS-ILUT algorithm is competitive with the ParILUT algorithm. In most cases, the resulting factors after 5 update cycles are similar in performance to the ParILUT factors. However, the ATS-ILUT algorithm is often able to achieve a significantly better result after the first update cycle relative to ParILUT at the same update cycle.

Next, we show results for four more matrices. We test on `abnormal_sandia` an internal Sandia matrix that arises when solving a nonlinear thermal-fluid problem, as well as three more SuiteSparse matrices: `af_shell3`, `G3_circuit`, and `parabolic_fem`. This is given next in Table 6.

Similarly, these results largely show parity between the ATS-ILUT and ParILUT algorithms. ATS-ILUT with the "No Extra" configuration was often better at lower fill levels, while the "One Extra" configuration tended to converge to a good preconditioner faster.

We also tested these methods on the matrices `ecology2` and `offshore`. No algorithm produced a preconditioner that converged to the tolerance within the specified number of GMRES iterations for the `ecology2` matrix. For the `offshore` matrix, all methods produced a preconditioner that converged to the tolerance within the specified number of iterations if left to update for only one or two cycles. When updating for three cycles, all methods (all tested variants of ATS-ILUT and ParILUT) exhibited strange behavior and did not converge to the tolerance within the specified number of GMRES iterations. There was little difference in the number of GMRES iterations required to converge between the ATS-ILUT and ParILUT algorithms for the `offshore` matrix after the first or second update.

TABLE 6.2
Comparison of ATS-ILU Variants with PAR-ILUT across Different Matrices, Fill Levels, and Iterations with an $\boldsymbol{A}^1$ Starting Sparsity Pattern. The best at each iteration is bolded.

Matrices:		abnormal_sandia					af_shell3					G3_circuit					parabolic_fem				
Method	Fill	Iterations																			
		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
ATS-ILUT No Extra	1.0	54	**44**	**42**	**42**	**42**	905	723	594	**605**	**568**	1169	**1140**	**1148**	**1133**	**1131**	**1183**	**1089**	**1090**	**1083**	**1071**
	2.0	51	33	**25**	**24**	**23**	872	564	395	334	299	860	639	467	426	396	765	561	464	492	444
	3.0	51	31	22	**18**	**17**	872	559	378	308	258	860	638	448	373	318	765	551	422	434	381
ATS-ILUT One Extra	1.0	**50**	45	45	46	45	**797**	657	638	625	631	**1153**	1186	1189	1187	1183	1261	1192	1224	1220	1201
	2.0	**42**	**29**	27	27	27	**651**	**402**	319	289	274	**690**	**520**	408	424	414	**729**	719	505	547	446
	3.0	**42**	**24**	**20**	20	20	**651**	**397**	**290**	**226**	**204**	**690**	**520**	**357**	326	305	**729**	715	681	525	378
ParILUT	1.0	54	45	45	45	45	822	**597**	**581**	616	592	1188	1170	1180	1215	1217	1232	1168	1190	1201	1197
	2.0	49	32	26	25	25	752	415	**311**	**279**	**268**	758	531	**390**	**365**	**360**	864	**482**	**379**	**353**	**354**
	3.0	49	30	21	**18**	**17**	752	415	293	234	**204**	758	531	379	**295**	**269**	864	**479**	**320**	**219**	**191**

7. Conclusions. In this paper we have introduced the ATS-ILU and ATS-ILUT algorithms based on alternating inexact triangular solves. The ATS-ILU algorithm is a novel approach to computing ILU(k)-type factors that is deterministic and parallel. The ATS-ILUT algorithm is a thresholded variant of the ATS-ILU algorithm that allows for a variable sparsity pattern based on the level of fill-in. We have shown that the ATS-ILU and ATS-ILUT algorithms are competitive with the ParILUT algorithm on a variety of matrices from the SuiteSparse collection as well as the abnormal Sandia matrix. Our algorithm is deterministic without a loss in performance, which is a significant advantage over the ParILU algorithm. Additionally, our algorithm has better memory reuse than the ParILU algorithm, which is important for performance on modern architectures.

In terms of the number of GMRES iterations to converge, there was relatively little difference in the performance between the ParILUT and ATS-ILUT algorithms after a few update cycles. One reason for this could be that both algorithms are successful in solving for factors that are nearly-optimal given the sparsity pattern. We note that both of our algorithms use a similar method for the addition of new candidate locations to the sparsity pattern at each iteration. Indeed, we found that utilizing a different starting sparsity pattern ($\boldsymbol{A}^2$) sometimes resulted in better factors ultimately, but not in a manner that was consistent across all matrices or levels of fill for a given matrix. We believe that there is further work to be done in two main domains: choice of starting sparsity pattern and the method for adding new candidate locations to the sparsity pattern. We believe that the choice of starting sparsity pattern is important and that the behavior with the $\boldsymbol{A}^2$ starting sparsity pattern provides some empirical evidence for this.

Even with our algorithm ultimately creating factors that are similarly effective to the ParILUT factors, we believe that there is still merit in further exploring the ATS-ILU and ATS-ILUT algorithms. Our algorithm produces factors that work as a decent preconditioner for GMRES with only a single update cycle (sweep). Thus, our method has an advantage when short setup time is important. For certain applications, such as, one in which the preconditioner needs to be updated frequently or if the matrix is changing rapidly, this could be a significant advantage. We believe that the ATS-ILU and ATS-ILUT algorithms are a promising new approach to computing thresholded ILU factors and that there is further work to be done in this area.

Acknowledgments. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC.,

a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

REFERENCES

[1] H. Anzt, E. Chow, and J. Dongarra, *Parilut—a new parallel threshold ILU factorization*, SIAM Journal on Scientific Computing, 40 (2018), pp. C503–C519, https://doi.org/10.1137/16M1079506, https://doi.org/10.1137/16M1079506.

[2] H. Anzt, T. Ribizel, G. Flegar, E. Chow, and J. Dongarra, *Parilut - a parallel threshold ilu for gpus*, in 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, May 2019, https://doi.org/10.1109/ipdps.2019.00033, http://dx.doi.org/10.1109/IPDPS.2019.00033.

[3] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, Journal of Computational Physics, 182 (2002), p. 418–477, https://doi.org/10.1006/jcph.2002.7176, http://dx.doi.org/10.1006/jcph.2002.7176.

[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A fresh approach to numerical computing*, SIAM Review, 59 (2017), pp. 65–98, https://doi.org/10.1137/141000671, https://epubs.siam.org/doi/10.1137/141000671.

[5] E. Chow and A. Patel, *"a fine-grained parallel ILU factorization"*, SIAM Journal on Scientific Computing, 37 (2015), pp. C169–C197.

[6] T. Davis, *Direcet Methods for Sparse Linear Systems*, SIAM, 2006.

[7] T. A. Davis and Y. Hu, *The university of florida sparse matrix collection*, ACM Transactions on Mathematical Software, 38 (2011), p. 1–25, https://doi.org/10.1145/2049662.2049663, http://dx.doi.org/10.1145/2049662.2049663.

[8] X. Dong and G. Cooperman, *A Bit-Compatible Parallelization for ILU(k) Preconditioning*, Springer Berlin Heidelberg, 2011, p. 66–77, https://doi.org/10.1007/978-3-642-23397-5_8, http://dx.doi.org/10.1007/978-3-642-23397-5_8.

[9] H. C. Edwards, C. R. Trott, and D. Sunderland, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, Journal of Parallel and Distributed Computing, 74 (2014), pp. 3202 – 3216, https://doi.org/https://doi.org/10.1016/j.jpdc.2014.07.003, http://www.sciencedirect.com/science/article/pii/S0743731514001257. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[10] J. R. Gilbert and T. Peierls, *Sparse partial pivoting in time proportional to arithmetic operations*, SIAM Journal on Scientific and Statistical Computing, 9 (1988), pp. 862–874, https://doi.org/10.1137/0909058, https://doi.org/10.1137/0909058.

[11] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 49 (1952), p. 409, https://doi.org/10.6028/jres.049.044, http://dx.doi.org/10.6028/jres.049.044.

[12] D. Hysom and A. Pothen, *Efficient parallel computation of ilu(k) preconditioners*, in Proceedings of the 1999 ACM/IEEE conference on Supercomputing, SC '99, ACM, Jan. 1999, https://doi.org/10.1145/331532.331561, http://dx.doi.org/10.1145/331532.331561.

[13] P. Hénon and Y. Saad, *A parallel multistage ilu factorization based on a hierarchical graph decomposition*, SIAM Journal on Scientific Computing, 28 (2006), p. 2266–2293, https://doi.org/10.1137/040608258, http://dx.doi.org/10.1137/040608258.

[14] T. M. Inc., *Matlab version: 9.13.0 (r2022b)*, 2022, https://www.mathworks.com.

[15] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on Scientific Computing, 20 (1998), p. 359–392, https://doi.org/10.1137/s1064827595287997, http://dx.doi.org/10.1137/S1064827595287997.

[16] N. Li, Y. Saad, and E. Chow, *Crout versions of ilu for general sparse matrices*, SIAM Journal on Scientific Computing, 25 (2003), p. 716–728, https://doi.org/10.1137/s1064827502405094, http://dx.doi.org/10.1137/S1064827502405094.

[17] A. Montoison and D. Orban, *Krylov.jl: A Julia basket of hand-picked Krylov methods*, Journal of Open Source Software, 8 (2023), p. 5187, https://doi.org/10.21105/joss.05187.

[18] Y. Saad, *Ilut: A dual threshold incomplete lu factorization*, Numerical Linear Algebra with Applications, 1 (1994), p. 387–402, https://doi.org/10.1002/nla.1680010405, http://dx.doi.org/10.1002/nla.1680010405.

[19] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, Philadelphia, MS, 2 ed., 2003.

[20] Y. Saad and M. H. Schultz, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), p. 856–869, https://doi.org/10.1137/0907058, http://dx.doi.org/10.1137/0907058.

[21] C. Trott, L. Berger-Vergiat, D. Poliakoff, S. Rajamanickam, D. Lebrun-Grandie, J. Mad-

sen, N. Al Awar, M. Gligoric, G. Shipman, and G. Womeldorff, *The kokkos ecosystem: Comprehensive performance portability for high performance computing*, Computing in Science Engineering, 23 (2021), pp. 10–18, https://doi.org/10.1109/MCSE.2021.3098509.

[22] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, and J. Wilke, *Kokkos 3: Programming model extensions for the exascale era*, IEEE Transactions on Parallel and Distributed Systems, 33 (2022), pp. 805–817, https://doi.org/10.1109/TPDS.2021.3097283.

TENSOR PARAMETRIC OPERATOR INFERENCE WITH HAMILTONIAN STRUCTURE

ARJUN VIJAYWARGIYA*, SHANE A. MCQUARRIE†, AND ANTHONY GRUBER‡

Abstract. This work presents a tensor-based approach to constructing data-driven reduced-order models corresponding to semi-discrete partial differential equations. By expressing parameter-varying operators with affine dependence as contractions of a generalized parameter vector against a constant tensor, this method leverages the operator inference framework to capture parametric dependence in the reduced-order model via the solution to a convex, least-squares optimization problem. This leads to a simple and straightforward implementation which directly extends to learning parametric operators with symmetry constraints, a key feature required for constructing accurate surrogates of systems with a Hamiltonian structure. The method is demonstrated on a scalar heat equation with variable diffusion coefficient and a wave equation with variable wave speed.

1. Introduction. Mathematical models based on partial differential equations (PDEs) are often formed through the combination of terms describing locally distinct physical processes. For example, an advection-diffusion equation describing the evolution of a scalar quantity c on a domain $M \subset \mathbb{R}^n$, given by $\partial_t c = \nabla \cdot (D\nabla c - \boldsymbol{v}c)$ for appropriate tensor D and velocity $\boldsymbol{v}$, combines the term $\nabla \cdot (D\nabla c)$ expressing diffusion with a corresponding term $\boldsymbol{v} \cdot \nabla c$ describing material transport. As the evolution of many important physical phenomena can be described clearly and simply with differential operators, this provides a high degree of flexibility in phenomenological modeling, allowing the practitioner to construct compact and descriptive mathematical tools in a plug-and-play fashion. Importantly, the practical utility of PDE-based models typically relies on the calibration of a number of parameters ($D, \boldsymbol{v}$ in the example above), which are critical to the behavior of solutions and hence also to physical realism. Since these parameters are application dependent, this means that any technique for constructing reduced-order models (ROMs) must be amenable to preserving parametric structure.

In the case of intrusive, projection-based ROMs, this is well understood and straightforward to formulate: since the ROM equations are directly generated through Galerkin projection onto a data-driven reduced basis, any parametric dependence in the discrete operators at the full-order model (FOM) level will automatically be inherited at the ROM level. However, intrusive methods require direct manipulation of the governing equations and access to the underlying operators in the high-fidelity source code [1, 2, 6]. This may not always be possible since high-fidelity models often have complex code implementations which are not amenable to modifications. Manipulating these codes can be highly non-trivial and error-prone, or in some cases, entirely impossible due to licensing restrictions.

An alternative strategy is non-intrusive model reduction, where the operators governing the ROM must be inferred from data and not directly constructed through projection. However, in this case, a lack of access to the discrete FOM precludes the classical approach to building in parametric dependence, and it is less clear how to efficiently design an effective surrogate which approximates the Galerkin-optimal intrusive ROM. This makes powerful tools such as Operator Inference (OpInf) [15] less useful in their standard form, as their restriction to convex, least-squares regression will "average out" this dependence on parameters. Furthermore, while methods based on nonlinear regression can accommodate parametric dependence, they do not necessarily fare better in general due to their complicated optimization and propensity to converge to poor local minima [4, 8, 11].

*Division 01442, Sandia National Laboratories, avijayw@sandia.gov
†Division 01441, Sandia National Laboratories, smcquar@sandia.gov
‡Division 01442, Sandia National Laboratories, adgrube@sandia.gov

In view of this, the present work offers a simple and general way to incorporate parametric dependence into OpInf, retaining the advantages of the approach while allowing for parametric variability in the learned operators. The following simple observation is key: when a matrix operator $\boldsymbol{M}(\boldsymbol{\mu}) : \mathbb{R}^N \to \mathbb{R}^N$ depends linearly on a known function $\boldsymbol{\mu}' = \boldsymbol{\theta}(\boldsymbol{\mu}) \in \mathbb{R}^{p'}$ of the parameter vector $\boldsymbol{\mu} \in \mathbb{R}^p$, then $\boldsymbol{M}(\boldsymbol{\mu}) = \boldsymbol{T}\boldsymbol{\mu}'$ can be written as the contraction of a constant tensor $\boldsymbol{T} \in \mathbb{R}^{N\times N\times p'}$ against the transformed parameters $\boldsymbol{\mu}' \in \mathbb{R}^{p'}$. Said differently, the adjoint relationship $A \mapsto (B \mapsto C) \cong (A \otimes B) \mapsto C$ between continuous linear mappings and the tensor product yields a description of parametric dependence in terms of the linear action of a constant object. This notion is general enough to encompass a wide range of parametric behavior and will be shown to enable an OpInf learning problem which does not rely on local approximation techniques such as Taylor expansion [5] or linear interpolation [15]. The method proposed here reformulates the affine-parametric OpInf methods of [14, 21] with a tensorized description of the core learning problem, which leads to a concise theory and streamlined implementation. Our method also facilitates an important generalization to a class of systems where the preservation of certain structures is critical.

The contributions of this article are the following:

- A general, tensor-based approach to parametric OpInf which is simple to implement and recovers previous parametric OpInf work as a special case.
- An incorporation of symmetry constraints into the core learning problem, leading to an algorithm for Hamiltonian structure-preservation which guarantees symplecticity and conservative reduced dynamics.

The remainder of the work is structured as follows. Section 2 discusses the proposed tensor parametric approach in the structure-agnostic case, including necessary background, the core inference problem and its solution, followed by a demonstration of the approach on a 2-D heat equation with domain-varying diffusivity. Section 3 extends this approach to parametric systems with Hamiltonian structure, reviewing additional background on these systems, presenting the modified inference problem, and discussing its solution, defore demonstrating the approach on a 1-D wave equation with domain-varying wave speed. Finally, Section 5 offers some concluding remarks and avenues for future work.

2. Parametric ODEs.

2.1. Background. Given a vector $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$ of parameters, consider a linear parametric system of ordinary differential equations of the form

$$\dot{\boldsymbol{q}} = \boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{q} = (\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{q}, \;\; \boldsymbol{q}_0 = \boldsymbol{q}(0), \tag{2.1}$$

where $\boldsymbol{q} = \boldsymbol{q}(t, \boldsymbol{\mu}) \in \mathbb{R}^N$ is the ODE state, $\boldsymbol{A} : \mathbb{R}^{N_p} \to \mathbb{R}^{N\times N}$ is a matrix-valued function with affine parameter-dependence, so that $\boldsymbol{A}(\boldsymbol{\mu}) = \sum_{i=1}^{N_p} \mu_i \boldsymbol{A}_i$, and $\boldsymbol{T} \in \mathbb{R}^N \otimes \mathbb{R}^N \otimes \mathbb{R}^{N_p}$ is the order-3 tensor obtained from $\boldsymbol{A}$ by tensor-Hom adjunction. Such systems frequently arise from the semidiscretization of continuous PDEs through the "method of lines" [20]. An ODE system of the form (2.1) shall be referred to as a Full-Order Model (FOM) in the remainder of this paper as it is solved in its full dimensionality. A Reduced Order Model (ROM) can be obtained from (2.1) through a projection-based model reduction strategy [1]. Given a trial space $\boldsymbol{U} \in \mathbb{R}^{N\times r}$ and an approximate ansatz $\boldsymbol{q} = \tilde{\boldsymbol{q}}(t, \boldsymbol{\mu}) = \boldsymbol{U}\hat{\boldsymbol{q}}(t, \boldsymbol{\mu})$, with an unknown coefficient vector $\hat{\boldsymbol{q}} \in \mathbb{R}^r$, a straightforward Galerkin projection of the system (2.1) onto the span of $\boldsymbol{U}$ gives the reduced ODE system

$$\dot{\hat{\boldsymbol{q}}} = \boldsymbol{U}^\intercal \boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{U}\boldsymbol{q} = \boldsymbol{U}^\intercal(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{q}} \coloneqq \left(\hat{\boldsymbol{T}}\boldsymbol{\mu}\right)\hat{\boldsymbol{q}}, \;\; \hat{\boldsymbol{q}}_0 = \boldsymbol{U}^\intercal\boldsymbol{q}. \tag{2.2}$$

where $\hat{\boldsymbol{T}} \in \mathbb{R}^r \otimes \mathbb{R}^r \otimes \mathbb{R}^{N_p}$ is a reduced order-3 tensor with components

$$\hat{T}^a_{bx} = \sum_{i=1}^{N}\sum_{j=1}^{N} U^a_i T^i_{jx} U^j_b. \tag{2.3}$$

The system (2.2) provides a lower dimensional representation of the FOM (2.1) by only pertaining to the evolution of the reduced state vector $\hat{\boldsymbol{q}}$. If the trial basis $\boldsymbol{U}$ is properly constructed to capture the dominant modes of the system dynamics, the ROM can accurately recover the essential characteristics of the original system at a significantly reduced computational cost. The next section provides a detailed description of how a ROM like (2.2) can be constructed non-intrusively by directly inferring the tensor operator (2.3) directly data, without requiring an access to the underlying source code of the FOM.

2.2. Methodology. Now that the necessary background has been reviewed, it is prudent to explain how the tensors in (2.2) can be inferred. Notably, the tensor $\hat{\boldsymbol{T}}$ is independent of the reduced state $\hat{\boldsymbol{q}}$ and the parameters $\boldsymbol{\mu}$, a fact which will be crucial to the optimization strategy employed here. Precisely, the goal is to learn $\hat{\boldsymbol{T}}$ from data via a convex, least-squares regression. Before the inference procedure can be described, it is beneficial to review how a trial basis $\boldsymbol{U}$ needed for the Galerkin projection may be constructed. This is briefly outlined in the following subsection.

2.2.1. Proper Orthogonal Decomposition. This subsection briefly outlines a strategy to construct the trial basis matrix $\boldsymbol{U}$ through the technique of Proper Orthogonal Decomposition (POD). POD is a powerful and widely used tool to reduce the dimensionality of high-dimensional systems by identifying a lower dimensional subspace that retains the key dynamics. [3, 12, 17]. It can be implemented in the current setup as follows. Denote by $\boldsymbol{\mu}_s$ the s^{th} sampled vector of training parameters, where $1 \leq s \leq N_s$. Let $\boldsymbol{Q}_s = \left\{q^i(t_\alpha, \boldsymbol{\mu}_s)\right\}_{i,\alpha} \in \mathbb{R}^{N\times N_t}$ be the s^{th} matrix containing the full-order solution states collected at the N_t times $\{t_\alpha\}$. The matrix $\boldsymbol{Q}_s$ shall be referred to as a "snapshot matrix" corresponding to the parameter $\boldsymbol{\mu}_s$. Let $\boldsymbol{Y} \in \mathbb{R}^{N\times N_t N_s}$ denote the concatenation $[\boldsymbol{Q}_1\ \boldsymbol{Q}_2\ \cdots\ \boldsymbol{Q}_{N_s}]$ of the N_s snapshot matrices. A POD basis $\boldsymbol{U} \in \mathbb{R}^{N\times r}$ can now be constructed by computing the Singular Value Decomposition $\boldsymbol{Y} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\intercal$ such that $\boldsymbol{U}$ contains the first r left-singular vectors of $\boldsymbol{Y}$. The columns of $\boldsymbol{U}$ represent the dominant modes of the system dynamics, and the matrix can now be used to obtain a ROM via Galerkin projection of the FOM as described previously in (2.2).

2.2.2. Tensor Parametric Operator Inference. Let $\dot{\boldsymbol{Q}}_s$ represent the corresponding data matrix of the approximate time-derivatives obtained from $\boldsymbol{Q}_s$ using the fourth order finite-difference stencil given by[19]

$$\dot{\boldsymbol{q}}_n \approx \begin{cases} \frac{1}{\Delta t}\left(-\frac{25}{12}\boldsymbol{q}_n + 4\boldsymbol{q}_{n+1} - 3\boldsymbol{q}_{n+2} + \frac{4}{3}\boldsymbol{q}_{n+3} - \frac{1}{4}\boldsymbol{q}_{n+4}\right), & n = 0, \\ \frac{1}{\Delta t}\left(-\frac{1}{4}\boldsymbol{q}_{n-1} - \frac{5}{6}\boldsymbol{q}_n + \frac{3}{2}\boldsymbol{q}_{n+1} - \frac{1}{2}\boldsymbol{q}_{n+2} + \frac{1}{12}\boldsymbol{q}_{n+3}\right), & n = 1 \\ \frac{1}{\Delta t}\left(\frac{1}{12}\boldsymbol{q}_{n-2} - \frac{2}{3}\boldsymbol{q}_{n-1} + \frac{2}{3}\boldsymbol{q}_{n+1} - \frac{1}{12}\boldsymbol{q}_{n+2}\right), & n = 2, \ldots, N_t - 2 \\ \frac{1}{\Delta t}\left(-\frac{1}{12}\boldsymbol{q}_{n+1} + \frac{1}{2}\boldsymbol{q}_n - \frac{3}{2}\boldsymbol{q}_{n-1} + \frac{5}{6}\boldsymbol{q}_{n-2} + \frac{1}{4}\boldsymbol{q}_{n-3}\right), & n = N_T - 1 \\ \frac{1}{\Delta t}\left(\frac{25}{12}\boldsymbol{q}_n - 4\boldsymbol{q}_{n-1} + 3\boldsymbol{q}_{n-2} - \frac{4}{3}\boldsymbol{q}_{n-3} + \frac{1}{4}\boldsymbol{q}_{n-4}\right), & n = N_t, \end{cases} \tag{2.4}$$

where $\dot{\boldsymbol{q}}_n$ and $\boldsymbol{q}_n$ are the n-th columns of $\dot{\boldsymbol{Q}}$ and $\boldsymbol{Q}$. Let $||\cdot||$ denote the Frobenius norm. The reduced tensor $\hat{\boldsymbol{T}}$ in (2.2) can be inferred by solving the minimization problem

$$\arg\min_{\hat{\boldsymbol{T}}} \frac{1}{2}\sum_{s=1}^{N_s}\left|\left|\dot{\hat{\boldsymbol{Q}}}_s - \left(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{Q}}_s\right|\right|^2,$$

where the reduced matrices $\hat{\boldsymbol{Q}}_s, \dot{\hat{\boldsymbol{Q}}}_s \in \mathbb{R}^{r\times N_t}$ are obtained from $\boldsymbol{Q}_s$ and $\dot{\boldsymbol{Q}}_s$ as

$$\hat{\boldsymbol{Q}}_s = \boldsymbol{U}^\intercal \boldsymbol{Q}_s, \;\; \dot{\hat{\boldsymbol{Q}}}_s = \boldsymbol{U}^\intercal \dot{\boldsymbol{Q}}_s.$$

For convenience, let $\text{vec}_{ij}\hat{\boldsymbol{T}}$ denote the column-wise partial vectorization operator, which unrolls the adjacent indices i, j of the tensor $\hat{\boldsymbol{T}}$ into a single index with the same neighbors. The solution to the inference procedure for $\hat{\boldsymbol{T}}$ is outlined in the following theorem.

THEOREM 2.1. *Suppose $\hat{\boldsymbol{C}}_s \in \mathbb{R}^{r\times N_t}$ and $\hat{\boldsymbol{B}}_s \in \mathbb{R}^{r\times N_t}$. The unique solution to the least-squares minimization problem*

$$\arg\min_{\hat{\boldsymbol{T}}} \frac{1}{2}\sum_{s=1}^{N_s}\left|\left|\hat{\boldsymbol{C}}_s - \left(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s\right|\right|^2,$$

is given by the solution to the linear system

$$\hat{\boldsymbol{l}}\,\hat{\boldsymbol{t}}^\intercal = \hat{\boldsymbol{n}}^\intercal, \tag{2.5}$$

where $\hat{\boldsymbol{t}} \in \mathbb{R}^{r\times rN_p}$, $\hat{\boldsymbol{l}} \in \mathbb{R}^{rN_p\times rN_p}$, and $\hat{\boldsymbol{n}} \in \mathbb{R}^{r\times rN_p}$ denote the vectorized expressions:

$$\hat{\boldsymbol{t}} \coloneqq \text{vec}_{23}\,\hat{\boldsymbol{T}},$$

$$\hat{\boldsymbol{l}} \coloneqq \text{vec}_{12}\text{vec}_{34}\left(\sum_{s=1}^{N_s}\boldsymbol{\mu}_s\otimes\hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^\intercal\otimes\boldsymbol{\mu}_s\right),\;\; \hat{\boldsymbol{n}} \coloneqq \text{vec}_{23}\left(\sum_{s=1}^{N_s}\hat{\boldsymbol{C}}_s\hat{\boldsymbol{B}}_s^\intercal\otimes\boldsymbol{\mu}_s\right).$$

Proof. The proof is a direct calculation. Consider the Lagrangian

$$L\left(\hat{\boldsymbol{T}}\right) \coloneqq \frac{1}{2}\sum_{s=1}^{N_s}\left|\left|\left(\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s\right|\right|^2.$$

Differentiating with respect to $\hat{\boldsymbol{T}}$, it follows that

$$\begin{aligned} dL\left(\hat{\boldsymbol{T}}\right) &= \sum_{s=1}^{N_s}\left\langle\left(d\hat{\boldsymbol{T}}\,\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s, \left(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s - \hat{\boldsymbol{C}}_s\right\rangle \\ &= \left\langle d\hat{\boldsymbol{T}}, \sum_{s=1}^{N_s}\left[\left(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s - \hat{\boldsymbol{C}}_s\right]\hat{\boldsymbol{B}}_s^\intercal\otimes\boldsymbol{\mu}_s\right\rangle = \left\langle d\hat{\boldsymbol{T}}, \nabla L\left(\hat{\boldsymbol{T}}\right)\right\rangle, \end{aligned}$$

so that the stationarity condition $\nabla L\left(\hat{\boldsymbol{T}}\right) = \boldsymbol{0}$ implies

$$\sum_{s=1}^{N_s}\left(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^\intercal\otimes\boldsymbol{\mu}_s = \sum_{s=1}^{N_s}\hat{\boldsymbol{C}}_s\hat{\boldsymbol{B}}_s^\intercal\otimes\boldsymbol{\mu}_s.$$

Now, notice that the left-hand side can be expressed as

$$\sum_{s=1}^{N_s}\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^\intercal\otimes\boldsymbol{\mu}_s=\hat{\boldsymbol{T}}:\sum_{s=1}^{N_s}\boldsymbol{\mu}_s\otimes\hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^\intercal\otimes\boldsymbol{\mu}_s,$$

and therefore (2.5) follows directly after using the definitions of $\hat{\boldsymbol{t}}$, $\hat{\boldsymbol{m}}$, and $\hat{\boldsymbol{n}}$. ∎

Theorem 2.1 provides a tool for carrying out the basic tensor-based parametric OpInf presented here. In particular, note that the partial vectorization vec_{ij} has a natural inverse mat_{ij}, and therefore it is easy to form the desired tensor $\hat{\boldsymbol{T}}=\mathrm{mat}_{23}\,\hat{\boldsymbol{t}}$ once the matrix $\hat{\boldsymbol{t}}$ has been learned. Moreover, since $\hat{\boldsymbol{T}}$ is independent of the parameter vector $\boldsymbol{\mu}$, this inference can be carried out even in the case that $N_s=1$.

Remark 2.1. *Note that the entire calculation leading to Theorem 2.1 goes through unchanged if $\boldsymbol{\mu}$ is replaced by $\boldsymbol{\mu}'\coloneqq\boldsymbol{\theta}(\boldsymbol{\mu})\in\mathbb{R}^{\bar{N}_p}$ where $\boldsymbol{\theta}$ is some (known) nonlinear function of the parameters. The only difference in this case will be the dimension of the tensor $\hat{\boldsymbol{T}}$, which must be of size $\bar{N}_p$ in its last index.*

Remark 2.2. *We can turn $l(\mathrm{vec}\,\boldsymbol{T})$ around into standard OpInf form with more cumbersome tensors. Let $\boldsymbol{A}\in\mathbb{R}^n\otimes\mathbb{R}^{N_t}\otimes\mathbb{R}^{N_p}\otimes\mathbb{R}^{N_s}$ be the tensor with components $A^{bx}_{\alpha s}=\hat{Q}^b_{\alpha s}\mu^x_s$. Then, $l(\boldsymbol{O})=||\boldsymbol{D}\boldsymbol{O}^\intercal-\boldsymbol{R}^\intercal||^2$ where $\boldsymbol{D}=(\mathrm{vec}_{13}\mathrm{vec}_{24}\boldsymbol{A})^\intercal\in\mathbb{R}^{N_tN_s\times nN_p}$, $\boldsymbol{O}=\mathrm{vec}_{23}\boldsymbol{T}\in\mathbb{R}^{n\times nN_p}$, and $\boldsymbol{R}=\mathrm{vec}_{23}\dot{\hat{\boldsymbol{Q}}}\in\mathbb{R}^{n\times N_tN_s}$. As usual, this problem decouples over the rows of $\boldsymbol{O}$.*

2.3. Model Problem. This subsection describes how a full-order model (FOM) for a parametrized heat equation problem can be expressed in the tensor ODE form (2.1) and how the proposed model reduction and operator inference procedures can be applied to it. To that end, consider the following initial boundary value problem with Dirichlet boundary conditions

$$\begin{cases}\partial_t q(\boldsymbol{x},t)=c(\boldsymbol{\mu})\Delta q(\boldsymbol{x},t), & \boldsymbol{x}\in\Omega\times(0,t_f],\\ q(\boldsymbol{x},0)=q_0(\boldsymbol{x}), & \boldsymbol{x}\in\Omega\\ q(x,t)=0, & \boldsymbol{x}\in\partial\Omega\times(0,t_f],\end{cases}\tag{2.6}$$

where $c(\boldsymbol{\mu}):\mathbb{R}^{N_p}\to\mathbb{R}$ is the parameterized thermal conductivity coffecient given by

$$c(\boldsymbol{\mu})=\mu_1 1_{\Omega_1}+\mu_2 1_{\Omega_2}+\ldots+\mu_{N_p}1_{\Omega_{N_p}}.$$

Here, $\Omega=\bigcup_{i=1}^{N_p}\Omega_i$ is a decomposition into non-overlapping subdomains and 1_{Ω_i} denotes the indicator function:

$$1_{\Omega_i}(\boldsymbol{x})=\begin{cases}1, & \boldsymbol{x}\in\Omega_i,\\ 0, & \text{otherwise.}\end{cases}$$

A FOM for (2.6) can be constructed using a Continuous Galerkin discretization in space. Let $\Omega_h\coloneqq\{K_i\}_{i=1}^{N_E}$ be a conforming triangulation of the domain Ω containing N_E elements. The discretization takes the following form: find $\dot{q}_h\in V_h$ such that for all $v_h\in V_h$,

$$(\dot{q}_h,v_h)_{\Omega_h}=-(c(\boldsymbol{\mu})\nabla q_h,\nabla v_h)_{\Omega_h},\tag{2.7}$$

where $(\cdot,\cdot)_{\Omega_h}$ denotes the L^2 inner product on Ω_h and V_h is the H^1-conforming finite element space

$$V_h := \{v_h \in H^1(\Omega) : v_h|_K \in P_1(K),\ \forall K \in \Omega_h;\ v_h|_{\partial\Omega} = 0\},$$

with $P_1(K)$ representing the space of linear polynomials on element K. The FOM (2.7) can be put in a familiar matrix-form suitable to tensorization. Let $\{\phi_i(x)\}_{i=1}^N$ be the set of basis functions of the space V_h. By using the expansion $q_h = \sum_{i=1}^N q_i(t)\phi_i(x)$ and test function $v_h = \phi_j$ in (2.7), it follows that

$$\sum_{i=1}^{N_W}(\phi_i,\phi_j)_{\Omega_h}\dot{q}_i = -\sum_{i=1}^{N}(c(\boldsymbol{\mu})\nabla\phi_i,\nabla\phi_j)_{\Omega_h}q_i$$

which is equivalent to the matrix-vector form

$$\boldsymbol{M}\dot{\boldsymbol{q}} = -\boldsymbol{S}(\boldsymbol{\mu})\boldsymbol{q} \implies \dot{\boldsymbol{q}} = -\boldsymbol{M}^{-1}\boldsymbol{S}(\boldsymbol{\mu})\boldsymbol{q} = \boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{q}, \tag{2.8}$$

where $\boldsymbol{A}(\boldsymbol{\mu}) = -\boldsymbol{M}^{-1}\boldsymbol{S}(\boldsymbol{\mu})$, $\boldsymbol{q} = [q_1, q_2, \ldots, q_N]^\intercal \in \mathbb{R}^N$ is a vector containing the degrees of freedom of q_h, $\boldsymbol{M} \in \mathbb{R}^{N\times N}$ is a mass matrix with components

$$M_{i,j} = \sum_{i=1}^{N}(\phi_i,\phi_j)_{\Omega_h},$$

and $\boldsymbol{S} \in \mathbb{R}^{N\times N}$ is a parameter dependent stiffness matrix with components

$$S_{ij} = \sum_{i=1}^{N}(c(\boldsymbol{\mu})\nabla\phi_i,\nabla\phi_j)_{\Omega_h}.$$

Since the matrix $\boldsymbol{A}(\boldsymbol{\mu})$ has an affine parametric dependence, (2.8) can now be tensorized by replacing $\boldsymbol{A}(\boldsymbol{\mu})$ by the contraction $\boldsymbol{T}\boldsymbol{\mu}$. Model reduction can then be applied to the system as outlined in Algorithm 1.

Algorithm 1 Tensor Operator Inference without Structure (OpInf)

Input: Snapshot set $\{\boldsymbol{Q}_s\}_{s=1}^{N_s}$ of the FOM solution at times $\mathcal{T} = \{t_0, \ldots, t_{N_t}\}$, and reduced dimension $r > 1$.

Output: Reduced tensor operator $\hat{\boldsymbol{T}} \in \mathbb{R}^{r\times r\times N_p}$ in the ODE $\dot{\hat{\boldsymbol{q}}} = (\hat{\boldsymbol{T}}\boldsymbol{\mu})\boldsymbol{q}$ for $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$.

1: Compute the time derivative data $\{\dot{\boldsymbol{Q}}_s\}_{s=1}^{N_s}$ from $\{\boldsymbol{Q}_s\}_{s=1}^{N_s}$ using the finite difference formula (2.4).

2: Construct a reduced basis $\boldsymbol{U} \in V_r(\mathbb{R}^N)$ through a truncated SVD of the matrix $\boldsymbol{Y} = [\boldsymbol{Q}_1\ \ \ldots\ \ \boldsymbol{Q}_{N_s}]$.

3: Set the reduced snapshots $\hat{\boldsymbol{Q}}_s = \boldsymbol{U}^\intercal\boldsymbol{Q}_s$ and $\dot{\hat{\boldsymbol{Q}}}_s = \boldsymbol{U}^\intercal\dot{\boldsymbol{Q}}_s \in \mathbb{R}^{r\times N_t}$ for $s = 1, \ldots, N_s$

4: Solve for $\hat{\boldsymbol{T}}$ using (2.5) with $\boldsymbol{C}_s = \dot{\hat{\boldsymbol{Q}}}_s$ and $\boldsymbol{B}_s = \hat{\boldsymbol{Q}}_s$.

3. Parametric Hamiltonian ODEs.

3.1. Background.

Consider a linear parametric Hamiltonian system of the form

$$\dot{\boldsymbol{y}} = \boldsymbol{J}\nabla_{\boldsymbol{y}}H(\boldsymbol{y},\boldsymbol{\mu}) = \boldsymbol{J}\boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{y} = \boldsymbol{J}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y},\ \ \boldsymbol{y}_0 = \boldsymbol{y}(0), \tag{3.1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$ is the parameter vector, $\boldsymbol{y} = [\boldsymbol{q}\ \ \boldsymbol{p}]^\intercal \in \mathbb{R}^N$ with $N = 2M$ denotes the state vector, $\boldsymbol{q}$ and $\boldsymbol{p}$ in $\mathbb{R}^M$ are the canonical position and momentum variables, $H : \mathbb{R}^N \to \mathbb{R}$ is the Hamiltonian functional and $\boldsymbol{J} = -\boldsymbol{J}^\intercal \in \mathbb{R}^{N\times N}$ is a skew-symmetric symplectic matrix given by

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix}$$

Also, the matrix $\boldsymbol{A} : \mathbb{R}^{N_p} \to \mathbb{R}^{N\times N}$ is assumed to be symmetric so that $\boldsymbol{A}(\boldsymbol{\mu}) = \boldsymbol{A}(\boldsymbol{\mu})^\intercal$ (since $2\boldsymbol{x}^\intercal\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^\intercal(\boldsymbol{A} + \boldsymbol{A}^\intercal)\boldsymbol{x}$) and that it has an affine parametric dependence allowing for the decomposition $\boldsymbol{A}(\boldsymbol{\mu}) = \sum_{i=1}^{N_p} \mu_i \boldsymbol{A}_i$. The tensor $\boldsymbol{T} \in \mathbb{R}^N \otimes \mathbb{R}^N \otimes \mathbb{R}^{N_p}$ is a tensor obtained from $\boldsymbol{A}(\boldsymbol{\mu})$ via tensor-Hom adjunction which is symmetric in its first two indices so that $\boldsymbol{T}\boldsymbol{\mu} = (\boldsymbol{T}\boldsymbol{\mu})^\intercal$. An important property of the ODE system (3.1) is that the system Hamiltonian is conserved in time as can be seen in the following identity

$$\dot{H} = \dot{\boldsymbol{y}} \cdot \nabla H = \boldsymbol{J}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y} \cdot (\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y} = -(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y} \cdot \boldsymbol{J}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y} = 0,$$

where the skew-symmetry of $\boldsymbol{J}$ has been used.

Given a POD trial space $\boldsymbol{U} \in \mathbb{R}^{N\times 2r}$ and ansatz $\boldsymbol{y} \approx \tilde{\boldsymbol{y}} = \boldsymbol{U}\hat{\boldsymbol{y}}$, a ROM can be constructed from (3.1) through a Galerkin projection as

$$\dot{\hat{\boldsymbol{y}}} = \boldsymbol{U}^\intercal\boldsymbol{J}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y}, \ \ \hat{\boldsymbol{y}}_0 = \boldsymbol{U}^\intercal\boldsymbol{y}_0. \tag{3.2}$$

However, the ROM in (3.2) is not Hamiltonian since $(\boldsymbol{U}^\intercal\boldsymbol{J})^\intercal = \boldsymbol{J}^\intercal\boldsymbol{U} = -\boldsymbol{J}\boldsymbol{U} \neq -\boldsymbol{U}^\intercal\boldsymbol{J}$. This implies that the sympletic structure is lost, and the reduced Hamiltonian functional $\hat{H} = H \circ \hat{\boldsymbol{y}} : \mathbb{R}^{2r} \to R$ is not conserved (c.f., [9, 10]) since

$$\dot{\hat{H}} = \dot{\hat{\boldsymbol{y}}} \cdot \nabla\hat{H} = \boldsymbol{U}^\intercal\boldsymbol{J}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{y}} \cdot \boldsymbol{U}^\intercal(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{y}} = \boldsymbol{U}\boldsymbol{U}^\intercal(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{y}} \cdot \boldsymbol{J}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{y}} \neq 0, \ \ \forall \boldsymbol{\mu} \in \mathbb{R}^{N_p}.$$

To get around this defect, a useful strategy was developed in [7], whereby a skew-symmetric matrix $\hat{\boldsymbol{J}} \in \mathbb{R}^{2r\times 2r}$ is obtained by treating the equivariance condition $\boldsymbol{U}^\intercal\boldsymbol{J} = \hat{\boldsymbol{J}}\boldsymbol{U}^\intercal$ as an overdetermined system for $\hat{\boldsymbol{J}}$. Solving this system in a least-squares sense with $\hat{\boldsymbol{J}} = \boldsymbol{U}^\intercal\boldsymbol{J}\boldsymbol{U}$, the following ROM can then be constructed:

$$\dot{\hat{\boldsymbol{y}}} = \hat{\boldsymbol{J}}\boldsymbol{U}^\intercal(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{y}} = \hat{\boldsymbol{J}}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} = \boldsymbol{U}^\intercal\boldsymbol{J}\boldsymbol{U}\boldsymbol{U}^\intercal(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y}. \tag{3.3}$$

The above ROM preserves the reduced Hamiltonian since the symmetry conditions on $\hat{\boldsymbol{T}}$ and $\hat{\boldsymbol{J}}$ imply

$$\dot{\hat{H}} = \dot{\hat{\boldsymbol{y}}} \cdot \nabla\hat{H} = \hat{\boldsymbol{J}}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} \cdot (\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} = -(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} \cdot \hat{\boldsymbol{J}}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} = 0. \tag{3.4}$$

However, the Hamiltonian ROM (3.3) is variationally inconsistent with the original FOM due to the presence of an additional projection $\boldsymbol{U}\boldsymbol{U}^\intercal$ arising due to the least-squares projection of $\boldsymbol{J}$ on $\boldsymbol{U}$ [10]. A Galerkin projection of the Hamiltonian variational principle satisfied by the state $\boldsymbol{y}$ does not yield the variational principle satisfied by the reduced state $\hat{\boldsymbol{y}}$. This inconsistency in the ROM may lead to a considerably negative impact on the ROM's accuracy in several cases of interest.

To resolve this inconsistency, the authors of [10] perform a Petrov-Galerkin projection using the test space $\boldsymbol{J}\boldsymbol{U}$ to obtain the variationally consistent Hamiltonian ROM

$$\dot{\hat{\boldsymbol{y}}} = \hat{\boldsymbol{J}}^{-\intercal}\nabla\hat{H} = \hat{\boldsymbol{J}}^{-\intercal}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}}, \tag{3.5}$$

which is symplectic since $\hat{\boldsymbol{J}}^{-\intercal} = -\hat{\boldsymbol{J}}^{\intercal}$ and preserves the Hamiltonian as indicated by the following identity:

$$\dot{\hat{H}} = \dot{\hat{\boldsymbol{y}}} \cdot \nabla \hat{H} = \hat{\boldsymbol{J}}^{-\intercal}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} \cdot (\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} = -(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} \cdot \hat{\boldsymbol{J}}^{-\intercal}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}} = 0.$$

There is alternate way to construct a trial basis for the model reduction of Hamiltonian systems utilizing the technique of Proper Symplectic Decomposition (PSD) [16]. A basis $\boldsymbol{U}$ constructed through PSD has the desirable property that $\boldsymbol{V}^{\intercal}\boldsymbol{J} = \boldsymbol{J}_{2r}\boldsymbol{U}^{\intercal}$, where $\boldsymbol{J}_{2r} \in \mathbb{R}^{2r \times 2r}$ is the canonical symplectic matrix of dimensions $2r$. The advantage of this is variationally consistent Hamiltonian ROM which follows directly from the Galerkin projection,

$$\dot{\hat{\boldsymbol{y}}} = \boldsymbol{U}^{\intercal}\boldsymbol{J}\boldsymbol{A}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{y}} = \boldsymbol{J}_{2r}\boldsymbol{U}^{\intercal}\boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{U}\hat{\boldsymbol{y}} = \boldsymbol{J}_{2r}\hat{\boldsymbol{A}}(\boldsymbol{\mu})\hat{\boldsymbol{y}}. \tag{3.6}$$

It is common for the matrix $\boldsymbol{A}(\boldsymbol{\mu})$ has a block diagonal structure so that (3.1) may be written in the block form

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_1(\boldsymbol{\mu}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_2(\boldsymbol{\mu}) \end{bmatrix} \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{bmatrix}, \tag{3.7}$$

For many systems, the parameter dependence may only be carried by the block $\boldsymbol{A}_2(\boldsymbol{\mu})$ with $\boldsymbol{A}_1(\boldsymbol{\mu}) = \boldsymbol{A}_1$. Hence, depending on the specific application, the entire matrix $\boldsymbol{A}(\boldsymbol{\mu})$ or a block of it may be replaced by the contraction $\boldsymbol{T}\boldsymbol{\mu}$. It is also interesting to note here that the use of a PSD basis retains this block structure at the ROM level. This is discussed in greater detail later in Subsection 3.2.

3.2. Methodology. In this subsection, the two strategies for constructing the trial matrix $\boldsymbol{U}$ are first introduced. Following that, the non-intrusive inference of symmetry-constrained tensor operators from data is described. Let $\boldsymbol{Q}_s = \{q^i(t_\alpha, \boldsymbol{\mu}_s)\}_{i,\alpha}$, $\boldsymbol{P}_s = \{p^i(t_\alpha, \boldsymbol{\mu}_s)\}_{i,\alpha} \in \mathbb{R}^{M \times N_t}$ be the s-th position and momentum snapshot matrices respectively, collected by solving the FOM at N_t times, for parameter vector $\boldsymbol{\mu}_s$. Let $\boldsymbol{Y}_s = [\boldsymbol{Q}_s^{\intercal} \ \boldsymbol{P}_s^{\intercal}]^{\intercal} \in \mathbb{R}^{N \times N_t}$ be the corresponding state matrix.

3.2.1. Proper Orthogonal Decomposition. Suppose $\boldsymbol{Y}$ represents the vertical concatenation $[\boldsymbol{Y}_1 \ \boldsymbol{Y}_2 \ \cdots \ \boldsymbol{Y}_{N_s}]$ of the state snapshots. A POD trial basis $\boldsymbol{U} \in \mathbb{R}^{N \times 2r}$ may be constructed by computing the truncated SVD $\boldsymbol{Y} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\intercal}$ so that $\boldsymbol{U}$ contains the first $2r$ left-singular vectors of $\boldsymbol{Y}$. Once $\boldsymbol{U}$ is obtained, a ROM of form (3.4) or (3.5) can be constructed via Galerkin or Petrov-Galerkin projection of the FOM such that the reduced tensor $\hat{\boldsymbol{T}} \in \mathbb{R}^{2r} \otimes \mathbb{R}^{2r} \otimes \mathbb{R}^{N_p}$ has components

$$\hat{T}^a_{bx} = \sum_{i=1}^{N}\sum_{j=1}^{N} U^a_i T^i_{jx} U^j_b.$$

and can be inferred from data.

3.2.2. Proper Symplectic Decomposition. Suppose $\boldsymbol{R}_s \in \mathbb{R}^{M \times 2N_t}$ denotes the s-th horizontally concatenated matrix $\boldsymbol{R}_s = [\boldsymbol{Q}_s \ \boldsymbol{P}_s]$ and $\boldsymbol{R} \in \mathbb{R}^{M \times 2N_t N_s}$ be the concatenation $[\boldsymbol{R}_1 \ \boldsymbol{R}_2 \ \cdots \ \boldsymbol{R}_{N_s}]$. A symplecticity-preserving basis can be obtained via the contangent lift algorithm [16] by writing $\boldsymbol{R} = \tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\boldsymbol{V}^{\intercal}$, where $\tilde{\boldsymbol{U}} \in \mathbb{R}^{M \times r}$ contains the first r left singular vectors of $\boldsymbol{R}$. A trial basis can be obtained from $\tilde{\boldsymbol{U}}$ simply as $\boldsymbol{U} = \mathrm{Diag}(\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{U}})$. The matrix $\boldsymbol{U}$ has now been constructed through a Proper Symplectic Decomposition (PSD) and yields the reduced Hamiltonian system (3.6) through a direct Galerkin projection of the FOM.

Furthermore, if a Hamiltonian system has the block form (3.7), the use of a PSD basis retains this form at the reduced level so that the following may be written

$$\begin{bmatrix}\dot{\hat{\boldsymbol{q}}}\\ \dot{\hat{\boldsymbol{p}}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}_r\\ -\boldsymbol{I}_r & 0\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{A}}_1(\boldsymbol{\mu}) & \boldsymbol{0}\\ \boldsymbol{0} & \hat{\boldsymbol{A}}_2(\boldsymbol{\mu})\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{q}}\\ \hat{\boldsymbol{p}}\end{bmatrix},$$

allowing one to split the ROM into two subcomponents

$$\dot{\hat{\boldsymbol{q}}} = \hat{\boldsymbol{A}}_2(\boldsymbol{\mu})\hat{\boldsymbol{p}} = (\hat{\boldsymbol{T}}_2\boldsymbol{\mu})\hat{\boldsymbol{p}}, \tag{3.8a}$$

$$\dot{\hat{\boldsymbol{p}}} = -\hat{\boldsymbol{A}}_1(\boldsymbol{\mu})\hat{\boldsymbol{q}} = -(\hat{\boldsymbol{T}}_1\boldsymbol{\mu})\hat{\boldsymbol{q}}, \tag{3.8b}$$

where $\hat{\boldsymbol{T}}_1, \hat{\boldsymbol{T}}_2 \in \mathbb{R}^r \otimes \mathbb{R}^r \otimes \mathbb{R}^{N_p}$ are now two smaller tensors with components

$$(\hat{T}_1)^a_{bx} = \sum_{i=1}^{N}\sum_{j=1}^{N} \tilde{U}^a_i (T_1)^i_{jx} \tilde{U}^j_b, \;\; (\hat{T}_2)^a_{bx} = \sum_{i=1}^{N}\sum_{j=1}^{N} \tilde{U}^a_i (T_2)^i_{jx} \tilde{U}^j_b.$$

If the matrix $\hat{\boldsymbol{A}}_2$ is independent of $\boldsymbol{\mu}$, the system can be further simplified as only one of the two equations above will need to be endowed with a tensor operator, easing the inference procedure.

3.2.3. Tensor Parametric Operator Inference with Symmetry Enforcement. For parametric Hamiltonian ROMs in (3.4), (3.5), (3.8a), the learned tensor operator needs to satisfy a symmtery constraint, for example one can solve the minimization problem

$$\arg\min_{\hat{\boldsymbol{T}}} \frac{1}{2}\sum_{s=1}^{N_s}\left|\left|\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\left(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s\right|\right|^2 \quad \text{s.t.} \quad \hat{\boldsymbol{T}}\boldsymbol{\nu} = \pm\hat{\boldsymbol{M}}^{-1}\left(\hat{\boldsymbol{T}}\boldsymbol{\nu}\right)^{\intercal}\hat{\boldsymbol{M}} \quad \forall \boldsymbol{\nu} \in \mathbb{R}^{N_p},$$

where $\hat{\boldsymbol{M}}$ is a symmetric, positive-definite matrix, typically emerging as the reduced version of the mass matrix of a finite-element discretization. The following result establishes the inference procedure necessary for the inference.

THEOREM 3.1. *Suppose $\hat{\boldsymbol{A}}_s \in \mathbb{R}^{\bar{r}\times\bar{r}}$ is a reduced operator and $\hat{\boldsymbol{B}}_s \in \mathbb{R}^{\bar{r}\times N_t}, \hat{\boldsymbol{C}}_s \in \mathbb{R}^{\bar{r}\times N_t}$ are reduced snapshot matrices for each $1 \le s \le N_s$, and $\hat{\boldsymbol{M}} \in \mathbb{R}^{\bar{r}\times\bar{r}}$ is symmetric and positive-definite. If $\hat{\boldsymbol{A}}_s, \hat{\boldsymbol{B}}_s$ have full rank for each s, the unique solution to the least-squares minimization problem*

$$\arg\min_{\hat{\boldsymbol{T}}} \frac{1}{2}\sum_{s=1}^{N_s}\left|\left|\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\left(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\right)\hat{\boldsymbol{B}}_s\right|\right|^2 \quad \text{s.t.} \quad \hat{\boldsymbol{T}}\boldsymbol{\nu} = \pm\hat{\boldsymbol{M}}^{-1}\left(\hat{\boldsymbol{T}}\boldsymbol{\nu}\right)^{\intercal}\hat{\boldsymbol{M}} \quad \forall \boldsymbol{\nu} \in \mathbb{R}^{N_p}, \tag{3.9}$$

can be computed by solving the vectorized problem

$$\begin{aligned}\sum_{s=1}^{N_s}\left[\boldsymbol{\mu}_s\boldsymbol{\mu}_s^{\intercal} \otimes_K \left(\hat{\boldsymbol{A}}_s^{\intercal}\hat{\boldsymbol{A}}_s \mathbin{\bar{\oplus}_K} \hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^{\intercal}\right)\right]\operatorname{vec}\hat{\boldsymbol{T}}&\\ = \sum_{s=1}^{N_s}\boldsymbol{\mu}_s \otimes_K \operatorname{vec}\left(\hat{\boldsymbol{A}}_s^{\intercal}\hat{\boldsymbol{C}}_s\hat{\boldsymbol{B}}_s^{\intercal} \pm \hat{\boldsymbol{M}}\hat{\boldsymbol{B}}_s\hat{\boldsymbol{C}}_s^{\intercal}\hat{\boldsymbol{A}}_s\hat{\boldsymbol{M}}^{-1}\right),&\end{aligned} \tag{3.10}$$

where $\otimes_K$ denotes the Kronecker product of matrices and

$$\boldsymbol{X} \mathbin{\bar{\oplus}_K} \boldsymbol{Y} = (\hat{\boldsymbol{M}}^{-1}\boldsymbol{X}\hat{\boldsymbol{M}}^{-1}) \otimes_K (\hat{\boldsymbol{M}}\boldsymbol{Y}\hat{\boldsymbol{M}}) + \boldsymbol{Y} \otimes_K \boldsymbol{X}.$$

Proof. Let $\hat{\boldsymbol{\Lambda}} \in \mathbb{R}^{r \times r \times N_p}$ be a tensor of Lagrange multipliers and define the Lagrangian

$$L\Big(\hat{\boldsymbol{T}}, \hat{\boldsymbol{\Lambda}}\Big) = \frac{1}{2}\sum_{s=1}^{N_s}\Big|\Big|\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\Big|\Big|^2 + \Big\langle \hat{\boldsymbol{\Lambda}}, \hat{\boldsymbol{T}} \mp \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{T}}^{\intercal}\hat{\boldsymbol{M}}\Big\rangle,$$

where $\hat{\boldsymbol{T}}^{\intercal}$ indicates a transpose in the first two indices, i.e.,

$$\Big(\hat{T}^{\intercal}\Big)^i_{jx} = \hat{T}^j_{ix} \qquad \text{and} \qquad \Big(\hat{M}^{-1}\hat{T}^{\intercal}\hat{M}\Big)^i_{jx} = (\hat{M}^{-1})^i_l \hat{T}^k_{lx}\hat{M}^k_j.$$

Differentiating, it follows that

$$\begin{aligned}
dL\Big(\hat{\boldsymbol{T}}, \hat{\boldsymbol{\Lambda}}\Big) &= -\sum_{s=1}^{N_s}\Big\langle \hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s, \hat{\boldsymbol{A}}_s\Big(d\hat{\boldsymbol{T}}\,\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\Big\rangle \\
&\qquad + \Big\langle d\hat{\boldsymbol{\Lambda}}, \hat{\boldsymbol{T}} \mp \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{T}}^{\intercal}\hat{\boldsymbol{M}}\Big\rangle + \Big\langle \hat{\boldsymbol{\Lambda}}, d\hat{\boldsymbol{T}} \mp \hat{\boldsymbol{M}}^{-1}d\hat{\boldsymbol{T}}^{\intercal}\hat{\boldsymbol{M}}\Big\rangle \\
&= \left\langle d\hat{\boldsymbol{T}}, \hat{\boldsymbol{\Lambda}} \mp \hat{\boldsymbol{M}}\hat{\boldsymbol{\Lambda}}^{\intercal}\hat{\boldsymbol{M}}^{-1} - \sum_{s=1}^{N_s}\hat{\boldsymbol{A}}_s^{\intercal}\Big(\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\Big)\hat{\boldsymbol{B}}_s^{\intercal} \otimes \boldsymbol{\mu}_s \right\rangle \\
&\qquad + \Big\langle d\hat{\boldsymbol{\Lambda}}, \hat{\boldsymbol{T}} \mp \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{T}}^{\intercal}\hat{\boldsymbol{M}}\Big\rangle \\
&= \Big\langle d\hat{\boldsymbol{T}}, \partial_{\hat{\boldsymbol{T}}}L\Big\rangle + \Big\langle d\hat{\boldsymbol{\Lambda}}, \partial_{\hat{\boldsymbol{\Lambda}}}L\Big\rangle.
\end{aligned}$$

Setting these gradients to zero yields the Euler-Lagrange equations

$$\hat{\boldsymbol{\Lambda}} \mp \hat{\boldsymbol{M}}\hat{\boldsymbol{\Lambda}}^{\intercal}\hat{\boldsymbol{M}}^{-1} = \sum_{s=1}^{N_s}\hat{\boldsymbol{A}}_s^{\intercal}\Big(\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\Big)\hat{\boldsymbol{B}}_s^{\intercal} \otimes \boldsymbol{\mu}_s \tag{3.11a}$$

$$\hat{\boldsymbol{T}} \mp \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{T}}^{\intercal}\hat{\boldsymbol{M}} = \boldsymbol{0}. \tag{3.11b}$$

In particular, the Lagrange multiplier $\boldsymbol{\Lambda}$ can be eliminated by symmetry considerations, leading to

$$\begin{aligned}
\sum_{s=1}^{N_s}\Big[&\hat{\boldsymbol{A}}_s^{\intercal}\Big(\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\Big)\hat{\boldsymbol{B}}_s^{\intercal} \\
&\pm \hat{\boldsymbol{M}}\hat{\boldsymbol{B}}_s\Big(\hat{\boldsymbol{C}}_s - \hat{\boldsymbol{A}}_s\hat{\boldsymbol{M}}\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\Big)^{\intercal}\hat{\boldsymbol{A}}_s\hat{\boldsymbol{M}}^{-1}\Big] \otimes \boldsymbol{\mu}_s = \boldsymbol{0},
\end{aligned}$$

which upon rearranging and applying (3.11b) yields

$$\begin{aligned}
\sum_{s=1}^{N_s}\Big(&\hat{\boldsymbol{A}}_s^{\intercal}\hat{\boldsymbol{A}}_s\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^{\intercal} + \hat{\boldsymbol{M}}\hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^{\intercal}\hat{\boldsymbol{M}}\Big(\hat{\boldsymbol{T}}\boldsymbol{\mu}_s\Big)\hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{A}}_s^{\intercal}\hat{\boldsymbol{A}}_s\hat{\boldsymbol{M}}^{-1}\Big) \otimes \boldsymbol{\mu}_s \\
&= \sum_{s=1}^{N_s}\Big(\hat{\boldsymbol{A}}_s^{\intercal}\hat{\boldsymbol{C}}_s\hat{\boldsymbol{B}}_s^{\intercal} \pm \hat{\boldsymbol{M}}\hat{\boldsymbol{B}}_s\hat{\boldsymbol{C}}_s^{\intercal}\hat{\boldsymbol{A}}_s\hat{\boldsymbol{M}}^{-1}\Big) \otimes \boldsymbol{\mu}_s.
\end{aligned}$$

This is a Sylvester equation in the first two indices of $\hat{\boldsymbol{T}}$, so applying the "vec trick" [9] and introducing the notation $\boldsymbol{X} \oplus\!\!\!\!\bar{}_K\, \boldsymbol{Y} = (\hat{\boldsymbol{M}}^{-1}\boldsymbol{X}\hat{\boldsymbol{M}}^{-1}) \otimes_K (\hat{\boldsymbol{M}}\boldsymbol{Y}\hat{\boldsymbol{M}}) + \boldsymbol{Y} \otimes_K \boldsymbol{X}$ yields the

partial vectorization of size $r^2 \times N_p$,

$$\sum_{s=1}^{N_s}\Big(\hat{\boldsymbol{A}}_s^\intercal\hat{\boldsymbol{A}}_s \mathbin{\bar{\oplus}_K} \hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^\intercal\Big)\operatorname{vec}_{12}\Big(\hat{\boldsymbol{T}}\Big)\boldsymbol{\mu}_s\boldsymbol{\mu}_s^\intercal \\ = \sum_{s=1}^{N_s}\operatorname{vec}\Big(\hat{\boldsymbol{A}}_s^\intercal\hat{\boldsymbol{C}}_s\hat{\boldsymbol{B}}_s^\intercal \pm \hat{\boldsymbol{M}}\hat{\boldsymbol{B}}_s\hat{\boldsymbol{C}}_s^\intercal\hat{\boldsymbol{A}}_s\hat{\boldsymbol{M}}^{-1}\Big)\boldsymbol{\mu}_s^\intercal.$$

Vectorizing again finally yields the equivalent matrix-vector system in the r^2N_p unknowns of $\operatorname{vec}\hat{\boldsymbol{T}}$,

$$\sum_{s=1}^{N_s}\Big[\boldsymbol{\mu}_s\boldsymbol{\mu}_s^\intercal \otimes_K \Big(\hat{\boldsymbol{A}}_s^\intercal\hat{\boldsymbol{A}}_s \mathbin{\bar{\oplus}_K} \hat{\boldsymbol{B}}_s\hat{\boldsymbol{B}}_s^\intercal\Big)\Big]\operatorname{vec}\hat{\boldsymbol{T}} \\ = \sum_{s=1}^{N_s}\boldsymbol{\mu}_s \otimes_K \operatorname{vec}\Big(\hat{\boldsymbol{A}}_s^\intercal\hat{\boldsymbol{C}}_s\hat{\boldsymbol{B}}_s^\intercal \pm \hat{\boldsymbol{M}}\hat{\boldsymbol{B}}_s\hat{\boldsymbol{C}}_s^\intercal\hat{\boldsymbol{A}}_s\hat{\boldsymbol{M}}^{-1}\Big),$$

since $\operatorname{vec}(\boldsymbol{u}\boldsymbol{v}^\intercal) = \boldsymbol{v} \otimes_K \boldsymbol{u}$. ∎

Theorem 3.1 illustrates how the present tensor-based OpInf can be applied to systems with additional structure arising from symmetry conditions on $\boldsymbol{A}(\boldsymbol{\mu})$. Since quadratic Hamiltonian systems require a gradient $\nabla H(\boldsymbol{\mu}) = \boldsymbol{A}(\boldsymbol{\mu}) = \boldsymbol{T}\boldsymbol{\mu}$ which is self-adjoint, this result guarantees an inferred operator with the desired mathematical behavior.

Before moving to a model problem, consider the inference of matrix operator $\hat{\boldsymbol{A}}_2$ in (3.8a) in the case it is parameter-independent and does not need to be replaced by a tensor contraction. Given snapshot sets $\{\dot{\boldsymbol{Q}}_s\}_{s=1}^{N_s}$ and $\{\boldsymbol{P}_s\}_{s=1}^{N_s}$, this requires solving the following minimization problem:

$$\arg\min_{\hat{\boldsymbol{A}}_2} \frac{1}{2}\sum_{s=1}^{N_s}\Big|\Big|\dot{\hat{\boldsymbol{Q}}}_s - \hat{\boldsymbol{A}}_2\hat{\boldsymbol{P}}_s\Big|\Big|^2 \quad \text{s.t.} \quad \hat{\boldsymbol{A}}_2 = \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{A}}_2^\intercal\hat{\boldsymbol{M}}.$$

This solution to this minimization problem turns out to be a simple Sylvester solve for the entries of $\hat{\boldsymbol{A}}_2$. Omitting the derivation, the linear system equivalent to this minimization is given by

$$[(\hat{\boldsymbol{I}}_r \otimes \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{B}}\hat{\boldsymbol{M}}) + (\hat{\boldsymbol{B}}^\intercal \otimes \hat{\boldsymbol{I}}_r)]\operatorname{vec}\hat{\boldsymbol{A}}_2 = \operatorname{vec}\hat{\boldsymbol{C}}, \tag{3.12}$$

where $\hat{\boldsymbol{I}}_r \in \mathbb{R}^{r\times r}$ is the identity matrix and

$$\hat{\boldsymbol{B}} = \sum_{s=1}^{N_s}\hat{\boldsymbol{P}}_s\hat{\boldsymbol{P}}_s^\intercal, \qquad \hat{\boldsymbol{C}} = \sum_{s=1}^{N_s}\dot{\hat{\boldsymbol{Q}}}_s\hat{\boldsymbol{P}}_s^\intercal + \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{P}}_s\dot{\hat{\boldsymbol{Q}}}_s^\intercal\hat{\boldsymbol{M}}.$$

For a detailed derivation, the reader is referred to [9].

3.3. Model Problem. This subsection how the proposed model reduction and tensor inference procedure can be applied to a parameterized wave equation system. Consider the following initial boundary value problem

$$\begin{cases} \partial_{tt}y(\boldsymbol{x},t) = c(\boldsymbol{\mu})^2\Delta y(\boldsymbol{x},t), & \boldsymbol{x}\in\Omega\times(0,t_f], \\ y(\boldsymbol{x},t) = 0, & \boldsymbol{x}\in\partial\Omega\times(0,t_f], \\ y(\boldsymbol{x},0) = y_0(\boldsymbol{x}), & \boldsymbol{x}\in\Omega, \\ \partial_t y(\boldsymbol{x},0) = 0, & \boldsymbol{x}\in\Omega, \end{cases} \tag{3.13}$$

where the parameterized wavespeed $c(\boldsymbol{\mu})^2 : \mathbb{R}^{N_p} \to \mathbb{R}$ takes the following form

$$c^2(\boldsymbol{\mu}) = \mu_1^2 1_{\Omega_1} + \mu_2^2 1_{\Omega_2} + \ldots + \mu_{N_p}^2 1_{\Omega_{N_p}}.$$

A full order solver for (3.13) can be constructed by borrowing the Hamiltonian-preserving mixed finite element scheme from [18]. The semi-discrete scheme is written in terms of the canonical variables (q, p) and takes the following form: find $(\dot{q}_h, \dot{p}_h) \in W_h \times W_h$ such that

$$(\dot{q}_h, w_h)_{\Omega_h} = (p_h, w_h)_{\Omega_h} \qquad \forall w_h \in W_h, \tag{3.14a}$$
$$(\dot{p}_h, w_h)_{\Omega_h} = (\nabla \cdot \boldsymbol{\sigma}_h, w_h)_{\Omega_h} \quad \forall w_h \in W_h, \tag{3.14b}$$

where $\boldsymbol{\sigma}_h \in V_h$ solves

$$(\frac{1}{c(\boldsymbol{\mu})^2}\boldsymbol{\sigma}_h, \boldsymbol{\xi}_h)_{\Omega_h} + (q_h, \nabla \cdot \boldsymbol{\xi}_h) = 0 \ \ \forall \boldsymbol{\xi}_h \in V_h. \tag{3.14c}$$

The discrete Hamiltonian for this scheme is given by

$$H_h(q_h, p_h) = \frac{1}{2}(p_h, p_h)_{\Omega_h} + \frac{1}{2}(\frac{1}{c(\boldsymbol{\mu})^2}\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h)_{\Omega_h}. \tag{3.15}$$

Here, W_h and V_h are chosen to be the finite element spaces

$$W_h := \{w_h \in L^2(\Omega) : w_h|_K = P_1(K), \forall K \in \Omega_h\},$$
$$V_h := \{\boldsymbol{v}_h \in H(\mathrm{div}, \Omega) : \boldsymbol{v}_h|_K = RT_1(K), \forall K \in \Omega_h; \boldsymbol{v}_h|_{\partial\Omega} = 0\},$$

where $RT_1(K)$ is the Raviart-Thomas space $[P_1(K)]^2 \oplus \boldsymbol{x}P_1(K)$ on element K with $P_1(K)$ denoting the space of polynomials of degree 1 on element K.

The FOM (3.14) can be put in the block form (3.7) as follows. Suppose $\{\phi_i\}_{i=1}^{N_W}$ and $\{\boldsymbol{\psi}_i\}_{i=1}^{N_V}$ are the sets of basis functions of spaces W_h and V_h respectively. Let $\boldsymbol{q}, \boldsymbol{p}$, and $\boldsymbol{\sigma}$ denote the vectors containing the degrees of freedom of q_h, p_h, and $\boldsymbol{\sigma}_h$. By using the test function $w_h = \boldsymbol{\psi}_j$ and the expansions $\boldsymbol{\sigma}_h = \sum_{i=1}^{N_V} \sigma_i(t)\boldsymbol{\psi}_i(\boldsymbol{x})$ and $\boldsymbol{q}_h = \sum_{i=1}^{N_W} q_i(t)\phi_i(\boldsymbol{x})$ in (3.14c), it follows that

$$\sum_{i=1}^{N_V}\left(\frac{1}{c(\boldsymbol{\mu})^2}\boldsymbol{\psi}_i, \boldsymbol{\psi}_j\right)_{\Omega_h} \sigma_i = -\sum_{i=1}^{N_W}(\phi_i, \nabla \cdot \boldsymbol{\psi}_j)_{\Omega_h} q_i,$$

which can be written in the following matrix-vector form

$$\boldsymbol{M}_V(\boldsymbol{\mu})\boldsymbol{\sigma} = -\boldsymbol{S}\boldsymbol{q}, \tag{3.16}$$

where the matrices $\boldsymbol{M}_V(\boldsymbol{\mu}) \in \mathbb{R}^{N_V \times N_V}$ and $\boldsymbol{S} \in \mathbb{R}^{N_V \times N_W}$ have components

$$(M_V)_{i,j} = \sum_{i=1}^{N_V}\left(\frac{1}{c(\boldsymbol{\mu})^2}\boldsymbol{\psi}_i, \boldsymbol{\psi}_j\right)_{\Omega_h}, \quad \boldsymbol{S}_{j,i} = \sum_{i=1}^{N_W}(\phi_i, \nabla \cdot \boldsymbol{\psi}_j)_{\Omega_h}$$

Similarly, after using the test function $w_h = \phi_j$, the expansion $p_h = \sum_{i=1}^{N_W} p_i(t)\phi_i(\boldsymbol{x})$, and the above expansion for $\boldsymbol{\sigma}_h$ in (3.14b), it follows that

$$\sum_{i=1}^{N_W}(\phi_i, \phi_j)_{\Omega_h}\dot{p}_i = \sum_{i=1}^{N_V}(\nabla \cdot \boldsymbol{\psi}_i, \phi_j)_{\Omega_h}\sigma_i,$$

which, after substituting in the solve for $\boldsymbol{\sigma}$ from (3.16), is equivalently written as

$$\boldsymbol{M}_W\dot{\boldsymbol{p}} = \boldsymbol{S}^\intercal\boldsymbol{\sigma} \implies \dot{\boldsymbol{p}} = -\boldsymbol{M}_W^{-1}\boldsymbol{S}^\intercal\boldsymbol{M}_V^{-1}(\boldsymbol{\mu})\boldsymbol{S}\boldsymbol{q}, \tag{3.17}$$

where the mass matrix $\boldsymbol{M}_W \in \mathbb{R}^{N_W \times N_W}$ has components

$$(M_W)_{i,j} = \sum_{i=1}^{N_W} (\phi_i, \phi_j)_{\Omega_h}.$$

Finally, with the test function $w_h = \phi_j$ and the above expansions of q_h and p_h, (3.14a) yields

$$\sum_{i=1}^{N_W} (\phi_i, \phi_j)_{\Omega_h} \dot{q}_i = \sum_{i=1}^{N_W} (\phi_i, \phi_j)_{\Omega_h} p_i,$$

which is equivalent to

$$\dot{\boldsymbol{q}} = \boldsymbol{I}\boldsymbol{p}, \tag{3.18}$$

where $\boldsymbol{I} \in \mathbb{R}^{N_W \times N_W}$ denotes the identity matrix. Together, (3.18) and (3.17) lead to the following block form of the FOM (3.14)

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_W^{-1}\boldsymbol{S}^\intercal\boldsymbol{M}_V^{-1}(\boldsymbol{\mu})\boldsymbol{S} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{bmatrix}, \tag{3.19}$$

Remark 3.1. *Note that the block form* (3.19) *can be rewritten simply as*

$$\dot{\boldsymbol{y}} = \boldsymbol{J}\nabla H = \boldsymbol{J}\boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{y} = (\boldsymbol{T}\tilde{\boldsymbol{\mu}})\boldsymbol{y}, \tag{3.20}$$

for $\boldsymbol{y} := [\boldsymbol{q}, \boldsymbol{p}]^\intercal$ *and* $\tilde{\boldsymbol{\mu}} = [\boldsymbol{\mu}\ 1]^\intercal$*, to which the tensor model reduction can be applied directly with a POD basis as detailed in Algorithm 2. However, if one wishes to use a PSD basis, only equation* (3.17) *is tensorized as*

$$\dot{\boldsymbol{p}} = -\boldsymbol{M}_W^{-1}\boldsymbol{S}^\intercal\boldsymbol{M}_V^{-1}(\boldsymbol{\mu})\boldsymbol{S}\boldsymbol{q} = -(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{q},$$

and (3.18) *is retained in its matrix form. Model reduction is then applied to the two blocks separately as outlined in Algorithm 3.*

As a final point, the expression for the discrete Hamiltonian in (3.15) can also be

formulated in a matrix-vector form as demonstrated in the following computation

$$\begin{aligned}
H_h(q_h,p_h) &= \frac{1}{2}\left(\sum_{i=1}^{N_W} p_i\phi_i, \sum_{j=1}^{N_W} p_j\phi_j\right)_{\Omega_h} + \frac{1}{2}\left(\frac{1}{c^2(\boldsymbol{\mu})}\sum_{i=1}^{N_V}\sigma_i\boldsymbol{\psi}_i, \sum_{j=1}^{N_V}\sigma_j\boldsymbol{\psi}_j\right)_{\Omega_h} \\
&= \frac{1}{2}\sum_{i=1}^{N_W} p_i \sum_{j=1}^{N_W}(\phi_i,\phi_j)_{\Omega_h} p_j + \frac{1}{2}\sum_{i=1}^{N_V}\sigma_i\sum_{j=1}^{N_V}(\frac{1}{c(\boldsymbol{\mu})^2}\boldsymbol{\psi}_i,\boldsymbol{\psi}_j)_{\Omega_h}\sigma_j \\
&= \frac{1}{2}\sum_{i=1}^{N_W}\sum_{j=1}^{N_W} p_i(\boldsymbol{M}_W)_{ij}p_j + \frac{1}{2}\sum_{i=1}^{N_V}\sum_{j=1}^{N_V}\sigma_i(\boldsymbol{M}_V)_{ij}\sigma_j \\
&= \frac{1}{2}\boldsymbol{p}^\intercal\boldsymbol{M}_W\boldsymbol{p} + \frac{1}{2}\boldsymbol{\sigma}^\intercal\boldsymbol{M}_V\boldsymbol{\sigma} \\
&= \frac{1}{2}\boldsymbol{p}^\intercal\boldsymbol{M}_W\boldsymbol{p} + \frac{1}{2}(-\boldsymbol{M}_V^{-1}\boldsymbol{S}\boldsymbol{q})^\intercal\boldsymbol{M}_V(-\boldsymbol{M}_V^{-1}\boldsymbol{S}\boldsymbol{q}) \\
&= \frac{1}{2}\boldsymbol{p}^\intercal\boldsymbol{M}_W\boldsymbol{p} + \frac{1}{2}\boldsymbol{q}^\intercal\boldsymbol{S}^\intercal\boldsymbol{M}_V^{-1}\boldsymbol{S}\boldsymbol{q} \\
&= \frac{1}{2}\langle\boldsymbol{p},\boldsymbol{p}\rangle_{\boldsymbol{M}_W} + \frac{1}{2}\langle\boldsymbol{q},\boldsymbol{M}_W^{-1}\boldsymbol{S}^\intercal\boldsymbol{M}_V^{-1}\boldsymbol{S}\boldsymbol{q}\rangle_{\boldsymbol{M}_W},
\end{aligned}$$

where $\langle\boldsymbol{a},\boldsymbol{b}\rangle_{\boldsymbol{M}_W} = \boldsymbol{a}^\intercal\boldsymbol{M}_W\boldsymbol{b}$ denotes an ℓ^2 inner-product.

REMARK 3.2. *With $\boldsymbol{M} = Diag(\boldsymbol{M}_W,\boldsymbol{M}_W)$, the Hamiltonian can be expressed in terms of the state variable $\boldsymbol{y}$ as $H_h(\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^\intercal\boldsymbol{M}\boldsymbol{A}(\boldsymbol{\mu})\boldsymbol{y}$ with the symmtery constraint $(\boldsymbol{M}\boldsymbol{A}(\boldsymbol{\mu}))^\intercal = \boldsymbol{M}\boldsymbol{A}(\boldsymbol{\mu})$. The equivalent tensor form is $H_h(\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^\intercal\boldsymbol{M}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y}$ with $(\boldsymbol{M}(\boldsymbol{T}\boldsymbol{\mu}))^\intercal = \boldsymbol{M}(\boldsymbol{T}\boldsymbol{\mu})$. The symmetry constraint can be simplified to $\boldsymbol{T}\boldsymbol{\mu} = \boldsymbol{M}^{-1}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{M}$ using the fact that M is a symmetric positive-definite matrix. This is precisely the constraint that is enforced in the minimization problem in theorem 3.1. It is evident that the proposed tensor operator inference procedure does require intrusive knowledge of the matrix $\boldsymbol{M}_W$.*

REMARK 3.3. *When Legendre polynomials on $[-1,1]$ are used as local basis functions for the finite element space W_h, the matrix $\boldsymbol{M}_W$ is diagonal. On a uniform mesh, if the local space $P_0(K)$, comprising of constant functions on K, is used instead of $P_1(K)$, M_W becomes a scaled identity matrix. In this case, only the scaling factor needs to be known intrusively.*

4. Numerical Results. In this section, the tensor operator inference framework is applied to the parameterized heat equation and wave equation model problems discussed in previous sections. In either case, both intrusive and non-intrusive ROMs are built, and their accuracy is bench-marked by recording the relative L^2 error of the ROM solution as a function of the number of reduced dimensions. The error is computed as follows. If $\boldsymbol{Y},\tilde{\boldsymbol{Y}} \in \mathbb{R}^{N\times Nt}$ are matrices containing the FOM and ROM solutions y_h and $\tilde{y}_h$ at N_t time points, then the L^2 error can be computed as

$$\text{Error}(\boldsymbol{Y},\tilde{\boldsymbol{Y}}) = RL_2(y_h(\boldsymbol{x}),\tilde{y}_h(\boldsymbol{x}) = \sqrt{\sum_{n=1}^{N_t}\frac{||y_h^n-\tilde{y}_h^n||^2}{||y_h^n||^2}}. \tag{4.1}$$

On the other hand, the projection error of a snapshot matrix $\boldsymbol{Y}$ with respect to given trial basis $\boldsymbol{U}_r$ is also computed using (4.1) by setting $\tilde{\boldsymbol{Y}}$ to be the projection $\boldsymbol{U}_r\boldsymbol{U}_r^\intercal\boldsymbol{Y}$.

The data matrices on the left-hand sides of (2.5) and (3.10) can often have large condition numbers, leading to numerical instabilities in the inference process. To mitigate this,

Algorithm 2 Tensor Hamiltonian Operator Inference with Structure and a POD basis (POD H-OpInf)

Input: Snapshot sets $\{\boldsymbol{Q}_s\}_{s=1}^{N_s}, \{\boldsymbol{P}_s\}_{s=1}^{N_s}$ of the FOM solution at times $\mathcal{T} = \{t_0, \ldots, t_{N_t}\}$, and training parameters $\mathcal{M} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{N_s}\}$, intrusive access to the mass matrix $\boldsymbol{M} \coloneqq \mathrm{Diag}(\boldsymbol{M}_W, \boldsymbol{M}_W)$, and reduced dimension $2r \geq 2$.

Output: Reduced tensor operator $\hat{\boldsymbol{T}} \in \mathbb{R}^{2r \times 2r \times N_p}$ in the Hamiltonian $\hat{H}(\hat{\boldsymbol{y}}) = \frac{1}{2}\hat{\boldsymbol{y}}^\intercal \hat{\boldsymbol{M}}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}}$ satisfying $\hat{\boldsymbol{M}}^{-1}(\hat{\boldsymbol{T}}\boldsymbol{\mu})^\intercal \hat{\boldsymbol{M}} = \hat{\boldsymbol{T}}\boldsymbol{\mu}$ for $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$.

1: Compute the time derivative data $\{\dot{\boldsymbol{Q}}_s\}_{s=1}^{N_s}$, $\{\dot{\boldsymbol{P}}_s\}_{s=1}^{N_s}$ from $\{\boldsymbol{Q}_s\}_{s=1}^{N_s}, \{\boldsymbol{P}_s\}_{s=1}^{N_s}$ using the finite difference formula (2.4).

2: Build the vertically concatenated data matrices $\boldsymbol{Y}_s = [\boldsymbol{Q}_s^\intercal \;\; \boldsymbol{P}_s^\intercal]^\intercal$ and $\dot{\boldsymbol{Y}}_s = [\dot{\boldsymbol{Q}}_s{}^\intercal \;\; \dot{\boldsymbol{P}}_s{}^\intercal]^\intercal \in \mathbb{R}^{N \times N_t}$ for $s = 1, \ldots, N_s$.

3: Construct a reduced basis $\boldsymbol{U} \in V_{2r}(\mathbb{R}^N)$ through a truncated SVD of the matrix $\boldsymbol{Y} = [\boldsymbol{Y}_1 \;\; \ldots \;\; \boldsymbol{Y}_{N_s}]$.

4: Set the reduced snapshots $\hat{\boldsymbol{Y}}_s = \boldsymbol{U}^\intercal \boldsymbol{Y}_s$ and $\dot{\hat{\boldsymbol{Y}}}_s = \boldsymbol{U}^\intercal \dot{\boldsymbol{Y}}_s \in \mathbb{R}^{2r \times N_t}$. for $s = 1, \ldots, N_s$

5: Obtain $\hat{\boldsymbol{J}} \in \mathbb{R}^{2r \times 2r}$ by setting $\hat{\boldsymbol{J}} = \boldsymbol{U}^\intercal \boldsymbol{J} \boldsymbol{U}$ with $\boldsymbol{J} \in \mathbb{R}^{N \times N}$.

6: Obtain $\tilde{\mathcal{M}} = \{\tilde{\boldsymbol{\mu}}_1, \ldots, \tilde{\boldsymbol{\mu}}_{N_s}\}$ where $\tilde{\boldsymbol{\mu}}_s = [\boldsymbol{\mu}_s \;\; 1]^\intercal$.

7: **if** Variational consistency is desired **then**

8: Solve (3.10) for $\hat{\boldsymbol{T}}$ with $\hat{\boldsymbol{C}}_s = \dot{\hat{\boldsymbol{Y}}}_s$, $\hat{\boldsymbol{B}}_s = \hat{\boldsymbol{Y}}_s$, $\boldsymbol{A}_s = \hat{\boldsymbol{J}}^{-\intercal}$ and $\boldsymbol{\mu}_s$ replaced by $\tilde{\boldsymbol{\mu}}_s$.

9: **else**

10: Solve (3.10) for $\hat{\boldsymbol{T}}$ with $\hat{\boldsymbol{C}}_s = \dot{\hat{\boldsymbol{Y}}}_s$, $\hat{\boldsymbol{B}}_s = \hat{\boldsymbol{Y}}_s$, $\boldsymbol{A}_s = \hat{\boldsymbol{J}}$ and $\boldsymbol{\mu}_s$ replaced by $\tilde{\boldsymbol{\mu}}_s$.

a regularization term, $\lambda \boldsymbol{I}$, where $\lambda > 0$ and $\boldsymbol{I}$ is the identity matrix of appropriate size, may be added to improve stability. For the numerical examples considered in this paper, $\lambda \sim 10^{-8}$ provides a sufficient amount of regularization; in more complex problems, the choice of regularization may play an important role, see [13, 14] for regularization selection strategies that can be applied to the standrd OpInf from described in Remark 2.2.

The following subsections provide a detailed description of the results of the numerical experiments.

4.1. Heat Equation.

4.1.1. 2D Experiment. This numerical experiment is performed on the 2D spatial domain $\Omega = [0, 2\pi]^2$ with $t_f = 1$. The square domain is spit into four equally sized smaller subdomains as

$$\Omega = [0, \pi]^2 \cup [\pi, 2\pi] \times [0, \pi] \cup [0, \pi] \times [\pi, 2\pi] \cup [\pi, 2\pi]^2.$$

The parameterized thermal conductivity $c(\boldsymbol{\mu})$ reads

$$c(\boldsymbol{\mu}) = \mu_1 1_{[0,\pi]^2} + \mu_2 1_{[\pi,2\pi]\times[0,\pi]} + \mu_3 1_{[0,\pi]\times[\pi,2\pi]} + \mu_4 1_{[\pi,2\pi]^2}.$$

The mesh Ω_h consists of 1496 unstructured elements with 801 degrees of freedom, and the time step size Δt is set to 0.001. Hence, for this experiment $N = 801$ and $N_t = 1001$. The initial condition is picked to be

$$q_0(x_1, x_2) = \exp\big(-(x_1 - \pi)^2 - (x_2 - \pi)^2\big)$$

Using the first-order backward euler scheme for the time-integration, the fully discrete FOM can be written as

$$\frac{\dot{\boldsymbol{q}}^{n+1} - \dot{\boldsymbol{q}}^n}{\Delta t} = -\boldsymbol{M}^{-1}\boldsymbol{S}(\boldsymbol{\mu})\boldsymbol{q}^{n+1} = (\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{q}^{n+1}. \tag{4.2}$$

Algorithm 3 Tensor Hamiltonian Operator Inference with Structure and a PSD basis (PSD H-OpInf)

Input: Snapshot sets $\{\boldsymbol{Q}_s\}_{s=1}^{N_s}, \{\boldsymbol{P}_s\}_{s=1}^{N_s}$ of the FOM solution at times $\mathcal{T} = \{t_0, \ldots, t_{N_t}\}$, and training parameters $\mathcal{M} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{N_s}\}$, intrusive access to the mass matrix $\boldsymbol{M}_W$, and reduced dimension $r > 1$.

Output: Reduced tensor operator $\hat{\boldsymbol{T}} \in \mathbb{R}^{r\times r\times N_p}$ and reduced matrix operator $\hat{\boldsymbol{I}} \in \mathbb{R}^{r\times r}$ in the Hamiltonian $\hat{H}(\hat{\boldsymbol{y}}) = \frac{1}{2}\hat{\boldsymbol{p}}^\intercal \hat{\boldsymbol{M}}_W \hat{\boldsymbol{I}}\hat{\boldsymbol{p}} + \frac{1}{2}\hat{\boldsymbol{q}}\hat{\boldsymbol{M}}_W\hat{\boldsymbol{T}}\hat{\boldsymbol{q}}$ satisfying $\hat{\boldsymbol{M}}_W^{-1}(\hat{\boldsymbol{T}}\boldsymbol{\mu})^\intercal \hat{\boldsymbol{M}}_W = \hat{\boldsymbol{T}}\boldsymbol{\mu}$ for $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$.

1: Compute the time derivative data $\{\dot{\boldsymbol{Q}}s\}_{s=1}^{N_s}$, $\{\dot{\boldsymbol{P}}s\}_{s=1}^{N_s}$ from $\{\boldsymbol{Q}_s\}_{s=1}^{N_s}, \{\boldsymbol{P}_s\}_{s=1}^{N_s}$ using the finite difference formula (2.4).

2: Build the horizontally concatenated snapshot matrices $\boldsymbol{R}_s = [\boldsymbol{Q}_s \ \boldsymbol{P}_s] \in \mathbb{R}^{N\times 2N_t}$ for $s = 1, \ldots, N_s$.

3: Construct a reduced basis $\boldsymbol{U} \in V_r(\mathbb{R}^N)$ through a truncated SVD of the matrix $\boldsymbol{R} = [\boldsymbol{R}_1 \ \ldots \ \boldsymbol{R}_{N_s}]$.

4: Set the reduced snapshots $\hat{\boldsymbol{Q}}_s = \boldsymbol{U}^\intercal \boldsymbol{Q}_s$, $\dot{\hat{\boldsymbol{Q}}}_s = \boldsymbol{U}^\intercal \dot{\boldsymbol{Q}}_s$, $\hat{\boldsymbol{P}}_s = \boldsymbol{U}^\intercal \boldsymbol{P}_s$, and $\dot{\hat{\boldsymbol{P}}}_s = \boldsymbol{U}^\intercal \dot{\boldsymbol{P}}_s \in \mathbb{R}^{r\times N_t}$ for $s = 1, \ldots, N_s$.

5: Solve (3.10) for $\hat{\boldsymbol{T}}$ with $\hat{\boldsymbol{C}}_s = \dot{\hat{\boldsymbol{P}}}_s, \boldsymbol{A}_s = -\boldsymbol{I}_r$, and $\hat{\boldsymbol{B}}_s = \hat{\boldsymbol{Q}}_s$, where $\boldsymbol{I}_r \in \mathbb{R}^{r\times r}$ denotes the identity matrix.

6: Solve (3.12) for $\hat{\boldsymbol{I}} = \hat{\boldsymbol{A}}_2$ using snapshot sets $\{\dot{\boldsymbol{Q}}_s\}_{s=1}^{N_s}$ and $\{\boldsymbol{P}_s\}_{s=1}^{N_s}$.

To generate the snapshots $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{N_s}$, the fully discrete FOM (4.2) is used with 10 random samples of $\boldsymbol{\mu}_s$ uniformly generated in the interval $(0.1, 0.3)^4$. A POD basis is then constructed from the snapshot data, and OpInf and Intrusive ROMs are built. The tensor operator in the OpInf ROM is inferred through the procedures outlined in Algorithm 1. Once the ROMs are built, validation is performed by using them to compute solutions with two different parameters $\boldsymbol{\mu}$ sampled from the interval $(0.1, 0.3)^4$, one from the training set and another not included in it. Similar to the FOM, the ROMs are integrated in time using a backward-euler scheme, and the solutions are computed up to the same terminal time $t_f = 1$. The plots of the ROM solutions at terminal time are displayed in Figure 4.1, along with the absolute error $|q_{\text{FOM}}(t_f) - q_{\text{ROM}}(t_f)|$, for $r = 28$ and $\boldsymbol{\mu}$ not present in the training set. These plots reveal that the ROM solution computed using the OpInf ROM closely matches the ROM solution computed using the intrusive ROM.

Relative spacetime L^2 errors of the different ROM solutions are computed using (4.1), and their behavior is studied as the reduced dimension r is increased, along with the projection error of the snapshot dataset. The error profiles for the two cases are displayed in the panels of Figure 4.2, showing the progressive decrease in ROM errors as well as the projection error as r increases. Naturally, the results are more accurate when $\boldsymbol{\mu}$ is selected from the training set, with the relative L^2 errors of both the intrusive and the OpInf ROM solutions reaching 10^{-4}. When $\boldsymbol{\mu}$ is chosen from outside the training set, the L^2 error of the intrusive ROM solution approaches 10^{-4}, while that of the OpInf ROM solution saturates at 10^{-3}, a sufficient level of accuracy for most practical applications.

4.2. Wave Equation. For this numerical experiment, the system (3.20) is considered on the 1D spatial domain $\Omega = [0, 2\pi]$, which is divided into four equally sized subdomains as

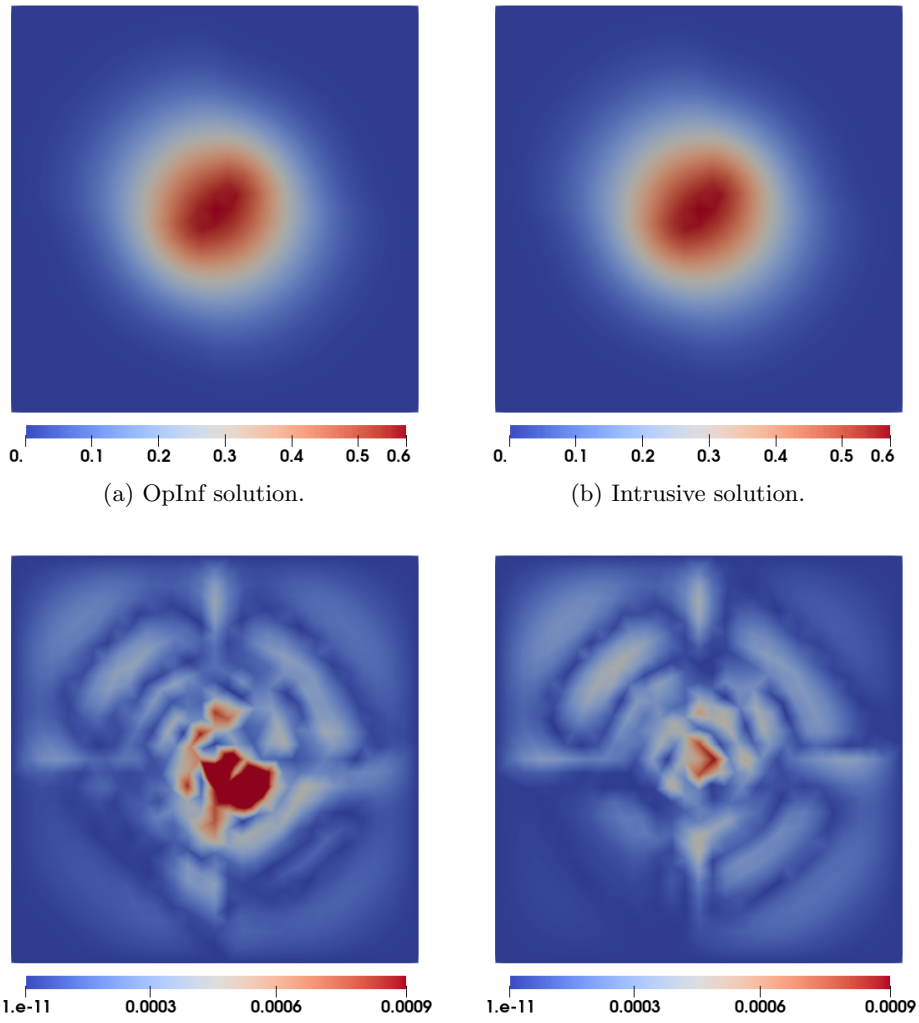
$$\Omega = [0, \frac{\pi}{2}] \cup [\frac{\pi}{2}, \pi] \cup [\pi, \frac{3\pi}{2}] \cup [\frac{3\pi}{2}, 2\pi].$$

The parameterized wavespeed $c(\boldsymbol{\mu})^2$ is then expressed as

$$c(\boldsymbol{\mu})^2 = \mu_1^2 \, 1_{[0,\frac{\pi}{2}]} + \mu_2^2 \, 1_{[\frac{\pi}{2},\pi]} + \mu_3^2 \, 1_{[\pi,\frac{3\pi}{2}]} + \mu_4^2 \, 1_{[\frac{3\pi}{2},2\pi]}.$$

The mesh Ω_h contains 2000 elements with 4000 degrees of freedom, and the time step size is set to $\Delta t = \frac{4\pi}{1000}$. Therefore, $M = 4000$ and $N_t = 1001$. The terminal time for this experiment is $t_f = 4\pi$. Since this is a one-dimensional example, the space for σ_h reduces to

$$V_h := \{v_h \in H^1(\Omega) : v_h|_K = P_1(K), \forall K \in \Omega_h; v_h|_{\partial\Omega} = 0\}.$$



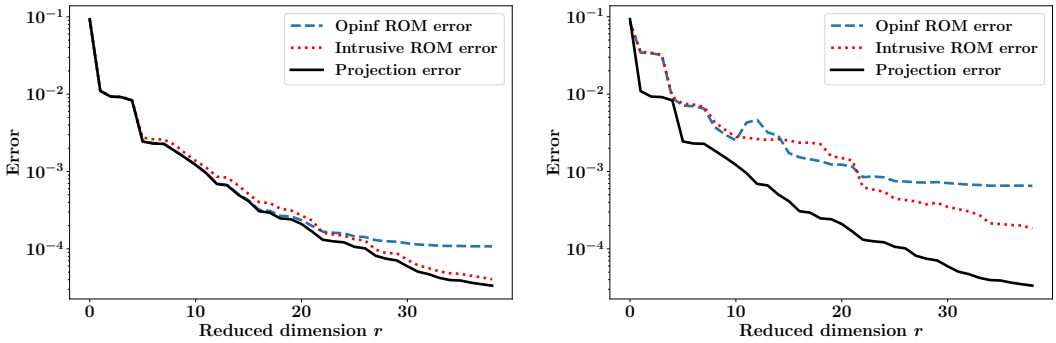
(a) OpInf solution.

(b) Intrusive solution.

(c) OpInf solution absolute error.

(d) Intrusive solution absolute error.

Fig. 4.1: ROM solutions and absolute errors of the parameterized heat equation at terminal time $t_f = 1$ for parameter $\boldsymbol{\mu} \in (0.1, 0.3)^4$ not present in the training set.



(a) $\boldsymbol{\mu}$ present in the training set. (b) $\boldsymbol{\mu}$ not present in the training set.

Fig. 4.2: Relative L^2 error in the 2D heat equation ROM solutions and projection error of the training snapshot matrix as a function of reduced dimension r for two different parameters $\boldsymbol{\mu} \in (0.1, 0.3)^4$, one present in the training set and the other not.

The initial condition for this experiment is chosen as

$$q_0(x) = \exp\big[-(x-\pi)^2\big].$$

The FOM is time-integrated using the implicit midpoint rule, resulting in the following fully discrete symplectic scheme:

$$\frac{\boldsymbol{y}^{n+1} - \boldsymbol{y}^n}{\Delta t} = \boldsymbol{J}(\boldsymbol{T}\boldsymbol{\mu})\boldsymbol{y}^{n+\frac{1}{2}}, \quad \boldsymbol{y}^0 = \boldsymbol{y}_0(x),$$

with $\boldsymbol{y} = [\boldsymbol{q} \quad \boldsymbol{p}]^\intercal$. To generate the training data, the fully discrete FOM is used to compute and construct the snapshot sets $\{\boldsymbol{Q}_s\}_{s=1}^{N_s}$ and $\{\boldsymbol{P}_s\}_{s=1}^{N_s}$ with 10 uniform samples of $\boldsymbol{\mu}_s \in (0.8, 1.1)^4$. Both POD and PSD bases are then constructed from the snapshots, and ROMs of the OpInf, H-OpInf, and intrusive types are built. The tensor operators for these ROMs are inferred using Algorithms 2 and 3.

Validation is performed using two different parameters $\boldsymbol{\mu}_s$, one present in the training set and the other absent. The ROMs are integrated using the implicit midpoint rule described in (4.2), and solutions are computed up to $t_f = 4\pi$, similar to the FOM case. Spacetime plots of the ROM solutions, for $\boldsymbol{\mu}$ absent from the training set, are displayed in Figure 4.3. These panels show that the ROM solutions computed using the different approaches are close to each other within a reasonable level of accuracy.

The L^2 errors of all ROM solutions, along with the projection errors of the snapshot data matrix with respect to the two bases, are recorded. Their behavior is studied as a function of the number of reduced dimensions r for both parameters $\boldsymbol{\mu}$. The error profiles are plotted in Figures 4.4 and 4.5. In the case of the POD ROMs, the L^2 errors of the computed solutions closely follow the decay in the projection error as r increases, for both values of $\boldsymbol{\mu}$. The only deviations occur in the case of the OpInf ROMs; the L^2 errors start increasing for larger r values for both values of $\boldsymbol{\mu}$. In the case when $\boldsymbol{\mu}$ not present in the training set, the ROMs become unstable beyond this point, producing spurious oscillations in the solutions and generating large errors. The other POD ROMs demonstrate appreciable accuracy and generalizability, as the error profiles for both values of $\boldsymbol{\mu}$ are nearly identical.

ROM	Description
OpInf	ROM using a POD basis, without a symmetric learned tensor operator.
PSD OpInf	ROM using a PSD basis, without a symmetric learned tensor operator.
PSD H-OpInf	ROM using a PSD basis, with a symmetric learned tensor operator.
Con OpInf	Variationally consistent ROM using a POD basis, without a symmetric learned tensor operator.
Con H-OpInf	Variationally consistent ROM using a POD basis, with a symmetric learned tensor operator.
Incon OpInf	Variationally inconsistent ROM using a POD basis, without a symmetric learned tensor operator.
Incon H-OpInf	Variationally inconsistent ROM using a POD basis, with a symmetric learned tensor operator.
PSD Intrusive	Intrusive ROM with a PSD basis.
Con Intrusive	Variationally consistent intrusive ROM with a POD basis.
Incon Intrusive	Variationally inconsistent intrusive ROM with a POD basis.

Table 4.1: A glossary of ROMs.

In contrast, the PSD ROM solutions show a different trend. The L^2 error appears to saturate beyond $2r = 10$ for the learned ROMs, even as the L^2 error for the Intrusive ROM solution and the projection error of the snapshot matrix continue to decrease with increasing basis size $2r$. The H-OpInf ROM provides more accurate results, with the L^2 error saturating at a smaller value than that of the OpInf ROM. The enforcement of the symmetry-like constraint for the learned operator results in a more accurate ROM. The learned PSD ROMs offer less generalizability than their POD counterparts, as for $\boldsymbol{\mu}$ not in the training set, the L^2 error of the solutions saturates at much larger values for both the H-OpInf ROM and the OpInf ROM, while the L^2 error of the intrusive ROM solution continues to decrease with the projection error. This can be attributed to the less-informative nature

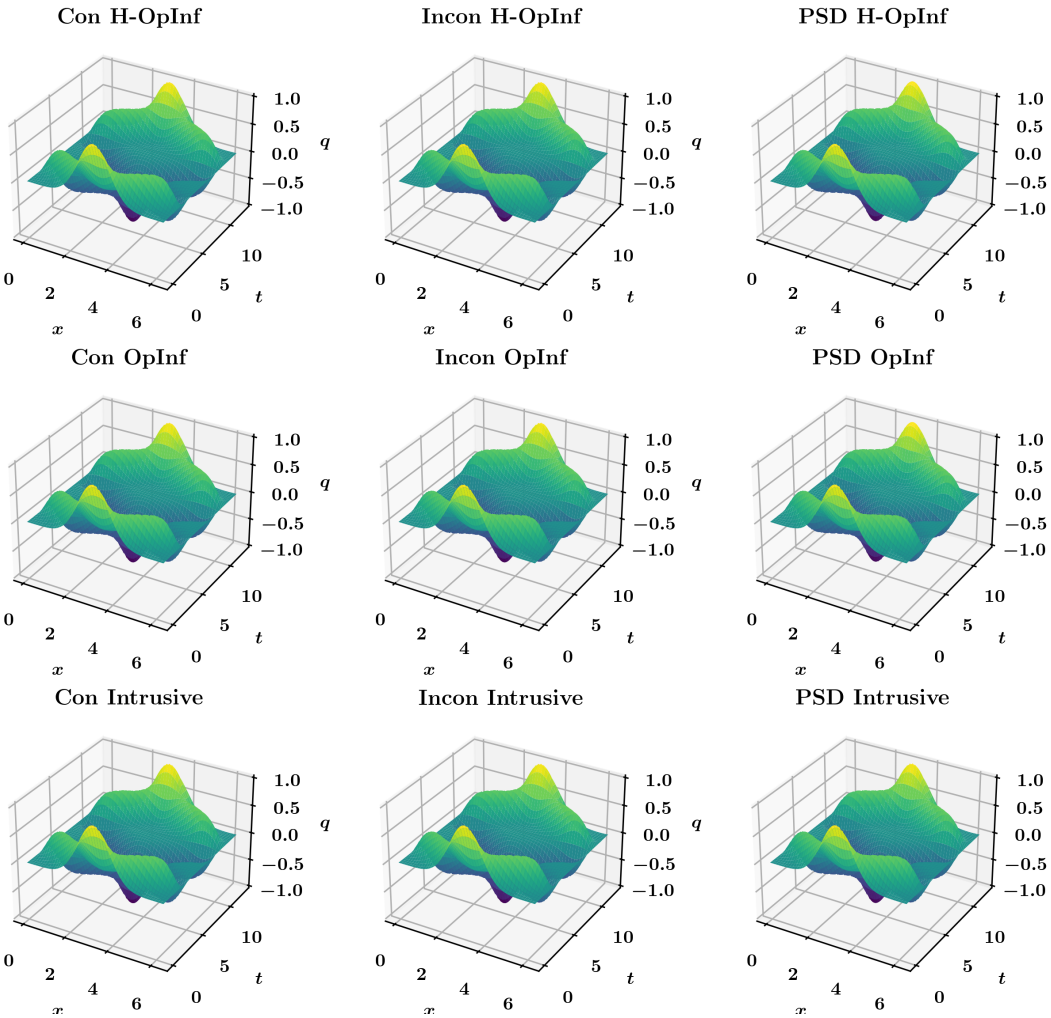


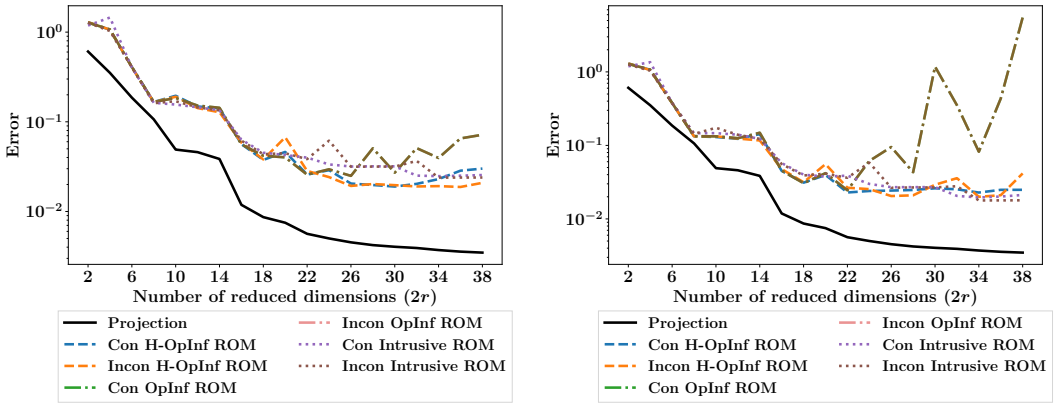
Fig. 4.3: Spacetime plots of the ROM solutions of the 1D parameterized wave equation for $\boldsymbol{\mu} \in (0.8, 1.1)^4$ not present in the training set and basis size $2r = 40$.

of a truncated PSD basis, the basis vectors of which fail to accurately capture the system dynamics unless $2r$ is large.

Finally, Figure 4.6 reveals the errors in the reduced Hamiltonian $\hat{H} = \frac{1}{2}\hat{\boldsymbol{y}}^\intercal \hat{\boldsymbol{M}}(\hat{\boldsymbol{T}}\boldsymbol{\mu})\hat{\boldsymbol{y}}$ associated with all ROM solutions. Panel (a) shows that OpInf ROMs do not preserve the reduced Hamiltonian, as the learned tensor operator does not maintain the necessary symmetry-like structure. Panel (b) shows that the Intrusive POD ROMs and H-OpInf PSD ROMs preserve the Hamiltonian with the highest level of accuracy. Interestingly, the Intrusive PSD ROM yields a larger error due to its construction. From (3.6), it is evident that the term $\boldsymbol{U}^\intercal \boldsymbol{A}(\boldsymbol{\mu})$ is not the true gradient of the reduced Hamiltonian, which is given by:

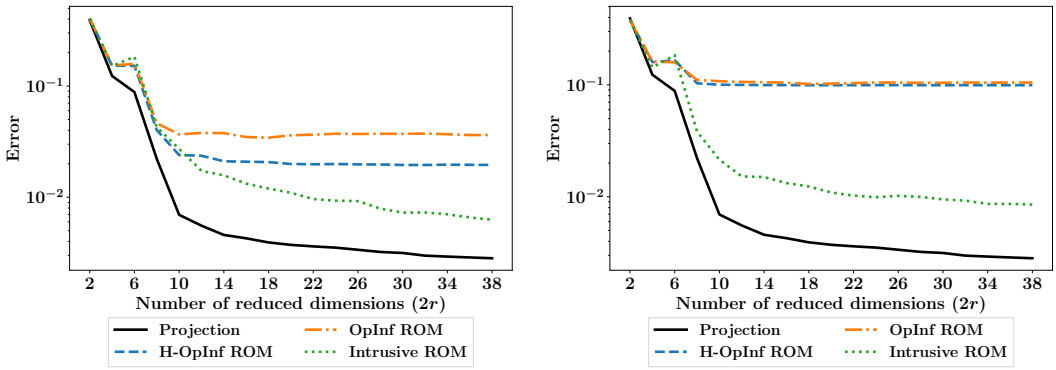
$$\nabla \hat{H} = \boldsymbol{U}^\intercal \boldsymbol{M} \boldsymbol{A}(\boldsymbol{\mu}) \boldsymbol{y}.$$

The PSD H-OpInf ROM learns the true gradient by enforcing the correct condition, as ex-



(a) $\boldsymbol{\mu}$ present in the training set. (b) $\boldsymbol{\mu}$ not present in the training set.

Fig. 4.4: Relative L^2 error in the wave equation POD ROM solutions and projection error of the training snapshot matrix as a function of reduced dimension r for two different parameters $\boldsymbol{\mu} \in (0.8, 1.1)^4$, one present in the training set and the other not. The plots for the OpInf ROMs overlap.

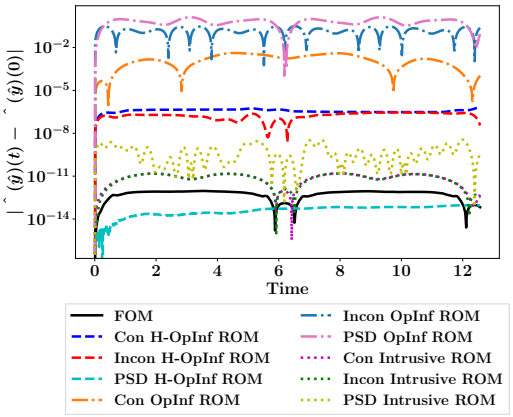


(a) $\boldsymbol{\mu}$ present in the training set. (b) $\boldsymbol{\mu}$ not present in the training set.

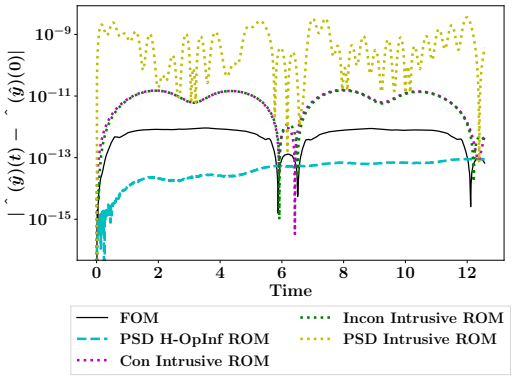
Fig. 4.5: Relative L^2 error in the wave equation PSD ROM solutions and projection error of the training snapshot matrix as a function of reduced dimension r for two different parameters $\boldsymbol{\mu} \in (0.8, 1.1)^4$, one present in the training set and the other not.

plained in Remark 3.2, resulting in better Hamiltonian conservation. Panel (c) shows that both variationally consistent and inconsistent POD H-OpInf ROMs preserve the Hamiltonian with similar accuracy.

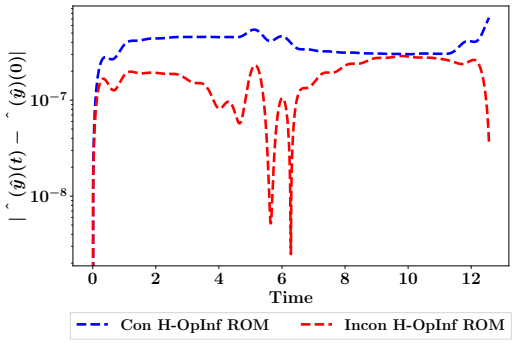
5. Conclusion. This paper presents a novel, tensor-based operator inference procedure for building ROMs of parametric systems of semi-discrete PDEs. The method replaces parameter-dependent operators in existing OpInf frameworks with the contraction of a constant tensor with a generalized parameter vector and formulates a convex, least-squares optimization problem for tensor inference. Additionally, by incorporating symmetry con-



(a) Hamiltonian errors for all ROMs.



(b) Hamiltonian errors for all the intrusive and the PSD H-OpInf ROM.



(c) Hamiltonian errors for the POD H-OpInf ROMs.

Fig. 4.6: Error in the Hamiltonian of the different ROMs as a function of time. (a) shows non-conservation of the Hamiltonian by OpInf ROMs. (b) shows near exact preservation of the Hamiltonian by PSD H-OpInf and POD Intrusive ROMS. (c) shows larger errors for POD H-OpInf ROMs.

straints into the learning problem, the approach extends to parametric Hamiltonian systems, enabling the construction of symplecticity-preserving ROMs. The procedure is tested on two model problems: a 2D parametric heat equation system without intrinsic structure and a 1D parametric linear wave system with an underlying Hamiltonian structure. In both cases, the approach yields stable and accurate ROMs.

The proposed framework is based on a one-shot approach where the reduced tensor operator is inferred by solving the normal equations corresponding to the stationarity of the Lagrangian in the core minimization problem. However, this can often result in ill-conditioned linear systems, potentially leading to unstable ROMs if an appropriate regularization is not introduced. Moreover, when enforcing the symmetry of the learned tensor operator, the procedure involves the construction of a large intermediate tensor, requiring a significant amount of memory to be allocated. These issues can be mitigated by using iterative solvers to directly solve the minimization problems without deriving and solving

the normal equations. Iterative solvers operate on better-conditioned systems, offer computational and memory efficiency, allow for preconditioning to improve convergence, and are easier to parallelize, making them ideal for modern high-performance computing. They will be the focus of future research directions.

REFERENCES

[1] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Review, 57 (2015), pp. 483–531.

[2] P. Benner, M. Ohlberger, A. Patera, G. Rozza, and K. Urban, *Model Reduction of Parametrized Systems*, Modeling, Simulation and Applications, 17, Springer International Publishing, 2017.

[3] G. Berkooz, P. Holmes, and J. L. Lumley, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annual review of fluid mechanics, 25 (1993), pp. 539–575.

[4] T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis, *On learning Hamiltonian systems from data*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 29 (2019), p. 121107.

[5] I. Farcas, R. Gundevia, R. Munipalli, and K. E. Willcox, *Parametric non-intrusive reduced-order models via operator inference for large-scale rotating detonation engine simulations*, in AIAA SCITECH 2023 Forum, 2023.

[6] O. Ghattas and K. Willcox, *Learning physics-based models from data: Perspectives from inverse problems and model reduction*, Acta Numerica, 30 (2021), p. 445–554.

[7] Y. Gong, Q. Wang, and Z. Wang, *Structure-preserving Galerkin POD reduced-order modeling of Hamiltonian systems*, Computer Methods in Applied Mechanics and Engineering, 315 (2017), pp. 780–798.

[8] S. Greydanus, M. Dzamba, and J. Yosinski, *Hamiltonian neural networks*, Advances in neural information processing systems, 32 (2019).

[9] A. Gruber and I. Tezaur, *Canonical and noncanonical Hamiltonian operator inference*, Computer Methods in Applied Mechanics and Engineering, 416 (2023), p. 116334.

[10] A. Gruber and I. Tezaur, *Variationally consistent Hamiltonian model reduction*, arXiv preprint, arXiv:2404.15315 (2024).

[11] P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis, *SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems*, Neural Networks, 132 (2020), pp. 166–179.

[12] Y. LIANG, H. LEE, S. LIM, W. LIN, K. LEE, and C. WU, *Proper orthogonal decomposition and its applications—part i: Theory*, Journal of Sound and Vibration, 252 (2002), pp. 527–544.

[13] S. A. McQuarrie, C. Huang, and K. E. Willcox, *Data-driven reduced-order models via regularised operator inference for a single-injector combustion process*, Journal of the Royal Society of New Zealand, 51 (2021), pp. 194–211.

[14] S. A. McQuarrie, P. Khodabakhshi, and K. E. Willcox, *Non-intrusive reduced-order models for parametric partial differential equations via data-driven operator inference*, SIAM Journal on Scientific Computing, 45 (2023), pp. A1917–A1946.

[15] B. Peherstorfer and K. Willcox, *Data-driven operator inference for nonintrusive projection-based model reduction*, Computer Methods in Applied Mechanics and Engineering, 306 (2016), pp. 196–215.

[16] L. Peng and K. Mohseni, *Symplectic model reduction of Hamiltonian systems*, SIAM Journal on Scientific Computing, 38 (2016), pp. A1–a27.

[17] L. Sirovich, *Turbulence and the dynamics of coherent structures. i. coherent structures*, Quarterly of applied mathematics, 45 (1987), pp. 561–571.

[18] M. A. Sánchez, B. Cockburn, N.-C. Nguyen, and J. Peraire, *Symplectic Hamiltonian finite element methods for linear elastodynamics*, Computer Methods in Applied Mechanics and Engineering, 381 (2021), p. 113843.

[19] C. R. Taylor, *Finite difference coefficients calculator.* `https://web.media.mit.edu/~crtaylor/calculator.html`, 2016.

[20] S. Yuan, *ODE-oriented semi-analytical methods*, in Computational Mechanics in Structural Engineering, F. Y. Cheng and Y. Gu, eds., Elsevier Science Ltd, Oxford, 1999, pp. 375–388.

[21] S. Yildiz, P. Goyal, P. Benner, and B. Karasözen, *Learning reduced-order dynamics for parametrized shallow water equations from data*, International Journal for Numerical Methods in Fluids, 93 (2021), pp. 2803–2821.

SIMULATING ATOMIC PRECISION ADVANCED MANUFACTURING (APAM) ENHANCED BJT

ELANOR WHITESIDES*, JUAN P. MENDEZ†, JEFF IVIE , XUJIAO GAO , AND SHASHANK MISRA

Abstract.
Atomic Precision Advanced Manufacturing (APAM) techniques enable the creation of 2D, electrically significant structures within semiconductors. These structures, called δ-layers, possess high doping densities unachievable by conventional manufacturing methods. δ-layers enhance the electrical properties of semiconductors due to their dopant concentration surpassing the solid solubility of silicon and being all electrically active. Integrating these regions into semiconductor devices using APAM fabrication techniques has strong potential for beyond-Moore and quantum computing applications. This work investigates the gain amplification capabilities of a novel BJT device equipped with 2D doped regions known as δ-layers fabricated using APAM. The strong gain amplification enhancement capabilities of a one-dimensional BJT device are also documented, demonstrating APAM's effectiveness in greatly increasing the current gain of bipolar devices.

1. Introduction. As we approach the fundamental physical limits of traditional transistor fabrication methods, continued miniaturization faces increasing challenges due to issues such as overheating and fabrication size constraints [1]. To address these challenges, there is a growing need to explore fabrication on the atomic scale. Atomically precise fabrication presents new opportunities to shrink and enhance devices, and can be leveraged for device prototyping in applications such as nanowire transistors, tunnel field-effect transistors (TFET), and spintronics [2]. Atomic Precision Advanced Manufacturing (APAM) stands out as the most viable method for fabricating devices at the atomic level [3]. Unlike other fabrication methods that are limited by physical constraints [4] or the solid solubility limit of silicon, APAM allows for the creation of atomically small δ-layer structures (2D structures) with high doping densities that overcome these restrictions.

Utilizing a precise fabrication process is crucial for improving quantum device performance, as it directly affects final atomic positions [5]. APAM fabrication utilizes a surface chemistry process that satisfies important performance requirements such as high precision lithography, controlled phosphorus incorporation, and epitaxial overgrowth that discourages defects [5]. APAM fabrication begins with hydrogen passivation, wherein hydrogen atoms individually attach to each surface silicon atom on the silicon substrate [3]. A Scanning Tunneling Microscope (STM) tip then strategically removes hydrogen atoms, revealing the reactive surface of the silicon in the shape of the desired feature. Phosphine molecules are introduced as a dopant precursor, and phosphorus atoms attach to the reactive sites [6]. Lastly, a silicon cap is applied to the entire surface of the device to protect it from damage and to activate the dopants in the δ-layers. APAM uses this process of 2D STM lithography to create ultra-thin planar δ-layers within silicon at nanometer resolution, consisting of one or several atomic layers of dopant atoms [7][8]. The fabrication process' open geometry control allows for any pattern of choice to be fabricated.

These 2D structures fabricated using APAM offer several key advantages for device enhancement. Their high dopant density can reach orders of 10^{21} cm^{-3}, several orders above that of standard manufacturing methods, which typically achieve maximum concentrations of 10^{19} cm^{-3}[9]. δ-layers' doping concentrations surpass that of the solid solubility limit of silicon, at levels that cause the semiconductor material to behave more as a metallic conductor, greatly increasing electrical conductivity and current density [8]. Additionally, δ-layer dopants are made to be all electrically active by encapsulating the surface phosphorus

*Sandia National Laboratories, (elanor.whitesides01@student.csulb.edu)
†Sandia National Laboratories, (jpmende@sandia.gov)

atoms with silicon at room temperature [5], thus each charge impurity contributes to the overall electrical conductivity.

The following ongoing investigation into a novel Bipolar Junction Transistor (BJT) device explores the potential benefits of δ-layer incorporation and their effect on the BJT current gain at room temperature. This work begins by discussing the results of a one-dimensional BJT device case, which demonstrates the modeling of an APAM δ-layer and its positive impact on current gain through simulation results. This work then presents preliminary findings from two-dimensional BJT device simulations that incorporate a δ-layer structure into the larger BJT environment. The work concludes with a suggestion of future research goals.

2. Simulation approach. This work utilizes the open-source TCAD code, Charon [10], a multi-dimensional, MPI-parallel, semi-classical device simulation code developed at Sandia National Laboratories. Sandia's TCAD Charon code solves self-consistently the Poisson equation and the continuity equations for electrons/holes. Device geometries and meshes are generated using the geometry and meshing tool Cubit [11]. Device simulation results are visualized using the open-source data analysis and visualization tool Paraview [12].

3. 1D BJT Simulations. The ideal one-dimensional BJT device pictured in Fig. 3.1 models a δ-layer structure, with n-type emitter and collector δ-regions possessing high APAM doping densities, and a moderately doped p-type base. The BJT is characterized as a common emitter n-p-n device. The emitter contact is grounded while the emitter-base and base-collector junctions are forward biased such that $V = V_{EB} = V_{EC}$. The base width is set to be 100 nm.

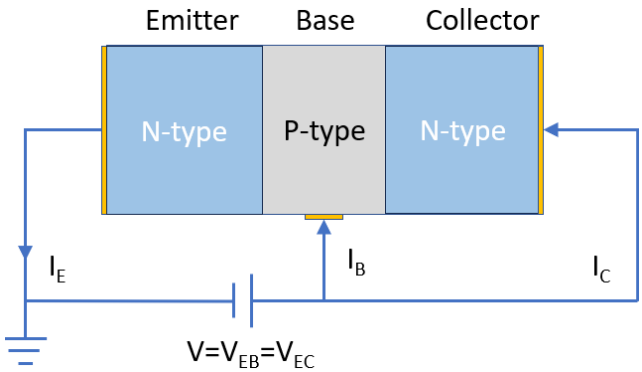


Fig. 3.1: Ideal one-dimensional BJT device.

Illustrated by the right axis of Fig. 3.2, the gain can be determined from the Gummel plot as the ratio of the collector current to the base current, I_C/I_B. In Fig. 3.3, through simulations investigating the doping effects of each region, an understanding of the δ-layer's behavior is developed, confirming its strong gain amplification capabilities. The contribution of each region to the overall current gain is assessed by evaluating the maximum gain for various doping density combinations in the base, emitter δ-layer, and collector δ-layer. N_B, N_E, and N_C represent the doping values in the base (p-type), emitter (n-type) and collector (n-type), respectively, measured in units of cm^{-3}.

Fig. 3.3 (a) shows that increasing the base doping density reduces the current gain. This effect can be attributed to higher base doping densities increasing both recombination within

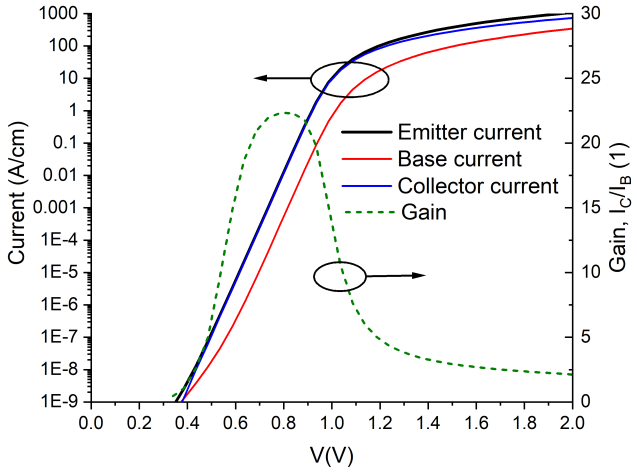


Fig. 3.2: Simulated common-emitter Gummel plots (left axis) for a 1D n-p-n BJT where the doping values for the emitter, base, collector are 10^{20} cm^{-3}, 10^{19} cm^{-3}, and 10^{20} cm^{-3}, respectively. The right axis shows the computed gain.

the base and back injection current from base to emitter. These increases subsequently raise the base current I_b and lower the collector current I_c, leading to a reduced overall current gain. A low base doping level of 10^{18} cm^{-3} is best to achieve a high gain, with a peak value around 275. Fig. 3.3 (b) demonstrates that the collector δ-layer doping density does not significantly influence the gain. Traditional common-emitter BJTs typically achieve gains ranging from tens to hundreds. Conversely, Fig. 3.3 (c) demonstrates that the emitter δ-layer doping density has a significant impact on current gain due to enhanced carrier injection across the emitter-base junction as emitter doping density increases. However, there is a limit to how much the emitter δ-layer doping can be increased, as higher doping levels can also raise the base current through band-to-band tunneling at the emitter-base junction. The one-dimensional δ-layer BJT simulations reveal that significantly higher gains can be attained through the application of APAM doping densities in the emitter and collector δ-layer regions.

4. 2D BJT Simulations. Building on insights gained from the one-dimensional simulations, δ-layer's gain amplification capabilities are applied to the development of a two-dimensional BJT prototype, illustrated in Fig. 4.1. In this prototype, the δ-layer is integrated into the larger, more complex device environment of the two-dimensional BJT. The n-p-n type device configuration consists of an emitter implant, an equivalently doped collector implant, and a lightly to moderately doped base substrate. The δ-layer structure is fabricated between the emitter and collector contacts, separated by a 100 nm wide δ-layer gap at the center of the device. The structure is encapsulated by a silicon cap with a doping density equivalent to that of the base substrate. Preliminary simulations pictured in Fig. 4.2 study the implant doping effects by varying doping densities in the base substrate (N_B), emitter implant ($N_{\text{Emit Impl}}$), and collector implant ($N_{\text{Col Impl}}$) to assess the influence of each region on transistor gain. Similar to the 1D simulations, a common emitter configuration is investigated, where the emitter contact is grounded while the emitter-base and base-collector junctions are forward biased such that $V = V_{EB} = V_{EC}$.

Fig. 4.2 (a) demonstrates that base doping has a significant impact on transistor gain

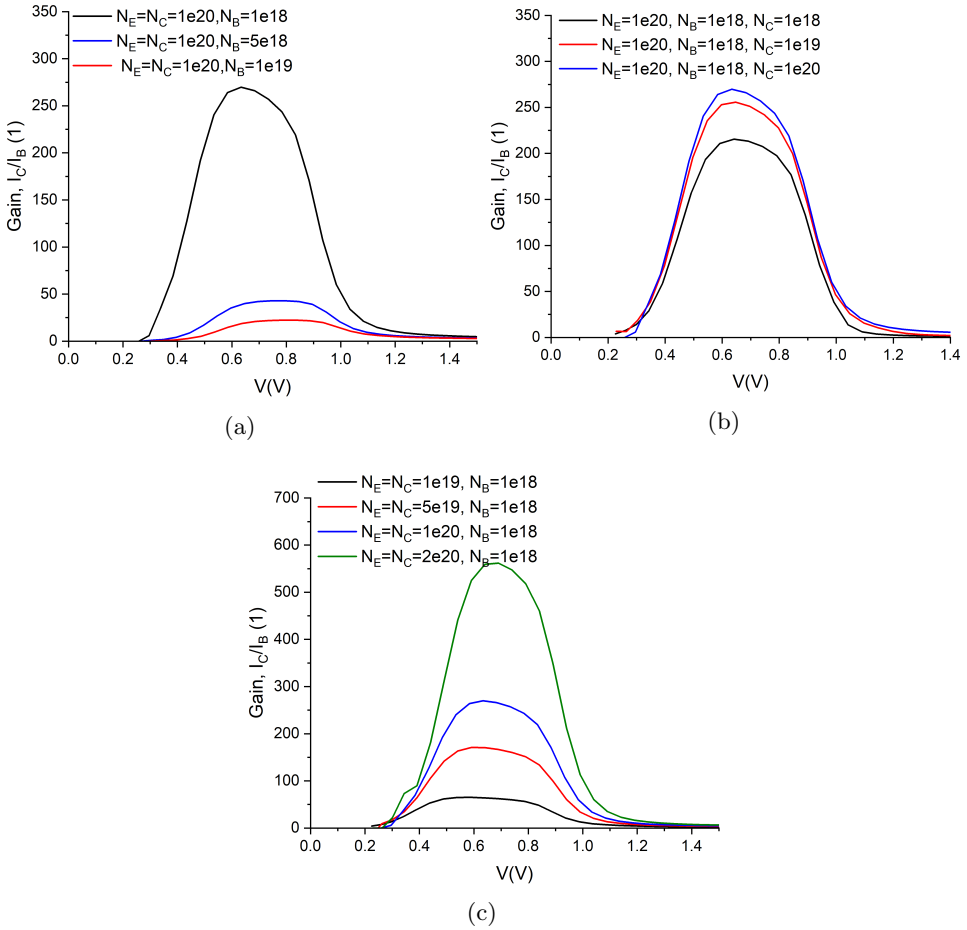


Fig. 3.3: Simulated current gain for the common-emitter BJT configuration shown in Fig. 3.1 as a function of the emitter-base voltage (V_{EB}) and the emitter-collector voltage (V_{EC}), where V=V_{EB}=V_{EC}: (a) for different base doping values; (b) for different collector doping values; and (c) for different emitter doping values. The units are in cm^{-3}.

in the device. When the base doping is reduced from 4×10^{17} cm^{-3} to 3×10^{17} cm^{-3}, the peak gain approximately doubles from 30 to 67. Decreasing base doping greatly reduces carrier recombination in the base, leading to a decrease in base current and a corresponding increase in gain. It is discovered that for base doping densities lower than approximately 3×10^{17} cm^{-3}, this BJT configuration experiences an early gain peaking effect. Lower base doping densities on the orders of 2.5×10^{17} cm^{-3} and 2×10^{17} cm^{-3} achieve high gains at much lower voltage values. A base doping of 2×10^{17} cm^{-3} yields a gain of nearly 6,700, orders above the maximum gains for the higher base doping densities. This early gain peaking effect occurs when the depletion region encroaches significantly into the base, reducing the effective base width and leading to abrupt increases in transistor gain, which results in earlier and higher gain spikes for lower doping densities. It is possible that widening the width of the δ-layer gap or scaling the base width appropriately may lessen this effect

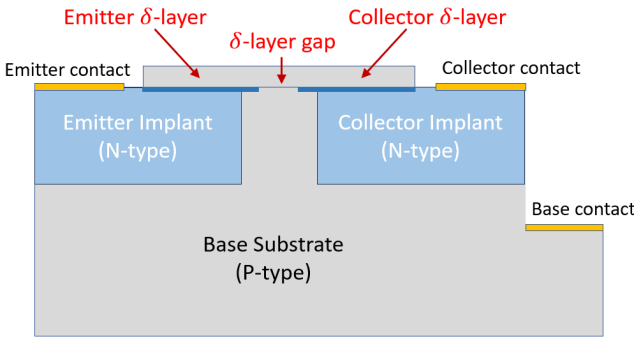


Fig. 4.1: 2D APAM BJT device

and result in more legitimate gains. Alternatively, these early high gains may be desirable for some device applications.

In Fig. 4.2 (b), varying the doping density of the collector implant has no appreciable effect on the gain at all. For each instance of $N_{\text{Col Impl}}$, the gain is approximately 31, which can be attributed to the emitter implant doping density of 5×10^{19} cm^{-3}. The introduction of APAM-fabricated δ-layers in the device shifts the primary influence on gain to the emitter and collector δ-layers, rather than the doped implants themselves. This observation is further demonstrated by Fig. 4.2 (c), which shows that increasing the emitter implant doping slightly increases overall gain, but is less effective than reducing the base substrate doping density. While higher emitter implant doping enhances gain up to a certain level, once the doping concentration competes with that of the δ-layers (around 10^{21} cm^{-3}), any further gain amplification becomes negligible. Raising $N_{\text{Emit Impl}}$ from 5×10^{18} cm^{-3} to 5×10^{19} cm^{-3} brings the maximum gain from 20 to 31, while raising $N_{\text{Emit Impl}}$ from 5×10^{19} cm^{-3} to 10^{20} cm^{-3} does not appreciably increase the gain.

To inspect the impact of the δ-layers on the gain, all of the device simulations are re-conducted with the δ-layers removed, the results of which are shown in Fig. 4.3. In the cases where δ-layers are removed, the n-type donor doping densities of the emitter and collector δ-layers, typically on the order of 10^{21} cm^{-3}, are replaced with a p-type acceptor dopant density of 4×10^{17} cm^{-3} equivalent to that of the base substrate. It can be observed that the presence of δ-layers alone increases the gain by at least an order. A non-δ-layer configuration with a base density of 3×10^{17} cm^{-3} results in a maximum gain of approximately 4, whereas the inclusion of δ-layers at the same doping level increases the maximum gain to 67.

5. Discussion. The simulated two-dimensional BJT device significantly enhances its current gains through the incorporation of the δ-layer structure. The discussed device configuration further optimizes its enhanced current gain by doping the emitter δ-layer sufficiently high and the base substrate appropriately low to avoid currents lost to recombination in the base and prevent excessive current flow from the base into the emitter. Notably, the device achieves its highest gain values with a lightly doped base substrate, around 3×10^{18} cm^{-3}. The combination of low base doping and high emitter δ-layer doping maximizes the benefits of increased conductivity and current density provided by the APAM-fabricated emitter and collector δ-layers. However, the early gain peaking effect observed as a result of base width depletion at low base doping densities must be addressed to ensure that the device is viable for applications where early gain peaking is undesirable. Adjustments such as widening the δ-layer gap or scaling the base width could potentially mitigate this effect and provide more

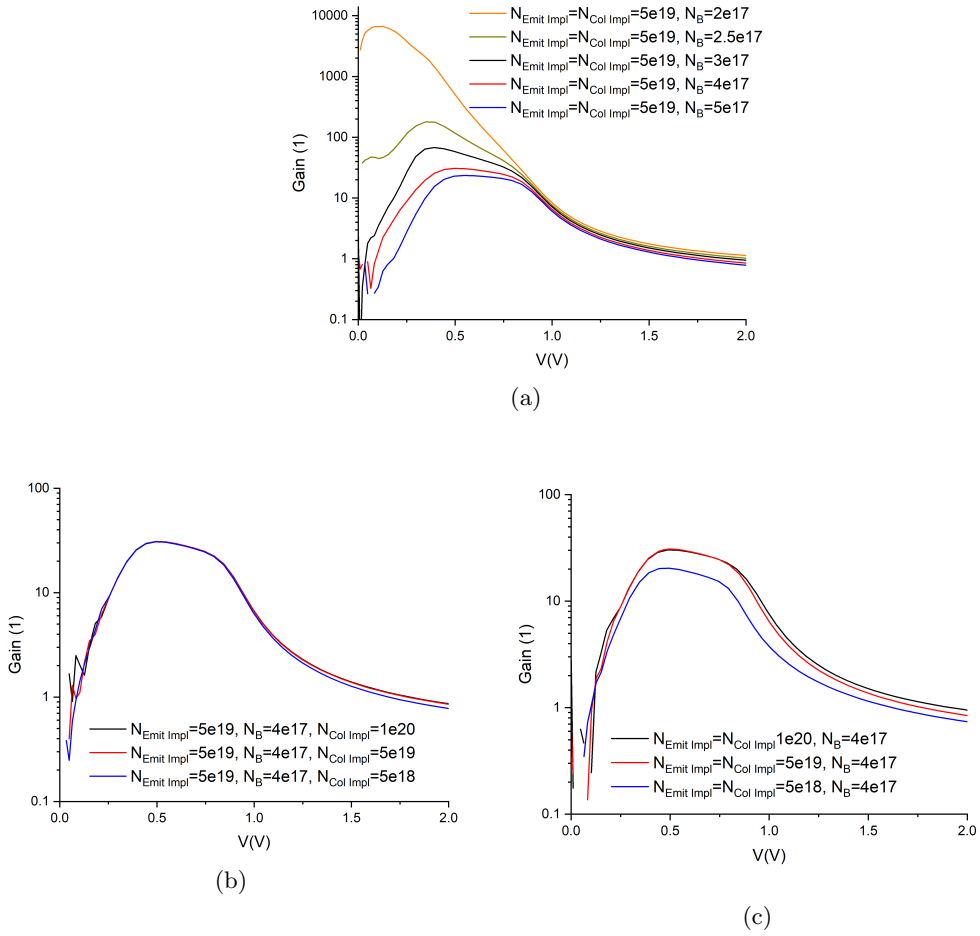


Fig. 4.2: Simulated current gain for the common-emitter BJT configuration shown in Fig. 4.1 as a function of the emitter-base voltage (V_{EB}) and the emitter-collector voltage (V_{EC}), where V=V_{EB}=V_{EC}: (a) for different base doping values; (b) for different collector doping values; and (c) for different emitter doping values. The units are in cm^{-3}.

stable high gains. Due to differences in geometry, the two-dimensional simulations yield lower gains overall when compared to those achieved in the one-dimensional simulations, despite having comparable doping configurations. It can be concluded that the δ-layers play a pivotal role in significantly enhancing gain in each BJT simulation case, independent of the variations in doping densities of the base (N_B), emitter implant ($N_{\mathrm{Emit\ Impl}}$), and collector implant ($N_{\mathrm{Col\ Impl}}$). Simulations incorporating δ-layers consistently achieved maximum gains that were at least an order of magnitude higher compared to configurations without δ-layers, underscoring their substantial impact on device performance.

6. Summary. This work seeks to understand the impact of δ-layer incorporation on the current gain of a two-dimensional APAM BJT device at room temperature. The study is conducted using semi-classical TCAD simulations. A one-dimensional device model composed of highly doped 2D regions known as δ-layers is simulated first to investigate the

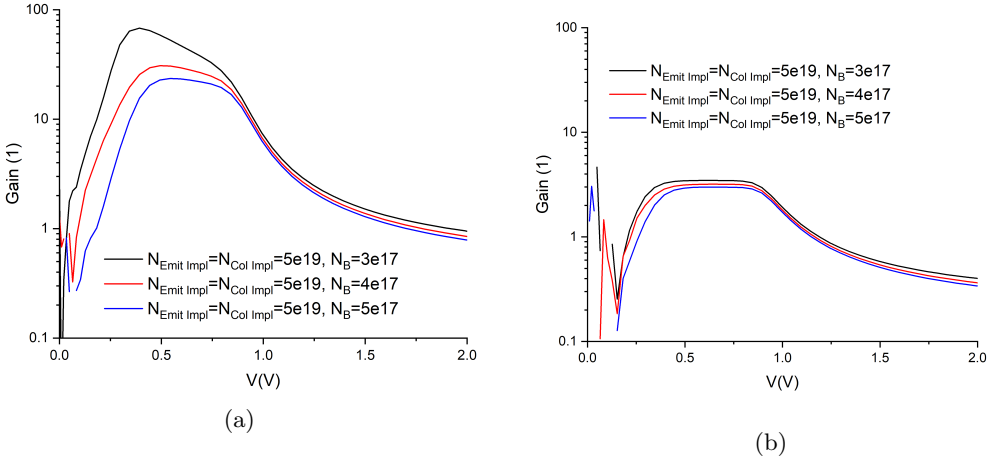


Fig. 4.3: Study of the effect of the δ-layer on the current gain. Simulated current gain for the common-emitter BJT configuration shown in Fig. 4.1 as a function of the emitter-base voltage (V_{EB}) and the emitter-collector voltage (V_{EC}), where V=V_{EB}=V_{EC}, for different base doping values with δ-layers in (a) and without δ-layers in (b). The units are in cm^{-3}.

impact of the δ-layers on the current gain. Building on insights gained from the 1D simulations, the incorporation of these planar structures into a larger, more complex device environment is investigated through simulations deriving the current gain of a δ-layer equipped two-dimensional BJT for various doping densities. Overall, the δ-layers play a pivotal role in significantly enhancing gain. It is found that the maximum gain is achieved by doping the emitter δ-layer sufficiently high and the base substrate appropriately low. It is also found that the collector δ-layer doping density does not significantly influence the gain.

7. Future Research. APAM's potential for beyond-Moore and quantum computing applications offers a significant opportunity for the continued miniaturization of transistor devices and the advancement of microelectronics as a whole. Investigating the performance of increasingly complex and practical APAM device prototypes will deepen the understanding of their physics and potential applications. The next essential step is to explore more devices fabricated at the atomic scale to achieve better performance, enhanced signal amplification, and faster switching times. While previous works, including an exploration of an APAM-based vertical 2D-2D tunnel FET [13], have primarily operated at cryogenic temperatures, further research is needed to assess the operation of APAM devices under room temperature conditions for practical applications. This study provides preliminary insights into the gain amplification capabilities of a novel two-dimensional BJT equipped with APAM δ-layers, but continued optimization is necessary to fully demonstrate its performance potential.

Acknowledgement. *This research was supported in part by an appointment to the National Nuclear Security Administration Minority Serving Institutions Internship Program (NNSA-MSIIP), sponsored by the National Nuclear Security Administration and administered by the Oak Ridge Institute for Science and Education.*

REFERENCES

[1] M. M. Waldrop. *The Chips are Down for Moore's Law*. *Nature*, 530 (7589):42, 2016.

[2] T.-M. Lu, X. Gao, E. M. Anderson, J. P. Mendez Granado, D. M. Campbell, J. A. Ivie, S. W. Schmucker, A. D. Grine, P. Lu, L. A. Tracy, R. Arghavani, and S. Misra. *Path towards a vertical TFET enabled by atomic precision advanced manufacturing*. United States, Jun 2021.

[3] D. R. Ward, S. W. Schmucker, E. M. Anderson, E. Bussmann, L. Tracy, T.-M. Lu, L. N. Maurer, A. Baczewski, D. M. Campbell, M. T. Marshall, and S. Misra. *Atomic precision advanced manufacturing for digital electronics*. *EDFAAO*, 22(1):4–10, 2020.

[4] C. K. Ober, H. Xu, V. Kosma, K. Sakai, and E. P. Giannelis. *EUV photolithography: resist progress and challenges*. In K. A. Goldberg, editor, *Extreme Ultraviolet (EUV) Lithography IX*, volume 10583, page 1058306. International Society for Optics and Photonics, SPIE, 2018.

[5] K. E. J. Goh, L. Oberbeck, M. Y. Simmons, A. R. Hamilton, and M. J. Butcher. *Influence of doping density on electronic transport in degenerate Si:P δ-doped layers*. *Phys. Rev. B*, 73:035401, Jan 2006.

[6] L. Oberbeck, N. J. Curson, M. Y. Simmons, R. Brenner, A. R. Hamilton, S. R. Schofield, and R. G. Clark. *Encapsulation of phosphorus dopants in silicon for the fabrication of a quantum computer*. *Applied Physics Letters*, 81(17):3197–3199, 10 2002.

[7] E. Crane, A. Kölker, T. Z. Stock, N. Stavrias, K. Saeedi, M. A. W. van Loon, B. N. Murdin, and N. J. Curson. *Hydrogen resist lithography and electron beam lithography for fabricating silicon targets for studying donor orbital states*. *Journal of Physics: Conference Series*, 1079(1):012010, Aug 2018.

[8] D. Mamaluy, J. P. Mendez, X. Gao, and S. Misra. *Revealing quantum effects in highly conductive δ-layer systems*. *Communications Physics*, 4(1):205, Sep 2021.

[9] G. J. Snyder and E. S. Toberer. *Complex thermoelectric materials*. *Nature Materials*, 7(2):105–114, Feb 2008.

[10] Charon — charon.sandia.gov. `https://charon.sandia.gov/`, [Accessed 11-10-2024].

[11] The Cubit® Geometry and Mesh Generation Toolkit — cubit.sandia.gov. `https://cubit.sandia.gov/`, [Accessed 11-10-2024].

[12] ParaView - Open-source, multi-platform data analysis and visualization application — paraview.org. `https://www.paraview.org/`. [Accessed 11-10-2024].

[13] X. Gao, J. P. Mendez, T. M. Lu, E. M. Anderson, D. M. Campbell, J. A. Ivie, S. W. Schmucker, A. Grine, P. Lu, L. A. Tracy, R. Arghavani, and S. Misra. *Modeling and Assessment of Atomic Precision Advanced Manufacturing (APAM) Enabled Vertical Tunneling Field Effect Transistor*. In *2021 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 102–106, 2021.

II. High Performance & Post-Moore Computing

Articles in this section discuss the implementation of software for high performance computing (HPC) or development of new types of scientific computing. In many cases, performance improvements and portability are demonstrated for many-core heterogenous architectures, such as conventional multicore CPUs, the Intel Many Integrated Core coprocessor (MIC), and graphical processing units (GPUs). Software and algorithms for non-traditional hardware such as SmartNICs or quantum computers offer other opportunities to advance scientific computing in the post-Moore era.

1. *A. Alvey-Blanco, K. Liegeois* and *B. Kelley* Toward Automatic Kernel Fusion for Kokkos Using MLIR
2. *C.N. Avans, J. Ciesko, C. Pearson, E.D. Suggs, S.L. Olivier,* and *A. Skjellum* Performance Insights into Supporting Kokkos Views In The Kokkos Comm MPI Library
3. *N. Bacon, S. Levy, P. Bridges,* and *K.B. Ferreira* Analysis of Modern Tools for Communication Impacts
4. *A. Epperly, K. Thompson,* and *O. Parekh* Sum of Squares Bounds on the Performance of the Quantum Approximate Optimization Algorithm
5. *A. Krishna* and *R. Milewicz* Experience Report on Observability and its Effect on Security and Usability in Software Systems
6. *C. O'Neil, M.D. Porter,* and *S.K. Seritan* Analyzing Qubit-runtime Tradeoffs in Parallelizing Unary Iteration
7. *J. Shawger* and *M.L. Curry* Storage System Characterization in Virtualized Testbed
8. *N.D. Siekierski, A.Q. Wilber-Gauthier,* and *S.K. Seritan* Scalable Application-Oriented Benchmarking of Quantum Computers

M. Adams
T. Casey
B.W. Reuter

December 17, 2024

TOWARD AUTOMATIC KERNEL FUSION FOR KOKKOS USING MLIR

ADDISON ALVEY-BLANCO *, KIM LIEGEOIS †, AND BRIAN KELLEY ‡

Abstract.
Heterogeneous compute platforms have become pervasive in HPC. This has created a demand for programming frameworks that promise performance portability so that developers can avoid writing many versions of the same code and lessen the difficulty of programming for accelerators. Typically, these frameworks dispatch a single API call to many different versions of a kernel so that a particular kernel implementation is selected for a given architecture. Developers string these API calls (kernels) together in a specific way to implement their programs. Since the provided kernels are typically primitives, this causes an explosion of kernel calls. The increase of kernel calls increases the launch overhead and misses data reuse opportunities across kernels. Kernel fusion can be used to avoid these pitfalls. Automatic kernel fusion has been an area of research for some time, but no approach has been widely adopted. This work presents a methodology for automatically fusing kernels in a user-driven, platform-agnostic way. We present results obtained by generating code for Kokkos, a popular performance portability framework, and relying on Kokkos and vendor-supplied compilers to determine platform-specific details after kernel fusion has been performed.

1. Introduction. The rise of accelerators and other specialized hardware have forced new and existing software stacks to support heterogeneous compute platforms. The Department of Energy compute platforms alone are made up of devices from each of the three major vendors: NVIDIA, Intel, and AMD. Each of these vendors have their own architectures, their own compilers, their own implementations of libraries like BLAS [1, 2, 4, 5, 6], and their own language implementations. NVIDIA uses CUDA, AMD uses HIP, and Intel uses SYCL. OpenCL exists as a one-size-fits-all language for running on all three vendors' devices, but there may not be a one-to-one mapping from, say, CUDA primitives to OpenCL requirements. Therefore, the same code written in OpenCL and CUDA may have differences in performance. Hence, developers are left with a choice: 1. maintain a separate version of their code for each compute platform, or 2. deal with drawbacks incurred from using a one-size-fits-all approach.

Performance portability frameworks like Legion [7], RAJA [8], and Kokkos [9] allow developers and users of scientific software stacks to "write once, run anywhere". In general, performance portability frameworks abstract away architecture-specific parallelization primitives, allowing users to focus on implementing their algorithms in a generic way. Based on the target architecture specified by a user, these frameworks generate architecture-specific code. Our target portability framework is Kokkos. There is an extension to Kokkos called Kokkos Kernels [25], focused on BLAS and sparse linear algebra kernels [23], that dispatches a single API call to various kernel implementations that are tuned for a particular architecture.

Loss of control over optimizations is a consequence of using libraries of linear algebra primitives. An example that illustrates this is $b = Ax + y$, with A a matrix of size $n \times n$, x a vector of size n, and y a vector of size n. While there are typically routines to compute $z = Ax$, i.e. GEMV in BLAS, and $b = z + y$, i.e. AXPY in BLAS, a routine that computes both operations as a single library call typically does not exist. The locality of the program is reduced as a result. Specifically, for small inputs, as is the case in machine learning applications and some scientific applications, we can compute a single entry, b_i, in cache and eliminate the need to materialize the intermediate variable z. Kernel fusion [14, 15, 16, 20, 22, 24, 26], inspired by loop fusion [12, 17], is an optimization technique

*Sandia National Laboratories, and University of Illinois at Urbana-Champaign, ajalvey@sandia.gov
†Sandia National Laboratories, knliege@sandia.gov
‡Sandia National Laboratories, bmkelle@sandia.gov

that aims to increase data locality and decrease kernel launch overhead. In our example, $b = Ax + y$, we can fuse the kernel computing $z = Ax$ and the kernel computing $b = z + y$ to increase locality.

Current vendor-supplied compilers do not support kernel fusion, so we need external solutions. One possibility is to use a compiler other than a vendor-supplied compiler. However, this risks missing optimizations performed by a vendor's compiler. Another option is to implement kernel fusion in a source-to-source compilation pipeline. By doing this, we can perform kernel fusion and still take advantage of architecture-specific optimizations implemented by a vendor's compiler.

The overall contribution of this work is an attempt at solving the problem of kernel fusion in a platform-agnostic manner. Specifically, we rely on users to supply information about what to fuse as a replacement for a profitability model. We then use the user-supplied information to partition a graph of kernel calls, with each partition representing a fused kernel. For our compiler choice, we use the Multi-Level Intermediate Representation (MLIR) compiler framework [21] due to its many levels of abstraction.

In section 2, we provide background on loop and kernel fusion. In section 3, we describe our approach to the kernel fusion problem. In section 4, we show the results of our method for kernel fusion on two examples. The first is a reference gradient computation commonly found in the spectral element method (SEM), the second is a batched matrix-vector multiplication and a vector addition. In section 5, we discuss the performance of our method compared to our expectations and directions for future work. In section 6, we conclude by providing a summary of our method and future work.

2. Background. Kernel fusion is a special case of loop fusion. Specifically, a computational kernel is a typed, perfectly nested set of parallel loops. A perfectly nested loop is a loop in which there are no instructions between loop headers, i.e. all instructions are contained in the innermost loop. Fusing kernels involves fusing the parallel loops the kernels represent. To this end, we begin with background on loop fusion.

2.1. Loop fusion. Loop fusion is a loop transformation that is used to increase data locality in a program [12, 17]. Loop fusion combines two or more compatible loops into a single loop. By creating a single loop, temporary variables storing intermediate results can be eliminated. This can allow intermediate results to stay in cache or in registers, thereby increasing temporal locality of the program.

The problem of loop fusion is more general than that of kernel fusion. Specifically, kernels typically do not interfere with a program outside of the scope of the kernel definition. A loop, on the other hand, can alter variables and program state outside of the scope of the loop body. Moreover, a kernel call is typed by the arguments passed and the results returned. As a result, it is trivial to determine dependences between two or more kernel calls. More analysis is required to determine the common variables between two or more loop nests since the information is not readily available.

2.2. Kernel fusion. Kernel fusion is a code transformation that aims to (1) increase data locality by combining two or more kernel definitions, and (2) reduce kernel launch overhead by reducing the total number of launched kernels. Since separate, data-dependent kernel invocations are typically run sequentially, one kernel must store a result in global memory and any dependent kernels must load the result from global memory. Kernel fusion, if legal to do, allows the data to be stored and accessed from fast memory. Kernel fusion is not always possible and it does not always result in a performance increase.

Since manually fusing kernels can miss optimization opportunities, there have been a number of works on automatic kernel fusion [14, 15, 16, 20, 22, 24, 26]. In [20, 24, 26], the

problem of kernel fusion is set up as graph partitioning problem. The graph is a directed acyclic graph of kernel calls, i.e. a dependency graph, with edges representing a read-write dependency between kernels. A directed edge from kernel A to kernel B means kernel A must execute before kernel B.

In [26], the graph partitioning problem is setup so as to minimize the total runtime of the program. A projected runtime is computed for each subkernel and a fused kernel created from the same subkernels. If fusion does not violate dependency constraints and the sum of the projected subkernel runtimes is greater than the projected fused kernel runtime, then the fused kernel is used. This approach requires an accurate runtime model which might not be available for all architectures targeted by a performance portability framework. The approach taken by [14] is similar, but pre-existing benchmarks for each subkernel are used in place of a performance model. This would require a significant amount of work if the benchmarks are not available for the possible target architectures at compile time. The approach taken by [24] computes edge weights by estimating locality improvement using domain- and architecture-specific knowledge and solves the resulting graph partitioning problem using minimum cut.

The implementation in [20] takes advantage of the widely used tools LLVM and Clang. A decorator is provided by the user for kernels that should be considered for fusion. However, users cannot specify specific kernels with which a particular kernel should be fused. The resulting fusion problem in [20] is solved via a topological sort of the DAG generated from the annotated kernels.

The proposed method in [15] utilizes a dataflow analysis method based on [18, 19] using a control-flow graph (CFG) of kernels. However, the results shown are only using CUDA [3] kernels.

2.3. Multi-Level Intermediate Representation (MLIR) framework. MLIR is a compiler infrastructure framework in the LLVM project. It provides various levels of abstraction in the intermediate representation (IR). Certain transformations can only be applied at specific levels of abstraction. Moving through the levels of abstraction is referred to as "lowering" or "raising" the IR. The levels of abstraction are referred to as dialects.

As an example, the Linalg dialect in MLIR represents linear algebra primitives and generic linear algebra operations. The Linalg dialect has a natural connection to the Tensor dialect, representing abstract, non-bufferized multidimensional arrays, and the Structured Control-Flow (SCF) dialect, representing control-flow constructs like if-statements and for loops. MLIR also provides various dialects that target certain devices and frameworks, i.e. the GPU dialect and LLVM dialect.

2.4. Linear Algebra Performance through Intermediate Subprograms (LAPIS). LAPIS is a source-to-source compiler focused on code transformations for sparse linear algebra. There is a utility for generating Kokkos C++ code from MLIR input. The work in this paper utilizes the existing MLIR-to-Kokkos translation tool in LAPIS. Our work directly contributes to the optimization pipeline in LAPIS, which had limited capabilities prior to the work in this paper. There are plans to release LAPIS as an open-source project in the future.

3. Approach. In this section, we provide the details of our proposed solution to the problem of fusing kernels automatically in a platform-agnostic way. In a similar way to previous works, we view kernel fusion as a partial graph partitioning problem. To generate the initial graph, we rely on the user to provide information about what kernels to fuse. We do not guarantee that all requested kernels are fused together.

3.1. Overview. We assume all kernel calls are made from a main routine. Further, we assume all kernel definitions and the main routine are available in the same module. Finally, we assume no aliasing occurs. These assumptions only need to hold for the IR that our kernel fusion routine will process. Since our main focus is on scientific computing and machine learning applications, we assume the kernel definitions represent some linear algebra operations.

As an overview of the process, we first build a graph of kernel calls found in the IR of the original program. We then process the graph and user-provided metadata to create *fusion sets*. Fusion sets are collections of kernel calls that are to be fused together into a single call. Once all fusion sets have been created, we remap arguments and results to make sense in the context of the fused kernels and remove previously materialized intermediate results. Once we have created all of the fused kernels, we then lower the new IR to Kokkos C++ using the LAPIS translation tool. At this point, we rely on Kokkos to generate architecture-specific code and various vendor-supplied compilers to further optimize the generated code.

3.2. Graph creation. To begin, we gather all calls found in the main routine. An edge is inserted between two kernels if there is an input dependence or a true dependence. An input dependence means two kernels load the same input data. Fusing two kernels with input dependence can reduce the number of times the common input data is retrieved from main memory. A true dependence means there is some intermediate data that must be materialized in one kernel and loaded in the another. By fusing, we can eliminate the need to materialize intermediates. We rely on the order-preserving property of MLIR to ensure kernels are processed in the proper order.

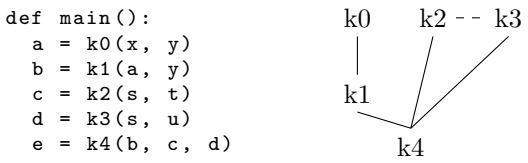


Fig. 3.1: Example of a graph generated from kernel calls

An example of source code and the corresponding graph are included in Figure 3.1. Since there is a read-write dependence between kernel 0 and kernel 1, there is an edge connecting them. Similarly, there is a read-write dependence between kernel 1 and kernel 4, kernel 2 and kernel 4, and kernel 3 and kernel 4 represented by an appropriate edge in the graph. Finally, the input dependence is shown as a dashed line from kernel 2 to kernel 3 since input dependence does not constrain the semantics of the program.

3.3. Generating fusion sets. Once we have generated a graph of kernel calls, we then process the graph and user-provided metadata to partition the graph. We create sets of kernel calls (nodes) called fusion sets. Edges between calls are replaced with edges between fusion sets.

First, we check that each kernel is legal to fuse with all other kernels in the fusion set. For legality checking, we utilize a modified version of the existing parallel loop fusion legality checking in MLIR. In particular, it is ensured that dependences do not exist across loop iterations, and that the parallel iteration spaces of the kernel definitions match. A kernel call is added to a fusion set if it is legal to fuse and the user has requested it should be fused with all other calls in the fusion set.

Following our example above, fusion sets correspond to partitions of the graph. In

Figure 3.2, fusion sets are represented by dashed circles around a subset of the nodes of the graph.

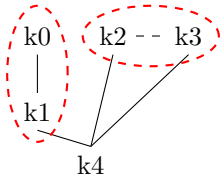


Fig. 3.2: Example of fusion sets

3.4. Fusing kernels. Lastly, we fuse all kernels in a fusion set together. We first move all kernel calls in a fusion set out of the main function and into a new function that will become the fused kernel. A call to the fused kernel is placed below the final subkernel call in the main function. Arguments and results are remapped to the fused kernel call, removing duplicates and unnecessary materialization. Once we have finished processing the fusion sets, we lower the kernels via a sequential process. First, we inline each subkernel into the fused kernels. Once all subkernels in all fused kernels have been inlined, we fuse the parallel loops of the inlined subkernels. Finally, once all parallel loop fusion is completed, we delete the subkernel definitions if they are unused elsewhere.

Figure 3.3 illustrates the fused version of our original example and the corresponding graph.

```
def main():
  b = k01(x, y)
  c, d = k23(s, t, u)
  e = k4(b, c, d)
```

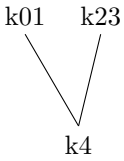


Fig. 3.3: Code & graph result of fusion

4. Results & Discussion. The results below are gathered by generating Kokkos C++ code from the MLIR containing the fused kernels and choosing a vendor-supplied compiler based on the device we are targetting. All of the results are compiled with all optimizations on, i.e. using `-O3`.

Execution times collected are an average over 1000 runs of each point in the figures below. In between each of the 1000 runs, the GPU cache is flushed by initializing a 512 MB array on the GPU. The initialization time is not included in the measured time for each point in the figures.

4.1. Finite Element Reference Gradient Computation.

4.1.1. Motivation. A common operation in finite-element methods is to apply an operator, typically a mass or stiffness matrix, to degrees-of-freedom (DOFs). The application of the operator can be decomposed into a sequence of single-axis tensor contractions if a tensor-product discretization is used. The tensor contraction method reduces the computational cost from $O(n^{2d})$ to $O(n^{d+1})$, where d is the physical dimension and n are the number of DOFs in a single element. This is commonly used in the spectral element method (SEM). For an for an overview, see Chapter 4 of [11].

Device	L1 Cache Size (KB)	L2 Cache Size (MB)
MI250	16 KB per CU	16 MB
H100	256 KB per SM	50 MB
V100	128 KB per SM	6 MB
Ponte Vecchio	64 KB per EU	418 MB

Fig. 4.1: L1 and L2 cache sizes for various GPU devices

4.1.2. Example: Reference Gradient. We apply a reference differentiation operator to DOF data of a tensor-product discretization to generate the reference gradient. If we wanted the full gradient, we would need to apply metric terms to the reference gradient. This is used in, for example, [10, 27]. The purpose of this example is to highlight fusion benefits when there is only input dependence, so we do not apply metric terms.

Suppose we have a 3D tensor-product discretization with n_d degrees of freedom per dimension, n_e elements. Then, we have a total of n_d^3 DOFs, and a rank-4 tensor of DOFs with size $n_e \times n_d \times n_d \times n_d$. Given a differentiation operator D of size $n_d \times n_d$, we can express the reference partial derivatives as tensor contractions over each of the DOF axes of the DOF tensor with D. In particular,

$$\left(\frac{\partial u}{\partial r}\right)^e_{ijk} = \sum_{\ell}^{n_d} D_{i\ell} u^e_{\ell jk}$$
$$\left(\frac{\partial u}{\partial s}\right)^e_{ijk} = \sum_{\ell}^{n_d} D_{j\ell} u^e_{i\ell k}$$
$$\left(\frac{\partial u}{\partial t}\right)^e_{ijk} = \sum_{\ell}^{n_d} D_{k\ell} u^e_{ij\ell},$$

where u^e_{ijk} denotes the (i, j, k)-th DOF of the e-th element.

A naive implementation would involve calling a separate kernel for each partial derivative. This implementation would suffer from poor data locality. Specifically, the operator D never changes and can stay in fast memory. By calling three separate kernels, we are loading the $n_d \times n_d$ operator D from global memory three times when we only need to do it once. Additionally, we lose data reuse in the DOFs by not writing all three reductions in a single kernel. We utilize our kernel fusion tool described above to fuse all three partial derivative computations into the same kernel. In our implementation, we parallelize over the elements.

The results gathered for this example are shown in Figures 4.2, 4.3 and 4.4. Figure 4.2 shows results gathered while varying the element count while Figure 4.3 shows results gathered while varying the DOF count. The plots show the fused over unfused ratio and raw measurements of L1 hit rate and execution time measured in seconds. In both figures, we can see that a larger ratio of L1 hit rates corresponds to a smaller ratio of execution times. This is to be expected since an increased L1 hit rate means the number of global memory accesses is reduced, hence saving on GPU cycles. In the case where element count is varied, a constant DOF count of 3 per dimension, i.e. 27 total DOFs, is used. In the case where the DOF count is varied, a constant element count of 128×10^3 is used.

In Figure 4.2, we can see that fusion improves performance for varied element counts. Since we are parallelizing over the elements, we expect the benefit of fusion to be constant for all element counts. However, this is not the case for the smallest element count on the

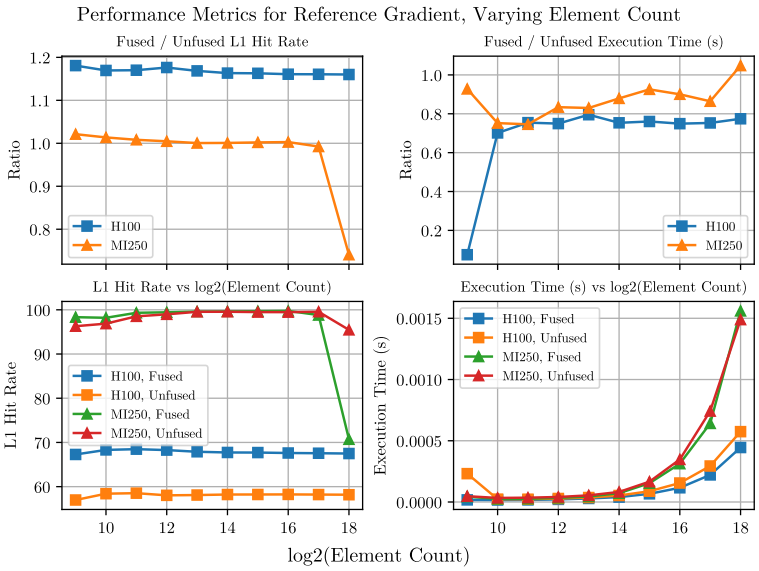


Fig. 4.2: L1 hit rates and execution times of reference gradient example on MI250 and H100 with varied element count

H100 and for the MI250 overall. This is due to architectural differences like the size of the L1 and L2 cache on each of the devices.

The behavior in Figure 4.3 shows how the number of DOFs in each element dictates fused performance. We can see that fusion results in an increased L1 hit rate only for the H100. The fused L1 hit rate is reduced to about 80% of the unfused L1 hit rate on the MI250. This is due to the small L1 cache of the MI250. The H100 has 256KB of L1 cache per streaming multiprocessor (SM) while the MI250 only has 16KB of L1 cache per compute unit (CU). Despite the increased H100 L1 hit rate across all DOF counts, the fused execution time is greater than the unfused execution time as the DOF count increases. Since we are parallelizing over elements, threads will begin to compete for L1 cache space. Hence, depending on the L1 cache space, fusion can be more or less beneficial.

We also gathered execution times for a Ponte Vecchio GPU and V100 GPU. We were unable to collect cache hit rate data due to issues with the available profiling tools. In Figure 4.4, the relative speed-up and execution time on all tested architectures is included. A negative value on the y-axis corresponds to a slowdown. We can see that the MI250 suffers more than any other device in Figure 4.4. However, other devices also have decreased performance when fused. If we compare the performance gain to the size of the L1 cache, we can see that a large L1 cache typically corresponds with a larger-by-comparison increase in performance. The Ponte Vecchio GPU is an exception, with the third smallest L1 cache. The large L2 cache makes up for the lack of L1 cache space since fetching from L2 to L1 cache is faster than from global memory to L2 cache.

4.2. Batched GEMV + AXPY.

4.2.1. Motivation. Batched operations have become popular in part due to machine learning applications. There have been efforts to create a standardized interface for batched BLAS routines [13]. Batched linear algebra operations are also relevant to scientific computing applications. For example, we can express the reference gradient computation above

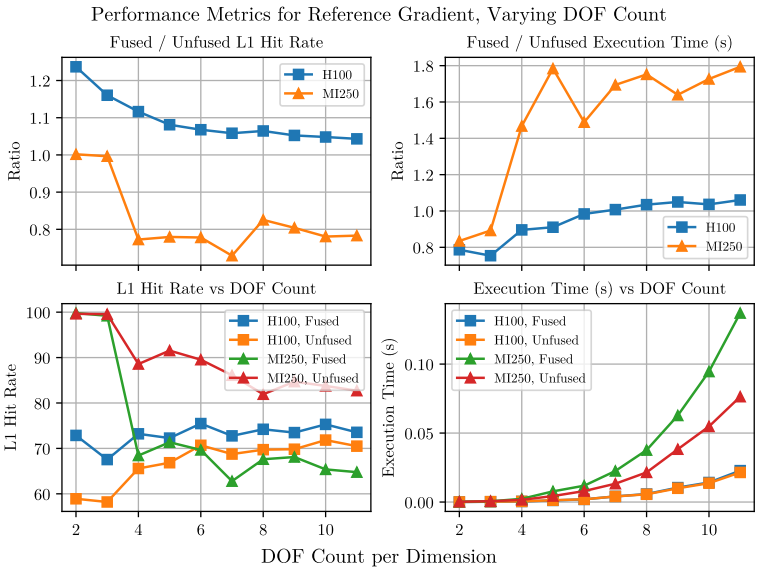


Fig. 4.3: L1 hit rates and execution times of reference gradient example on MI250 and H100 with varied DOF count

as a batched matrix-times-vector with batch size equal to the dimension, d. Specifically, if we do not exploit a tensor-product discretization, and we are working in $d = 3$ dimensions, then we need three differentiation operators. We can organize the DOFs as a matrix of size $n_e \times n_d^3$. We can express each entry of the gradient as a batched matrix-times-vector operation

$$\nabla u^e_{r,i} = \sum_j^{n_d^3} D_{r,ij} u^e_j, \text{ for } r = 1, \ldots, d$$

4.2.2. Example: Batched GEMV + AXPY. In this example, we consider a general batched matrix vector multiplication and a vector addition, where both the matrix and the vector have a batch axis. Specifically, our example is to implement

$$z_{\ell i} = \sum_j^n A_{\ell,ij} x_{\ell,j} + y_{\ell,i}, \text{ for } \ell = 1, \ldots, b,$$

with a batch size of b, A a 3-tensor of size $b \times n \times n$, and x, y, and z three matrices of size $b \times n$. The unfused implementation involves two kernel calls: one kernel call is responsible for the batched $c = Ax$, and the second kernel call is responsible for the batched $z = c + y$. The results collected for this experiment are shown in Figures 4.5, 4.6, and 4.7.

The H100 results found in Figure 4.5 show that varying the batch size has little impact on the L1 hit rate, and thus does not reduce the execution time. This is not the same for the MI250 results in Figure 4.6. The difference in the results can be explained, again, by the difference in the size of the L1 caches between the two GPUs.

In the case where the input matrix size is varied, we can see that fusion has a larger impact on the performance. A gradual increase in execution time and decrease in L1 hit

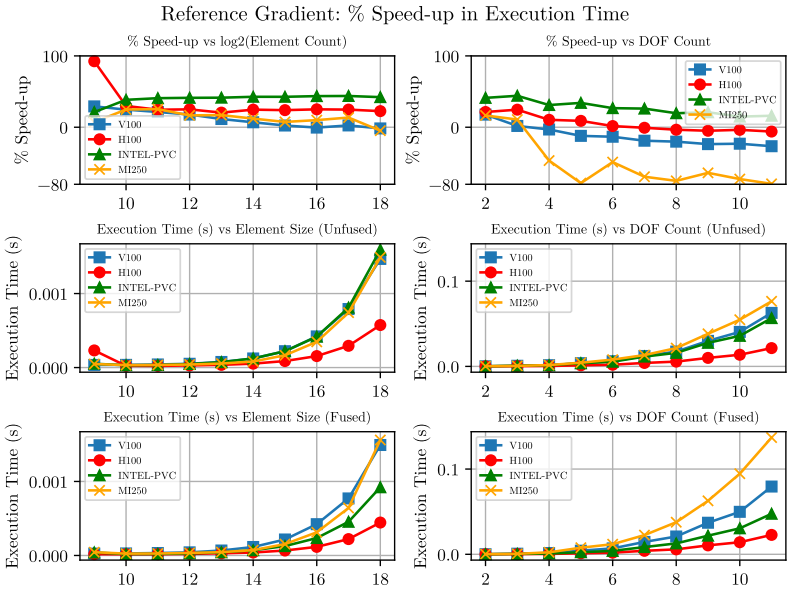


Fig. 4.4: Reference Gradient: Execution time comparisons on V100, H100, MI250, and Intel Ponte Vecchio

rate as the input size increases can be observed. Since we are parallelizing over the batches, the input data for each batch must fit in cache. As the input size increases, available space in the L1 cache becomes scarce.

As in the reference gradient example, we tested on a variety of architectures but could only collect L1 hit rate metrics for the H100 and MI250. We were able to collect execution times for the H100, MI250, V100, and Intel Ponte Vecchio. Those results are shown in 4.7.

In Figure 4.7, we can see that kernel fusion benefits all architectures at all batch sizes and almost all input sizes. However, this is not the case for the reference gradient results in Figure 4.4. This leads us to conclude that automatic kernel fusion in the context of performance portability has a benefit for some, but not all, problems. While our user-driven approach aims to address this, it is not effective if a user lacks the necessary knowledge. Hence, our method requires a way of determining when fusion could be detrimental to performance in the context of performance portability. All devices except the MI250 saw at least some performance gain from kernel fusion in our micro-benchmarks. The MI300, which was not available for testing, has twice as much L1 cache storage than the MI250. So, the story may not be the same for the MI300 as for the MI250. To adjust for various cache sizes, loop tiling could be used to increase locality within a fused kernel. Since tiling needs to be performed in a platform-agnostic way, this is a goal for future work.

4.3. Discussion. Overall, we have shown our implementation of kernel fusion is able to achieve expected results across architectures from various vendors. In particular, we show that kernel fusion results in an increase in L1 cache hit rates. This results in a decrease in the runtime of the fused kernel compared to the sum of the runtimes of the subkernels. We can see that there is a performance benefit across various architectures. This illustrates that kernel fusion can be implemented as a platform-agnostic code transformation. However, we notice a performance degradation in cases where the L1 cache is not large enough to

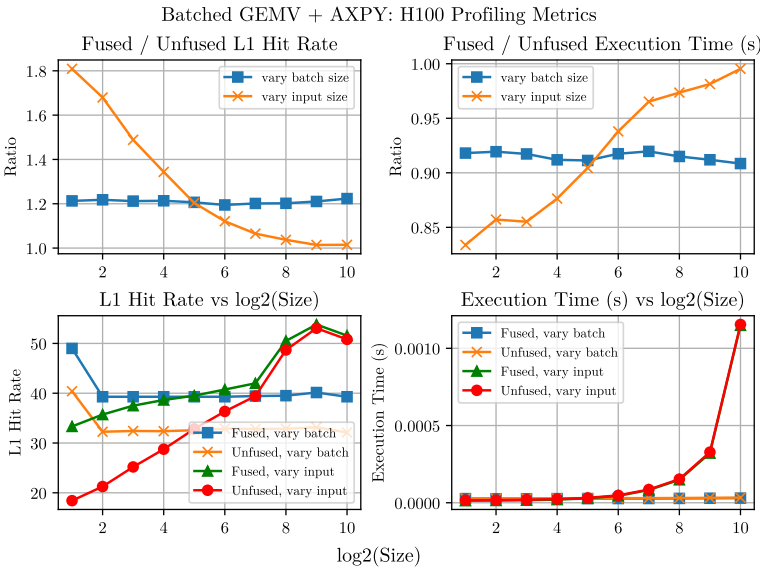


Fig. 4.5: H100 L1 hit rate and execution time metrics for batched GEMV + AXPY example

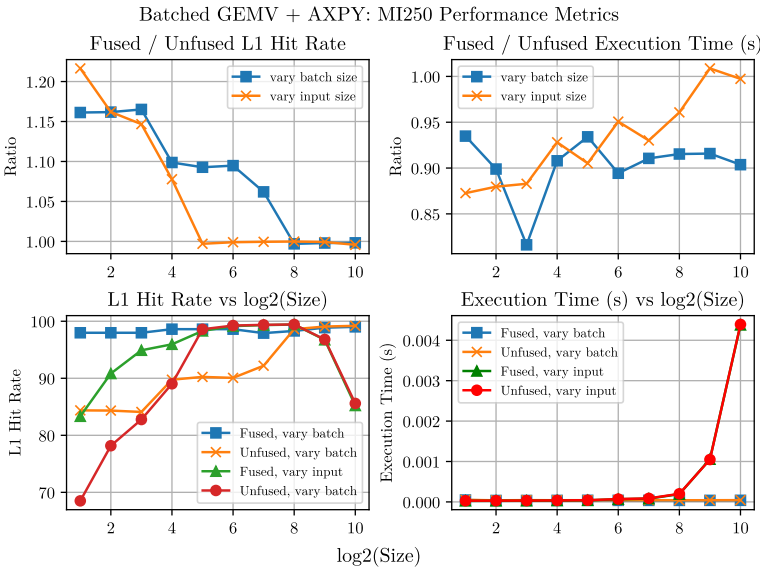


Fig. 4.6: MI250 L1 hit rate and execution time metrics for batched GEMV + AXPY example

support the amount of intermediate data. Additional transformations, such as tiling, may be necessary to improve the use of the L1 cache. This begs the question of whether other code transformations, like tiling, can be effectively applied with no knowledge of the target architecture.

Another question that is being investigated as an extension of this work is how to predict performance without knowledge of the target architecture. One replacement for

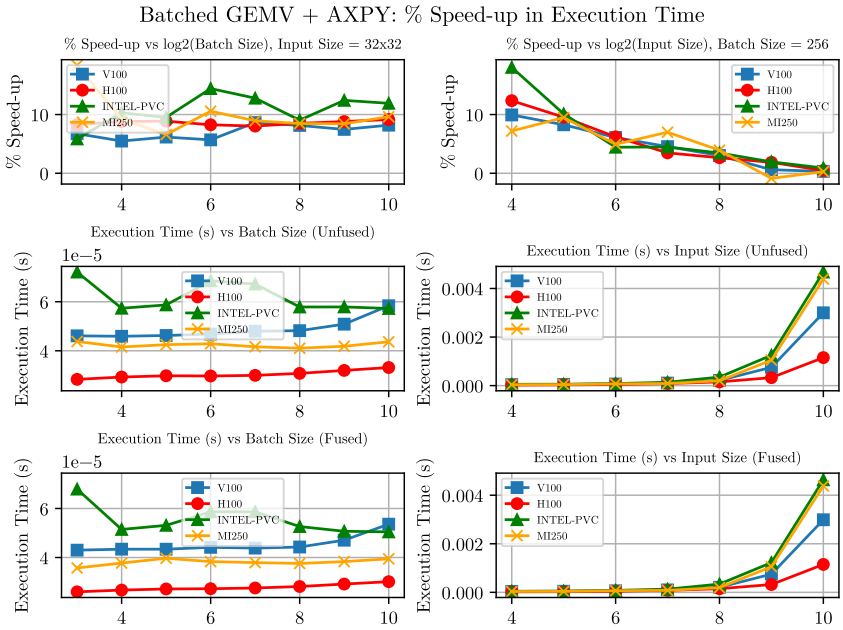


Fig. 4.7: Batched GEMV + AXPY: Execution time comparison for various devices with varied batch and input sizes

a performance model we are actively looking at is asymptotic computational cost. The overarching graph partitioning problem in kernel fusion could then be solved such that computational cost is minimized.

5. Conclusion. We have presented a platform-agnostic method for fusing kernels to improve data locality in a program. Our approach relies on user-provided metadata to choose candidate kernels for fusion. We have shown that our method achieves the expected results on a variety of architectures. A weakness in the current implementation is that if a user provides no input, then no kernels will be fused. While this is intentional, it can be off-putting to users who may lack required knowledge for useful fusion combinations. Hence, a topic of future work for this project is fusion profitability criteria that is platform-agnostic. One possibility is to utilize the computational cost of the (linear algebra) operations represented by the kernels.

REFERENCES

[1] *BLAS and Sparse BLAS Routines.*
[2] *cuBLAS.* Archive Location: CUDA API References.
[3] *CUDA C++ Programming Guide.*
[4] *cuSPARSE Library.*
[5] *rocBLAS documentation — rocBLAS 4.2.0 Documentation.*
[6] *rocSPARSE documentation — rocSPARSE 3.2.0 Documentation.*
[7] M. Bauer, S. Treichler, E. Slaughter, and A. Aiken, *Legion: Expressing locality and independence with logical regions*, in 2012 International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, Nov. 2012, IEEE, pp. 1–11.
[8] D. A. Beckingsale, J. Burmark, R. Hornung, H. Jones, W. Killian, A. J. Kunen, O. Pearce, P. Robinson, B. S. Ryujin, and T. R. Scogland, *RAJA: Portable Performance for Large-Scale*

Scientific Applications, in 2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC), Nov. 2019, pp. 71–81.

[9] H. Carter Edwards, C. R. Trott, and D. Sunderland, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, Journal of Parallel and Distributed Computing, 74 (2014), pp. 3202–3216.

[10] J. Chan, Z. Wang, A. Modave, J.-F. Remacle, and T. Warburton, *GPU-accelerated discontinuous Galerkin methods on hybrid meshes*, Journal of Computational Physics, 318 (2016), pp. 142–168.

[11] M. O. Deville, P. F. Fischer, and E. H. Mund, *High-Order Methods for Incompressible Fluid Flow*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2002.

[12] C. Ding and K. Kennedy, *Improving effective bandwidth through compiler enhancement of global cache reuse*, in Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001, Apr. 2001, pp. 10 pp.–. ISSN: 1530-2075.

[13] J. Dongarra, I. Duff, M. Gates, A. Haidar, S. Hammarling, N. J. Higham, J. Hogg, P. V. Lara, P. Luszczek, M. Zounon, S. D. Relton, T. Costa, and S. Knepper, *Batched BLAS (Basic Linear Algebra Subprograms) 2018 Specification*.

[14] J. Filipovič, M. Madzin, J. Fousek, and L. Matyska, *Optimizing CUDA Code By Kernel Fusion—Application on BLAS*, The Journal of Supercomputing, 71 (2015), pp. 3934–3957. arXiv:1305.1183 [cs].

[15] J. Fukuhara and M. Takimoto, *Automated kernel fusion for GPU based on code motion*, in Proceedings of the 23rd ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems, San Diego CA USA, June 2022, ACM, pp. 151–161.

[16] T. Gysi, C. Müller, O. Zinenko, S. Herhut, E. Davis, T. Wicky, O. Fuhrer, T. Hoefler, and T. Grosser, *Domain-Specific Multi-Level IR Rewriting for GPU*, July 2020. arXiv:2005.13014 [cs].

[17] K. Kennedy and K. S. McKinley, *Maximizing loop parallelism and improving data locality via loop fusion and distribution*, in Languages and Compilers for Parallel Computing, G. Goos, J. Hartmanis, U. Banerjee, D. Gelernter, A. Nicolau, and D. Padua, eds., vol. 768, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 301–320. Series Title: Lecture Notes in Computer Science.

[18] J. Knoop, O. Rüthing, and B. Steffen, *Lazy code motion*, SIGPLAN Not., 27 (1992), pp. 224–234.

[19] ———, *Optimal code motion: theory and practice*, ACM Trans. Program. Lang. Syst., 16 (1994), pp. 1117–1155.

[20] A. Lamzed-Short, T. R. Law, A. Mallinson, G. R. Mudalige, and S. A. Jarvis, *Towards Automated Kernel Fusion for the Optimisation of Scientific Applications*, in 2020 IEEE/ACM 6th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC) and Workshop on Hierarchical Parallelism for Exascale Computing (HiPar), Nov. 2020, pp. 45–55.

[21] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko, *MLIR: Scaling Compiler Infrastructure for Domain Specific Computation*, in 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), Feb. 2021, pp. 2–14.

[22] A. Li, B. Zheng, G. Pekhimenko, and F. Long, *Automatic Horizontal Fusion for GPU Kernels*, July 2020. arXiv:2007.01277 [cs].

[23] K. Liegeois, S. Rajamanickam, and L. Berger-Vergiat, *Performance Portable Batched Sparse Linear Solvers*, IEEE Transactions on Parallel and Distributed Systems, 34 (2023), pp. 1524–1535. Conference Name: IEEE Transactions on Parallel and Distributed Systems.

[24] B. Qiao, O. Reiche, F. Hannig, and J. Teich, *From Loop Fusion to Kernel Fusion: A Domain-Specific Approach to Locality Optimization*, in 2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), Feb. 2019, pp. 242–253.

[25] S. Rajamanickam, S. Acer, L. Berger-Vergiat, V. Dang, N. Ellingwood, E. Harvey, B. Kelley, C. R. Trott, J. Wilke, and I. Yamazaki, *Kokkos Kernels: Performance Portable Sparse/Dense Linear Algebra and Graph Kernels*, Mar. 2021. arXiv:2103.11991 [cs].

[26] M. Wahib and N. Maruyama, *Scalable Kernel Fusion for Memory-Bound GPU Applications*, in SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2014, pp. 191–202. ISSN: 2167-4337.

[27] K. Świrydowicz, N. Chalmers, A. Karakus, and T. Warburton, *Acceleration of tensor-product operations for high-order finite element methods*, The International Journal of High Performance Computing Applications, 33 (2019), pp. 735–757. Publisher: SAGE Publications Ltd STM.

PERFORMANCE INSIGHTS INTO SUPPORTING KOKKOS VIEWS IN THE KOKKOS COMM MPI LIBRARY

C. NICOLE AVANS*, JAN CIESKO†, CARL PEARSON‡, EVAN DRAKE SUGGS §, STEPHEN L. OLIVIER¶, AND ANTHONY SKJELLUM‖

Abstract.
We introduce Kokkos Comm, a new Kokkos ecosystem library designed to address the challenges of integrating Kokkos, a C++ performance portability ecosystem, with distributed memory programming models. Kokkos Comm aims to alleviate the difficulties associated with coordinating non-blocking MPI operations and Kokkos execution spaces, as well as handling non-contiguous data structures represented by Kokkos::Views. Additionally, it serves as a platform for researching improved methods of managing non-contiguous data and exploring new communication APIs with performance portability across various underlying transports. This research aims to aid in the development of Kokkos Comm by providing insights to guide further performance-oriented development.

1. Background. Increasingly complex high-performance computing (HPC) architectures have led to a demand for the efficient integration of on-node and off-node performance-portable programming models. Kokkos [11], a C++ library for on-node parallel execution and memory abstraction, is a popular solution, enabling developers to write single-source applications that can efficiently utilize diverse CPU and GPU hardware platforms from various vendors. Presently, integration with MPI presents difficulties with supporting the main Kokkos data structure, the `Kokkos::View`, and coordination of non-blocking MPI communication. We introduce initial work on Kokkos Comm, a new library in the Kokkos Ecosystem designed to bridge the gap between Kokkos and distributed memory programming. The primary objectives of Kokkos Comm are threefold:

- Alleviate the challenges associated with combining Kokkos and MPI [4], focusing on the coordination of non-blocking MPI operations with Kokkos execution spaces and efficient handling of non-contiguous `Kokkos::Views`.
- Provide a platform to investigate improved methods of managing non-contiguous data in the context of Kokkos and MPI interactions.
- Serve as a testbed to research novel communication APIs in addition to MPI, such as MPIAdvance [1], libfabric [8], UCX [12], and NCCL [6].

2. KokkosComm Functionality. The Kokkos Comm prototype [1] establishes a framework for future research and development within the Kokkos ecosystem, particularly in the realm of efficient and flexible inter-node communication. This work also builds on and complements prior work and experience with integrating an MPI subset, ExaMPI, with Kokkos as described in Suggs et al. [10] [9].

Initial work in Kokkos Comm is primarily concerned with MPI interoperability. Figure 2.1 shows the template specification of Kokkos Comm's `MPI_Send` interface. Kokkos Comm's operations are ordered with respect to a Kokkos execution space instance, so many operations take both a Kokkos execution space instance and an MPI communicator as arguments. To allow Kokkos Comm to flexibly handle various shapes of multidimensional data,

*Tennessee Technological University, cnavans42@tntech.edu
†Sandia National Laboratories, jciesko@sandia.gov
‡Sandia National Laboratories, cwpears@sandia.gov
§Tennessee Technological University, esuggs@tntech.edu
¶Sandia National Laboratories, slolivi@sandia.gov
‖Tennessee Technological University, askjellum@tntech.edu
[1]Available at `https://github.com/kokkos/kokkos-comm`.

```
template<typename SendMode, KokkosExecutionSpace ExecSpace, KokkosView SendView>
void KokkosComm::Send(const ExecSpace &space, const SendView &sv, int dest, int tag
    , MPI_Comm comm);
```

FIG. 2.1. *Example Kokkos Comm specification*

TABLE 2.1
Feature Table

Feature	Current Capability
Point-to-Point Functions	`Send, Recv, Isend, Irecv`
Collective Functions	`Allgather, Reduce, Barrier`
Packing Strategy	Temporary buffers and MPI Datatypes
Requests	Extend the lifetime of Kokkos::Views

communication routines accept Kokkos::Views rather than pointers, datatypes, and counts. Current capabilities of the Kokkos Comm library are outlined in Table 2.1.

All experimentation was conducted on the Weaver testbed; some features of that cluster are presented in Table 3.1. All results reported reflect the average of data collected over ten runs.

3. OSU Latency Benchmarks. The Ohio State University (OSU) provides a comprehensive suite of microbenchmarks for testing a variety of host and device code. We adapted one of these tests, a ping-pong point-to-point program designed to capture MPI latency. In this microbenchmark, a message is sent back and forth while latency is measured [7]. Experimentation revealed only modest increases in latency when employing Kokkos Comm as compared to vanilla MPI, specifically the OpenMPI implementation. These results are depicted in Figure 3.1. This demonstrates the performance viability of Kokkos Comm despite the necessary additional overheads that come with wrapping MPI functions.

4. 3D Slicer Microbenchmark. Kokkos Comm aims to provide a set of tools for users to more easily manage communication of non-contiguous data, including efficient but error-prone non-blocking communication. One of the key features Kokkos Comm provides is the abstraction of packing and unpacking of data away from the user. Kokkos Comm may handle it automatically through one of two currently implemented packing methods: Deep Copy and MPI Datatypes. One method utilizes a contiguous scratch-buffer and leverages Kokkos `deep_copy` to efficiently marshal, pack, and transmit data. The alternative method takes the approach of leveraging MPI derived datatypes, as outlined in the MPI Standard [4].

Performance is a critical consideration. To measure and compare performance of these methods, a 3D Slicer microbenchmark was developed. The program generates a cubic Kokkos::View, and then communicates either one- or two-dimensional subviews, or 'slices', of the cube that are composed of non-contiguous data. Figure 4.1 depicts the results of these comparisons. Interestingly, communication of the two-dimensional subviews performed better than communication of one-dimensional subviews on smaller data sizes. Further research is required to confirm the root cause(s) of the optimization, current hypotheses revolve around parallelization and cache.

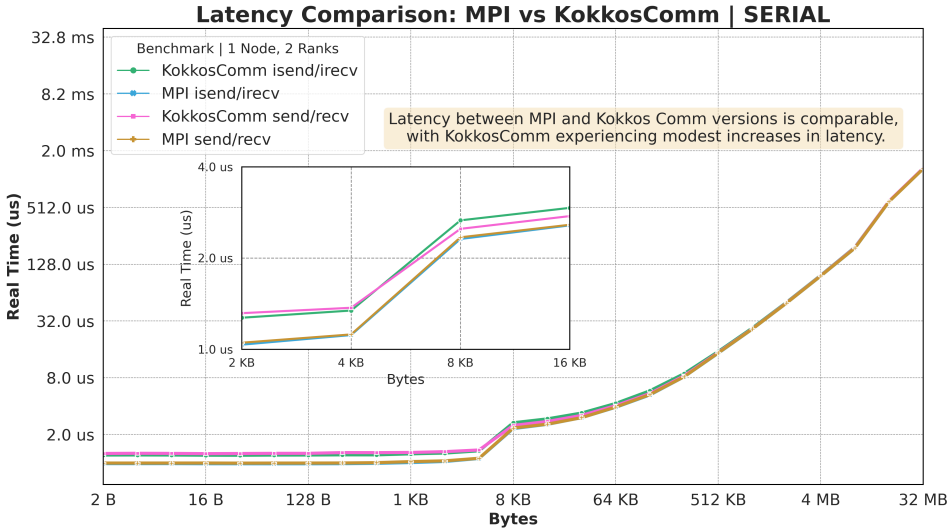


FIG. 3.1. *Latency Comparison*

TABLE 3.1
Weaver System Components

Feature	Description
CPU	Dual IBM Power0 (20 core)
GPU	Dual NVIDIA Tesla V100
Cores per Accelerator	5120 cores
Interconnect	Mellanox EDR Infiniband

5. Heat3D Halo Exchange Mini-Application.

$$\frac{\delta u}{\delta t} = \alpha\left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} + \frac{\delta^2 u}{\delta z^2}\right) \tag{5.1}$$

Term	Definition
u	temperature distribution function
t	time
α	thermal diffusivity of material
x, y, z	spatial coordinates

TABLE 5.1
Heat3D Terms

Heat3D refers to the three-dimensional heat equation (shown as Equation 5.1, terms laid out in Table 5), a partial differential equation. This problem provides the opportunity to measure performance of a common and useful operation for distributed computing, halo exchange. Execution of a three-dimensional heat equation also presents a deeper and more realistic picture of performance, as it is closer to a mini-application in scale than a microbenchmark.

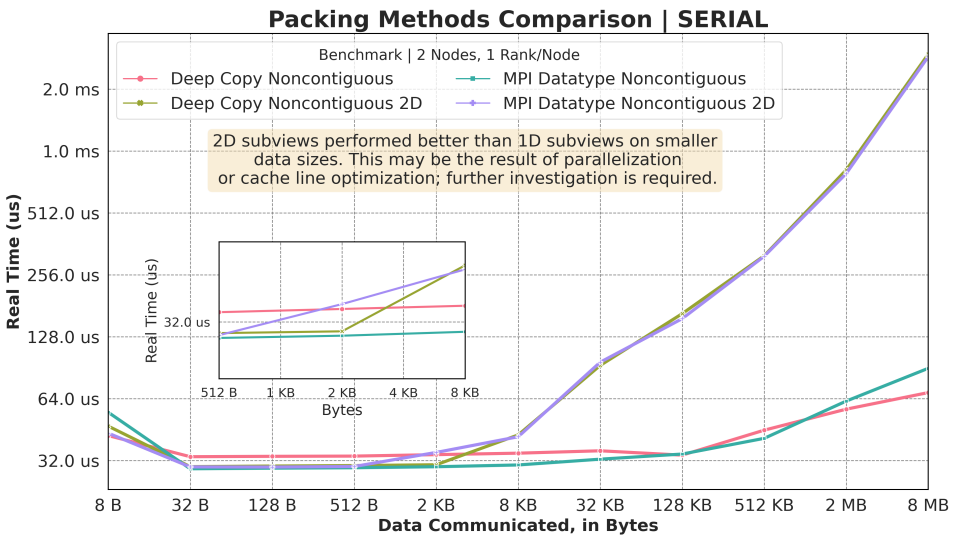


FIG. 4.1. *Packing Methods Comparison*

Figures 5.1 and 5.2 show two implementations of communication in the context of a 3D heat-diffusion stencil in C++, first utilizing vanilla MPI and then alternatively with Kokkos Comm. In Kokkos Comm, the buffer pointer, size, and datatype parameters are replaced with a single view. Furthermore, this example illustrates that, rather than an explicit pack step, a potentially non-contiguous subview representing the boundary data is provided directly to the Kokkos Comm function. Internally, Kokkos Comm can query the Kokkos::View and handle non-contiguous data appropriately. This is relevant as on GPU platforms, choosing an appropriate non-contiguous data strategy is crucial to performance.

Figure 5.3 compares the average real time of each implementation for a variety of configurations. Two configurations shown are with only the Serial backend of Kokkos enabled, once with two ranks on one node, and once with two nodes and one rank per node. Performance is comparable, with even a slight unexpected speed up when utilizing Kokkos Comm with the deep copy packing method.

The interesting results appear when the CUDA backend of Kokkos is enabled. There we see more significant overheads introduced when using Kokkos Comm generally, and a relatively drastic performance hit when using the derived MPI Datatype packing method specifically. The difference between the average real time using MPI and using Kokkos Comm with MPI Datatype packing is 33.9 milliseconds; MPI alone is over 10 times faster. This is likely the result of the overheads incurred when transferring data into GPU memory. These results are not unexpected, performance concerns surrounding derived MPI datatypes have persisted. Importantly, MPI performance and Kokkos Comm with Deep Copy packing performance are comparable while room for further optimization remains.

6. Future Opportunities and Challenges. New communication libraries afford the opportunity to improve correctness and ergonomics, but adoption will be hindered if the abstractions degrade performance. Microbenchmarks are included in the library's standard test suite to keep performance as a first-class concern.

Kokkos Tools [3] provided the framework to gather trace data on runs, which was

```
// Halo data
using buffer_t = Kokkos::View<double**, Kokkos::LayoutLeft, Kokkos::
    DefaultExecutionSpace>;
buffer_t T_left, T_left_out;
// ...
void setup_subdomain() {
    // incoming halos
    if (X_lo != 0)
        T_left = buffer_t("System::T_left", Y_hi - Y_lo, Z_hi - Z_lo);
    // ...
}

void pack_T_halo() {
    mpi_active_requests = 0; int mar = 0;
    if (X_lo != 0) {
        Kokkos::deep_copy(E_left, T_left_out, Kokkos::subview(T, 0, Kokkos::ALL,
            Kokkos::ALL));
        mar++;
    }
    // ...
}

template <class ViewType>
void isend_irecv(int partner, ViewType send_buffer, ViewType recv_buffer,
    MPI_Request* request_send, MPI_Request* request_recv) {
     MPI_Isend(send_buffer.data(), send_buffer.size(), MPI_DOUBLE, partner, 1, comm
         , request_send);
    // ...
}
```

FIG. 5.1. *Heat3D with MPI*

```
// Halo data
using lr_buffer_t = Kokkos::Subview<Dataview, int, decaltype(Kokkos::ALL),
    decaltype(Kokkos::ALL)>;
lr_buffer_t T_left, T_left_out;
// ...
void setup_subdomain() {
    // incoming halos
    if (X_lo != 0)
        T_left = Kokkos::subview(T, T.extent(0) -1, Kokkos::ALL, Kokkos::ALL);
    // ...
}

template <typename ExecSpace, class ViewType>
void isend_irecv(const ExecSpace &space,
    const ViewType &sv, ViewType &rv, int src, int dest, int tag, KokkosComm::Req &
        request_send, KokkosComm::Req &request_recv) {
    request_send = KokkosComm::isend(space, sv, dest, tag, comm)
    // ...
}
```

FIG. 5.2. *Heat3D with Kokkos Comm*

illuminating to problems other performance evaluation methods had left obscured. A trace of the Heat3D miniapplication is depicted in Figure 6.1, it shows a Heat3D application run three ways, once with vanilla MPI, once with Kokkos Comm using the deep copy packing method, and again with Kokkos Comm using the MPI derived datatypes packing method. Effort was expended to add custom profiling regions to Kokkos Comm to provide more granular information, and there is further room for optimization. Notably, the vanilla MPI implementation of Heat3D lacks a lot of the detail when compared to the Kokkos Comm implementations.

We will implement new MPI-style APIs as motivated for our application test cases. We expect to implement a similar interface for ISO C++ `mdspan`, a non-owning multidimen-

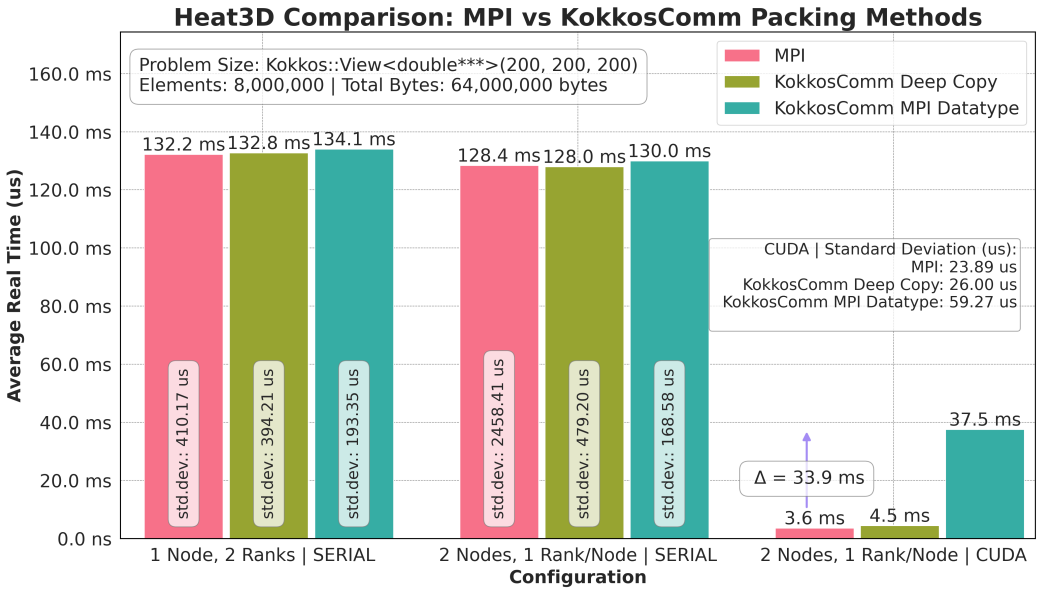


FIG. 5.3. *Heat 3D Comparison*

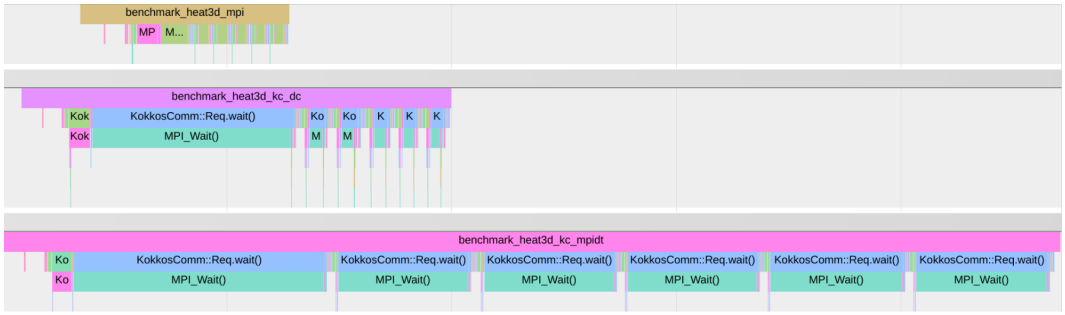


FIG. 6.1. *Heat3D Application, Traced with Kokkos Tools*

sional data structure.

A key objective of Kokkos Comm is to research communication APIs. We will use NCCL as our second transport layer to verify we can successfully abstract over non-MPI transports. We will also begin to investigate "stream-triggered" and device-initiated communication APIs, such as the HPE one-sided GPU triggering API [5], Intel's GPU-initiated communication feature [2], and MPICH's GPU triggering support [13].

Acknowledgment. This project wishes to acknowledge the fruitful collaboration with our colleagues at Université Paris-Saclay: Gabriel Dos Santos, Cédric Chevalier, Hugo Taboada, and Marc Pérache and their paper, KokkosComm: Communication Layer for Distributed Kokkos Applications.

REFERENCES

[1] A. BIENZ, D. SCHAFER, AND A. SKJELLUM, *MPI Advance : Open-Source Message Passing Optimizations*, 2023.

[2] Intel Corp., *Intel MPI Library Developer Reference for Linux: GPU Buffers Support*, June 2024.
[3] S. N. Laboratories, *Kokkos tools*, 2024. Accessed: 2024-08-29.
[4] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard Version 4.0*, June 2021.
[5] N. Namashivayam, K. Kandalla, J. B. W. I. au2, L. Kaplan, and M. Pagel, *Exploring Fully Offloaded GPU Stream-Aware Message Passing*, 2023.
[6] NVIDIA Corporation, *NVIDIA Collective Communications Library (NCCL)*, 2024. Version 2.22.3.
[7] Ohio State University, *MVAPICH::Benchmarks*. `https://mvapich.cse.ohio-state.edu/benchmarks/`.
[8] OpenFabrics Interfaces Working Group, *libfabric: High-Performance Fabric Interface*, 2024. Version 1.22.0.
[9] A. Skjellum, M. Rüfenacht, N. Sultana, D. Schafer, I. Laguna, and K. Mohror, *Exampi: A modern design and implementation to accelerate message passing interface innovation*, in High Performance Computing, J. L. Crespo-Mariño and E. Meneses-Rojas, eds., Cham, 2020, Springer International Publishing, pp. 153–169.
[10] E. Suggs, S. Olivier, J. Ciesko, and A. Skjellum, *View-aware Message Passing Through the Integration of Kokkos and ExaMPI*, in Proceedings of the 30th European MPI Users' Group Meeting, 2023, pp. 1–10.
[11] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, and J. Wilke, *Kokkos 3: Programming Model Extensions for the Exascale Era*, IEEE Transactions on Parallel and Distributed Systems, 33 (2022), pp. 805–817.
[12] UCX Consortium, *Unified Communication X (UCX)*, 2024. Version 1.17.0.
[13] H. Zhou, K. Raffenetti, Y. Guo, and R. Thakur, *MPIX Stream: An Explicit Solution to Hybrid MPI+X Programming*, in Proceedings of the 29th European MPI Users' Group Meeting, EuroMPI/USA '22, New York, NY, USA, 2022, Association for Computing Machinery, p. 1–10.

ANALYSIS OF MODERN TOOLS FOR COMMUNICATION IMPACTS

NICHOLAS BACON *, SCOTT LEVY†, PATRICK BRIDGES‡, AND KURT B. FERREIRA§

Abstract.
As applications scale to larger and larger processor counts, network communication performance can become a potential bottleneck. In this paper, we examine the ability of modern performance tools and analyses to measure the impact of communication on strong scaling of high-performance computing benchmarks and applications relevant to Sandia National Laboratories focus benchmarks and applications. Specifically, we evaluate the ability of Caliper communication instrumentation capabilities and Hatchet performance analyses to measure communication scaling costs in three mini-apps: MiniFE, MiniAero, and AMG2023. Our results show that existing tools can be used to profile communication performance and predict how changes in communication performance would impact application scaling. We also identify directions for future work, including quantifying the accuracy of these profiling-based predictions and creating additional communication performance analyses.

.

1. Introduction. As high-performance computing (HPC) systems and their workloads grow in complexity and size, network communication performance can become a potential bottleneck. For example, when strong scaling a workload (increasing the number of processors for a fixed amount of work), the serial costs of communication limit the potential speed up as described by Amdahl's Law [1]. Because of this, accurately measuring and predicting communication performance in HPC workloads is important for, for example, identifying communication operations that are worth optimizing.

In this paper, we describe and evaluate an approach to using state-of-the-art sample-based profiling tools and associated analyses to measure and give insight into the growth of communication overheads when strong scaling workloads. Specifically, we use Caliper [3] to sample and profile communication overheads and Hatchet [2] to analyze the resulting profiles. This profiling-based approach is in contrast to the trace and simulation-based techniques, e.g., LogGOPSim [6].The workloads chosen for this research are representative of scientific computations on HPC systems, including those at Sandia in particular – MiniFE [9], MiniAero [4], and AMG2023 [8] have been choose in part their ability to scale and/or their common style in other HPC workloads.

In this paper, we make the following contributions:

- A Python-based analysis of communication performance costs using Caliper-collected data;
- An initial quantitative evaluation of this approach on three workloads relevant to scientific computations on HPC systems; and
- An Identification of next steps to do qualitative analyses to evaluation the capabilities of sample-based profiling and analysis tools to predict communication performance impacts.

The remainder of this paper is organized as follows. Section 2 begins with a discussion of the state-of-the-art techniques in network analysis. Section 3 then follows with a discussion of the challenges of using sampling tools and the quality of there outputs. Section 4 describes the experimental setup used to evaluate the measurement approach on modern systems, and Section 5 presents and discusses the results of these experiments. Finally, Section 6 summarizes paper results and discusses directions for future work.

* University of New Mexico, nbacon@unm.edu
†Scalable System Software Department, sllevy@sandia.gov
‡University of New Mexico, patrickb@unm.edu
§Scalable System Software Department, kbferre@sandia.gov

2. Background. Our proposed approach uses three related tools, the Kokkos tools framework [10], Caliper [3], and Hatchet [2], profile and analyze a workload's communication performance. This section to describes these tools and discusses their application to HPC workloads.

We use the Kokkos Tools extension library and the Caliper performance monitoring tool to gather performance information from HPC workloads in the work described in this paper. Kokkos Tools is a performance monitoring extension for workloads and libraries written using the Kokkos [11] framework. It provides programmers a consistent library for integrating Kokkos workloads with a range of performance monitoring and debugging tools, including exporting information on Kokkos parallel regions and functions to HPC performance monitoring tools to make those tools easier to use and their results understandable.

Caliper is one such performance monitoring tool, and we chose to use it for performance monitoring because it supports a large range of data-collection approaches such as interrupt-based sampling and tracing of communication calls. In addition, Caliper and Kokkos allow programmers to manually annotate code regions of interest, and Caliper provides additional mechanisms for extracting and merging performance data from workloads such as call-stack unwinding and merging performance data across multiple processes. Finally, Caliper includes multiple output formats that are human(tree) and computer readable(spot, JSON).

Hatchet complements Caliper by providing a Python-based library for analyzing Caliper performance data. Specifically, Hatchet allows Caliper performance data to be imported into Python `pandas` *DataFrames*. These DataFrames are associated with a secondary GraphFrame, allowing indexing by structured tree and graph data. GraphFrame makes it easier to work with data with both structured and complex relationships. In addition, Hatchet provides functions for analyzing Caliper performance data and visualizations such as tree views, flame graphs, and directed acyclic graph views to aid in understanding collected performance data.

3. Approach. Our overall goal was to make a quantitative analysis of Caliper's ability to capture communication costs and impacts on workload performance. We chose Python, Caliper, and Hatchet for this purpose because they a focus on ease of use, with Caliper providing flexible performance monitoring and the combination of Python and Hatchet providing a rich ecosystem of data analyses and visualisations through `pandas` and `seaborn`. In the remainder of this section, we describe: (1) the high-level sequence of steps of our approach to understanding communication impact on performance, (2) the communication analyses we created to prototype this approach, and (3) additional features that would be useful to add to Hatchet to make designing similar analyses easier.

3.1. Overall Workflow. The general workflow we propose for analyzing communication costs in HPC workloads is:

1. Annotate the source code with Caliper or Kokkos code regions to group or name specific functionality, therefore making it easier to profile or analyze performance of these regions.
2. Use the command in Listing 1 to collect Caliper with 'spot' data for each run of the HPC workloads. In this study, we strong scale each workload by fixing the problem size and doubling the compute resources available until reaching 16 nodes on Sandia Nations Labs Glinda cluster. The result in five `.cali` files for each program.

```
export CALI_CONFIG=spot(output=$1.cali),time.exclusive,profile.
    mpi,mem.highwatermark,cuda.gputime,profile.kokkos
```

Listing 1: Use Caliper with Spot on the command line

3. Use Hatchet to import each programs `.cali` file collected by Caliper into Python `pandas` DataFrames.
4. Merge the `pandas` DataFrames into a single DataFrame,
5. Separate the `pandas` data into two distinct DataFrames: one for exclusive times and the other for inclusive times with key being `row['name']` in both.
6. Within the exclusive times DataFrame, utilize built-in `pandas` methods to consolidate similar functions to a row name (as shown in Listing 2) to one common name. Finally aggregated rows these rows with with common names adding execution times (Listing 3). These methods are used to combine `'like'` work (example the mutligrid function in AMG2023) and the MPI libary calls. This process only applies to exclusive times as Caliper's inclusive times represent the cumulative execution time of all helper' function calls within a function plus the work done in the function itself. Exclusive times measure, in contrast, the time spent only within the function itself, excluding any time consumed by helper functions.

```
def filter_function_name(row):
    if "mpi" in row['name'].lower():
        #Group all row that have the names that contain 'mpi'
        return "MPI"
    elif "kokkos" in row['name'].lower():
        #Group all row that have the names that contain 'kokkos'
        return "kokkos"
    #no name change
    return row['name']

...

 dataframe.apply(filter_function_name,axis=1)
```

Listing 2: Use Caliper with Spot on the command line

```
 dataframe.groupby('name').agg([np.sum]).reset_index()
```

Listing 3: Use Caliper with Spot on the command line

7. Perform the analysis on data with and without the cost of using the network.
8. Make data visualizations

3.2. Developed Analyses and Visualizations. To support this overall approach, we enhanced existing Hatchet analyses system previously described in Section 2 to better target communication performance. Specifically, we created an analyses that focuses on the "what if" speed-up graph that shows how removing communications cost would directly increase the speed of the program. This analysis computes speedup $\frac{f(1)}{f(n)}$ on the normal run times $f(n)$ reported by Caliper and the compared to the computed times by removing all networking costs from the speed up using formal $\frac{f(1)}{f(n)-\text{networking time}}$. This analysis was used done in conjunction with four visualizations described below that help identify performance bottlenecks related to communication overheads.

The visualizations include:

1. Stratigraphy View: Stacks functions to provide a layered representation of performance per function over time.

2. Stacked Time View: Displays the distribution of time spent in various functions, offering a clear comparison of execution times.
3. Speedup View: shows the relative speedup achieved across different runs. 5.1b,5.3b,5.2b
4. min,max, average View: Stacks functions with a min-max-average to easily spot irregularities and bottlenecks.

These visualizations allowed clearer understanding of the changing cost different functions especially the networking like MPI functions.

3.3. Potential Approach Extensions. As part of creating these analyses, we also identified functionality missing in Hatchet for working with Caliper-generated data that would have made constructing new analyses easier.

First analyses frequently needed to group together related functions from different parts of the profile tree, and we ended up coding this grouping by hand. Hatchet and Caliper functionality that made it easier to tag and merge such performance data would make constructing analyses such as the ones we designed easier. Similarly, the ability to filter data based on execution times, call structure, and function names would enhance the flexibility and efficiency of these analysis. The filtering performed to create custom speedup for AMG2023 [5] based off the Figure Of Merit (FOM) if there was no network cost was done mixing exclusive and inclusive times, and was awkward. The FOM for AMG2023 is simple by $\frac{\text{nnz\_AP}}{\text{Setup + Solve}}$ calulation in units of inverse seconds, requiring using the Caliper region name *Setup* and *Solve* in inclusive times and subtracting MPI exclusive times for them. These FOMs could have been calculated by using a function that gave all rows that were sub-child and post-parent instead, easing its implementing and writing similar analyses .

4. Experimental Setup. We evaluated the challenges and approach described in the previous section by studying if the results of the analysis described in the previous corresponded qualitatively with the expected impact of removing communication overheads from three different HPC workloads; a quantitative comparison of this analysis with a validated simulation is planned for future work.

To do so, for each workload we:

1. Determined the largest size problem that would run on one compute node.
2. Ran the program on this problem with Caliper profiling enabled from 1 to the largest number of nodes that the program would run on the test system
3. Performed the communication scalability impact analysis using the process described in Section 3
4. Visualized the resulting data in a speedup graph

We ran our tests on the Sandia National Laboratories Glinda cluster, an HPC cluster composed of compute nodes built around AMD EPYC 2.80 GHz CPUs, NVIDIA A100 GPU, and NVIDIA Mellanox ConnectX-6 2xHDR InfiniBand network interface cards.`gdrcopy` support is not provided on these nodes. The list of modules and libraries we used on this system is provided in Appendix A. The workloads we chose to study are representative of common scientific computations on HPC systems and are relevant to the lab mission and have relevant communication patterns. Specifically, we measured the performance of MiniFE [9], MiniAero [4], and AMG2023 [8] benchmarks, described in more detail below.

AMG2023 [8] is a mini-application developed by Lawrence Livermore National Laboratory (LLNL) and successor of the AMG2013 [5] benchmark. It uses a parallel algebraic multi-grid solver built on the hypre library [7]. AMG2023 solves two different diffusion problems on with that differ by the preconditioner problem one uses preconditioned conjugate gradients (PCG) while problem 2 uses generalized minimum residual method (GMRES). For our study, we chose problem 1 because PCG has a lager amount of network traffic when compared to the GMRES in problem 2.

This is the default problem when running AMG2023, it is 3D diffusion problem on a cuboid with a 27-point stencil.
For this experment we used "19bc10c925c4434da72a9cbb4fa1a009dbc52f33" variant of AMG2023. The configuration is different from the other workloads in this study, because AMG2023 is a weak scaling miniapp. If problem is kept the same, the input the problem size will be spreed out over the available nodes. To keep the problem size the same we need to use the system of equations:

$$\begin{cases} p = \#\ of\ process \\ a * b * c = p \\ max(a,b,c)/min(a,b,c) \leq 2 \\ a \leq b \\ b \leq c \\ a \leq c \end{cases} \tag{4.1}$$

in the following:

```
amg -P a b c -n 240/a 240/b 240/c -problem 1
```

Listing 4: AMG2023 input

The 240 numerator is largest problem size that fits on one node of Glinda without causing a segmentation fault while also being dividable by 16 our maximum run size on Glenda.

MiniAero is a mini-app from Mantevo project. This code is written in kokkos and uses MPI for commutation. The problem that this code is try to solve is the compressible Navier-Stokes equation by using a explicit unstructured finite volume solve.
For this paper we used the code of "f46d135479a5be19ec5d146ccaf0e581aeff4596". The problem for miniAero is a modification of the "FlatPlate Parallez" input deck to fit onto a single A100.

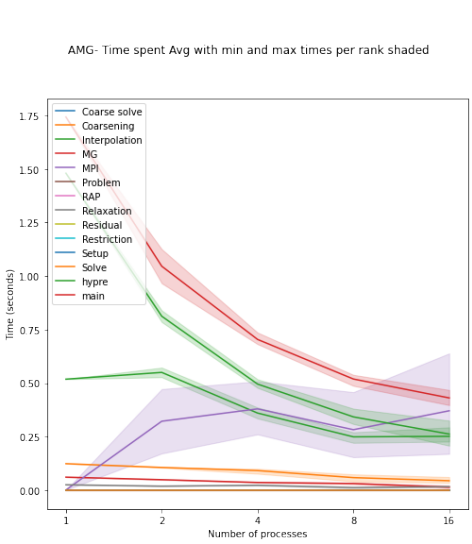
```
1
2.0 0.002 1.0 0.0
512 512 32
1600
1e-8
1
100
1
1
```

Listing 5: miniaero.inp file use as input in MiniAero

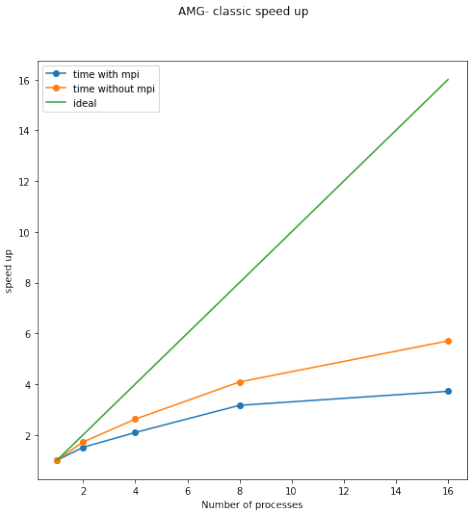
MiniFE is another mini-app out of the Mantevo project developed by Sandia National Laboratories (SNL). MiniFe is a proxy application for unstructured implicit finite element codes. For this project we are using a fork that has be updated to kokkos/4.3 hash b520fae9b1cff6d965d574e87816d85d6ab1f159. We have chosen the size to be a 400 by 400 by 400 cubic space.

```
miniFE.kokkos -nx 400 -ny 400 -nz 400
```

Listing 6: Command line to run miniFE on the kokkos varrent



(a) In this graph the Mpi time seem to be grow and dominating causing poor speed up.



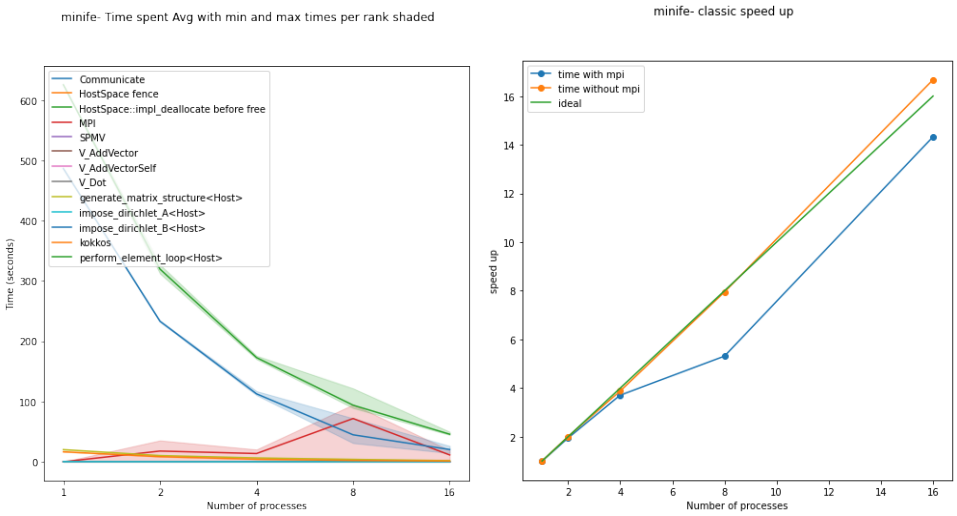
(b) In this graph results demonstrates a poor ability to strong scale but seem hopeful with networking removed.

Fig. 5.1: AMG2023: These graphs show the performance of AMG2023 over multiple runs.

5. Results. Figures 5.1, 5.2, and 5.3 present (1) the sampled Caliper breakdown of exclusive times spent in different portions of the relevant benchmarks as the number of processes is increased, (2) the measured strong scaling speedup of the program with and, (3) the projected strong scaling if MPI communication were removed. In the breakdown graph, the solid lines represent the average exclusive time for that function across all processes while the shaded area shows the minimum and maximum times spent in that function by some process. In the speedup graph, measured speedup and projected speedup are shown along with the a reference ideal speedup line.

5.1. Analysis of AMG2023. The results in Figure 5.1a, the AMG2023 time breakdowns, have lines with performance trends in three distinct groups. The first group contains the top 3 lines, MG (red), Interpolation (green) and hypre (green) with signifcant computation at small process counts that decease significantly as the number of processes increases. The second group contains multiple lines near the X axis with little computation time and a constant or slightly decreasing amount of time as the number of processes grows. The last group, the purple line and lavender shaded region, is the time spent in MPI networking calls and grows signicantly and with high variance.

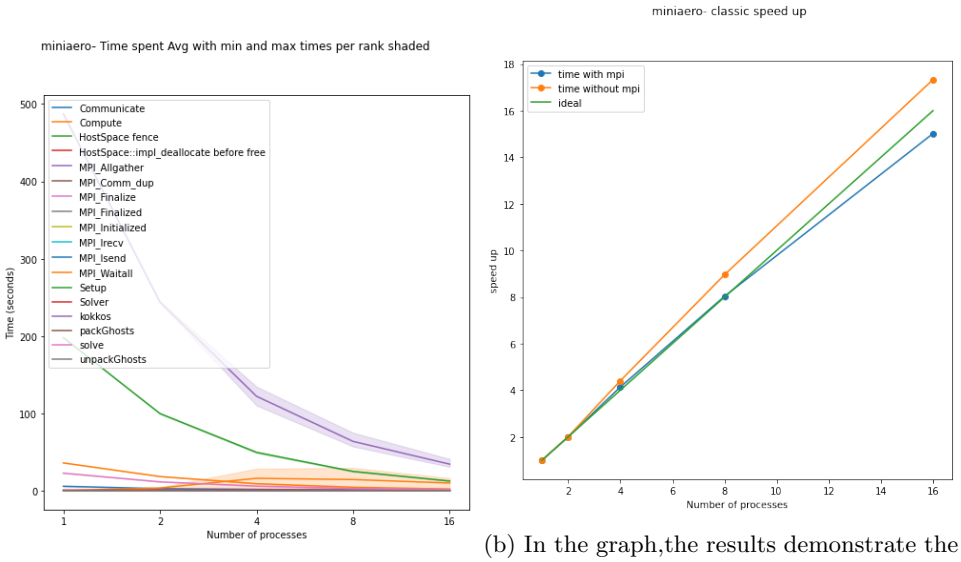
These results correspond with the results of the analysis shown in Figure 5.1b that compute the speedup if AMG2023, which is known to be communication sensitive, had infinitely fast communication. Specifically, this graph computes a substantial speedup from removing communicaiton time of over 60%. The results also shows that AMG2023 would not scaling ideally even with communication time removed because other non-communication methods are not scaling perfectly. For instance in Figure 5.1a the MG function (red) takes the longest and the highest proportion of times on a single node, but time in this function does not reduce by half when doubling the node count.



(a) In the graph, MPI time grows slightly, causing the application to not reach ideal speed up.

(b) In the graph,the results demonstrate a odd phenomenon were removing all networking cost gave better then ideal.

Fig. 5.2: These graphs show the performance of MiniFE over multiple runs.



(a) In this graph we can see the idea strong scaling situation were all function times are decreasing in time.

(b) In the graph,the results demonstrate the sane odd phenomenon as the miniFE were removing all networking cost gave better then ideal. This result of non-linear scaling is odd.

Fig. 5.3: These graphs show the performance of miniaero over multiple runs.

5.2. Analysis of Miniaero and MiniFE. Figures 5.3 and 5.2, show the time breakdown and speedup of MiniAero and MiniFE, both of which are compute-bound and demonstrate good strong scaling at larger problem sizes. This is clear in the breakdowns in Figures 5.3a and 5.2a which show the time in two compute methods rapidly dropping by half on when doubling the number of nodes. Both codes spend minimal time in MPI; the exception is an increase in the time spent in MPI in MiniFE when running on 8 nodes; this change could be due to variance in experimental performance, as this variance in performance is lower when running on 16 processes.

The communication-free analysis, shown in Figures 5.3b and 5.2b, correctly corresponds to these results. In particular, when MPI times are removed we get only slightly better performance than with MPI. However, these figures also predict slightly super-linear speedup; we have not yet diagnosed the cause or correctness of this prediction.

5.3. Results Summary. These results provide an initial demonstration of the usefulness of Caliper and Hatchet for measuring communication performance in HPC applications and their ability to support novel analyses on how communication cost changes would impact application performance. The ease of data collection and the creation of new analyses, in particular, are worth noting.

6. Conclusions and Future Work. The data from the scaling study highlight the influence of network performance on application performance, and the ability to profile these costs with Caliper and analyze and visualize them with Hatchet. The analysis of benchmark data from AMG2023 on Glinda, for example, shows that there is significant overhead when it comes to the communication, about 60%, which is consistent with what was expected when selecting this benchmark. In contrast, MiniAero and MiniFE performance improved by now more than 15% when removing communication time, again as expected.

We have identified multiple directions for future work as part of this work. First, extending this work to larger-scale runs, more workloads that are communication sensitive, and more workloads in general is a key direction for future work. In addition, we also want to perform quantitative students that compare the accuracy and expense of the predictions from this sampling-based approach to validated simulation-based approaches; to do so, we plan to compare the performance and accuracy of the analyses from the approach described in this paper with those predicted by the LogGOPSim tool [6].

Appendix A. System modules used

system modules	version
cmake	2.27.7
cuda	12.0.0
GCC	12.3.0
OPENMPI	4.1.6
Python	3.11.6

Appendix B. Github software with hashes

library	version	Hash
Kokkos-tools	2.5.00	58258bd6666dc6d11cb748441f62e68f9d22ba27
Kokkos	4.3.00	486cc745cb9a287f3915061455105a3ee588c616
hypre	2.31.0	7e7fc8ce09153c60ae538a52a5f870f93b9608ca
Caliper	v2.11.0	e5934eab072bcc2e5784a62783b16fc37e3d0cd1
Hatchet	v2024.1.2	a91e194f06ffe4c2b6604704541f46862844ab58

[1] G. Amdahl, *Validity of the single processor approach to achieving large scale computing capabilities, reprinted from the afips conference proceedings, vol. 30 (atlantic city, n.j., apr. 18–20)*, Solid-State Circuits Newsletter, IEEE, 12 (2007), pp. 19 – 20.

[2] A. Bhatele, S. Brink, and T. Gamblin, *Hatchet: pruning the overgrowth in parallel profiles*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '19, New York, NY, USA, 2019, Association for Computing Machinery.

[3] D. Boehme, T. Gamblin, D. Beckingsale, P.-T. Bremer, A. Gimenez, M. LeGendre, O. Pearce, and M. Schulz, *Caliper: performance introspection for hpc software stacks*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '16, IEEE Press, 2016.

[4] T. C. Fisher, T. C. Fisher, S. W. Bova, S. W. Bova, P. Lin, P. Lin, K. J. Franko, and K. J. Franko, *Cfd for next generation hardware: Experiences with proxy applications.*, (2015).

[5] V. Henson and U. Yang, *Boomeramg: A parallel algebraic multigrid solver and preconditioner*, Applied Numerical Mathematics, 41 (2002), pp. 155–177.

[6] T. Hoefler, T. Schneider, and A. Lumsdaine, *LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model*, in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ACM, Jun. 2010, pp. 597–604.

[7] *hypre: High performance preconditioners.* `https://llnl.gov/casc/hypre`, `https://github.com/hypre-space/hypre`.

[8] R. Li, U. M. Yang, and U. N. N. S. Administration, *Amg2023, version 1.0.0*, 3 2023.

[9] P. Lin, M. Heroux, R. Barrett, and A. Williams, *Assessing a mini-application as a performance proxy for a finite element method engineering application*, Concurrency and Computation: Practice and Experience, (2015).

[10] C. Trott, L. Berger-Vergiat, D. Poliakoff, S. Rajamanickam, D. Lebrun-Grandie, J. Madsen, N. Al Awar, M. Gligoric, G. Shipman, and G. Womeldorff, *The kokkos ecosystem: Comprehensive performance portability for high performance computing*, Computing in Science Engineering, 23 (2021), pp. 10–18.

[11] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, and J. Wilke, *Kokkos 3: Programming model extensions for the exascale era*, IEEE Transactions on Parallel and Distributed Systems, 33 (2022), pp. 805–817.

SUM OF SQUARES BOUNDS ON THE PERFORMANCE OF THE QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

AIDAN EPPERLY*, KEVIN THOMPSON†, AND OJAS PAREKH‡

Abstract.
The quantum approximate optimization algorithm (QAOA) [3] is a commonly cited variational quantum algorithm for solving combinatorial optimization problems. While rigorous upper bounds on the performance exist for single layer QAOA [6], [7], such results are not easily extended to higher numbers of layers due to the potentially exponential growth of the light cone size. We propose a variety of methods based on sum of squares optimization to provably upper bound the performance of QAOA when applied to the max-cut problem. We show that for some classes of graphs, the approximation ratio of two layer QAOA is worse than the best known classical approximation ratio for max-cut. These preliminary results lend support to usefulness of sum of squares optimization in bounding QAOA performance.

1. Introduction. The quantum approximate optimization algorithm (QAOA) [3] is a prototypical and widely cited example of a variational quantum algorithm (VQA). VQAs take advantage of classical techniques to optimize parameterized quantum circuits. Many classical optimization algorithms treat the evaluation of the objective function as a black-box operation. In a VQA, these black-box operations are performed using a quantum circuit that depends on the optimization parameters. Since the evaluation of the objective function is often the most time and space intensive aspect of the optimization, this is a natural division of labor.

QAOA, in particular, was designed to solve combinatorial optimization problems. Following in the footsteps of the original QAOA paper [3], we will focus on one such problem: graph max-cut. Given a graph G on n vertices, a *cut* S of G is a subset of the vertices of G. The associated *cut-value* $\mathrm{Cut}(S)$ is given by

$$\mathrm{Cut}(S) = \left|\{ij \in E(G) \;\middle|\; i \in S,\, j \in S^c\}\right|. \tag{1.1}$$

As the name suggests, the objective of max-cut is to find a cut $S^* \subseteq V(G)$ maximizing $\mathrm{Cut}(S^*)$. There is a well-known method to turn max-cut into a quadratically constrained quadratic program. To do this, we first assign to each vertex i of G a weight x_i. Given a cut S, we will let $x_i = 1$ if $i \in S$ and $x_i = -1$ otherwise. This yields the formula for the cut value

$$\mathrm{Cut}(S) = \sum_{ij \in E(G)} \frac{1 - x_i x_j}{2}, \tag{1.2}$$

and the reformulation of the max-cut problem as

$$\max_{x_i} \sum_{ij \in E(G)} \frac{1 - x_i x_j}{2}, \text{ s.t. } x_i^2 = 1. \tag{1.3}$$

This quadratic program can be turned into a local Hamiltonian problem by replacing x_i with Z_i, the Pauli Z operator on the ith qubit, in the formula. This yields the local Hamiltonian problem

$$\max_s \langle s| H |s\rangle, \; H := \sum_{ij \in E(G)} \frac{1 - Z_i Z_j}{2}. \tag{1.4}$$

*Sandia National Laboratory, acepper@sandia.gov
†Sandia National Laboratory, kevthom@sandia.gov
‡Sandia National Laboratory, odparek@sandia.gov

A priori, we might expect this to be a relaxation of max-cut as $\langle s| Z |s\rangle$ can attain any value in the interval $[-1, 1]$. However, as the Hamiltonian is diagonal in the Z basis, it is clear that the maximizer s^* will be such that $\langle s^*| Z_i |s^*\rangle \in \{-1, 1\}$, or, equivalently, $\left(\langle s^*| Z_i |s^*\rangle\right)^2 = 1$. We will proceed with this final formulation of the max-cut problem.

1.1. QAOA for Max-Cut. QAOA can be described by the application of successive "layers". A single QAOA layer, U_ℓ, consists of two parameterized unitary operations

$$U_H(\theta_\ell) := \exp(i\theta_\ell H), \tag{1.5}$$

$$U_X(\gamma_\ell) := \exp(i\gamma_\ell \bar{X}), \tag{1.6}$$

$$\bar{X} := \sum_{i \in V(G)} X_i, \tag{1.7}$$

so that $U_\ell(\theta_\ell, \gamma_\ell) = U_X(\gamma_\ell)U_H(\theta_\ell)$. The full p-layer QAOA circuit is then given by the unitary

$$U(\theta, \gamma) := U_p(\theta_p, \gamma_p) \cdots U_2(\theta_2, \gamma_2)U_1(\theta_1, \gamma_1). \tag{1.8}$$

We will use $U_{lc}(\theta, \gamma)$ to denote the unitary arising from the light cone. From this, we can express p-layer QAOA as the optimization problem

$$\max_{\substack{\gamma=(\gamma_1,\ldots,\gamma_p)\\ \theta=(\theta_1,\ldots,\theta_p)}} \langle t| U(\theta, \gamma)^\dagger H U(\theta, \gamma) |t\rangle, \ |t\rangle := \left|+^{\otimes n}\right\rangle. \tag{1.9}$$

The convergence of QAOA follows from the quantum adiabatic theorem and the Trotter product formula [3]. In particular, the p-term Trotterization of adiabatic evolution is always a valid QAOA circuit. While the Trotter product formula guarantees convergence in the limit, it does not tell us that the p-term Trotterization is in any sense optimal. The goal of QAOA is in some sense to find what the optimal p-term unitary with roughly the same structure as the Trotterization would be, where optimality to judged by the objective function.

2. Bounding QAOA. Despite the popularity of VQAs and QAOA in particular, bounds of the performance of these algorithms, particularly for small, fixed circuit depth, are rare [2]. To our knowledge, there is no known nontrivial bound on the performance of QAOA for $p > 1$ layers. As such, it is unknown how many layers one should expect to have to use to approximate max-cut on a given graph G to a higher degree of accuracy than classical methods. We aim to find such an upper bound on the performance of p-layer QAOA.

Let $\mathrm{QAOA}_p(G)$ denote the optimal value of Equation (1.9). While the objective function in Equation (1.9) has been computed exactly for $p = 1$ layers in the form of trigonometric a trigonometric polynomial as in [7], [6], similar expression have not been computed for higher p values. This makes it difficult to establish a priori estimates of QAOA performance. By establishing an upper bound on $\mathrm{QAOA}_p(G)$ for a given value of p, we can then lower-bound the number of layers needed for noiseless QAOA to perform above a certain threshold. We set our desired threshold at roughly 0.878, by which we mean that we want to find the smallest value p such that the ratio between $\mathrm{QAOA}_p(G)$ and the true max-cut value $\mathrm{MaxCut}(G)$ is greater than roughly 0.878. We chose this specific constant as it is the expected approximation ratio of the best-known classical, polynomial-time approximation algorithm for max-cut, the Goemans-Williamson randomized rounding algorithm [4].

2.1. Sum of Squares and the Method of Moments. Much like Goemans and Williamson, we turn to sum of squares (SoS) relaxation as our method of choice for bounding $\mathrm{QAOA}_p(G)$. In a great and lasting irony, when working with sum of squares, we almost always work with the dual problem of moments instead. We follow this course here as well. In particular, we will use a modification of a method first put forward by Lasserre [5] in order to obtain a hierarchy of semidefinite programs each of which provides a tighter bound on $\mathrm{QAOA}_p(G)$.

The general idea is as follows. Suppose $|s\rangle = U(\theta,\gamma)\,|t\rangle$ is some state resulting from QAOA. By virtue of its construction, the state $|s\rangle$ will have some non-trivial algebraic constraints on its entries. The easiest such constraint is that $\langle s \mid s\rangle = 1$, but there are many others. These algebraic constraints actually uniquely define the set of all possible QAOA states, which is the feasible set for our optimization problem. Unfortunately, there are so many of these relations that it would be intractable to enforce them all. Even if we took the convex hull of our feasible set, it would still be computationally difficult to optimize over. As such, we seek a tractable, convex relaxation of the feasible set. The method of moments provides us with an infinite family of such relaxations by deriving and enforcing smaller sets of nontrivial polynomial relations that can be efficiently enforced via a semidefinite constraint. Since we are relaxing the feasible set, the resulting value of the objective function will be an upper bound on the true optimal value.

Note that the Z_i's form a commutative algebra and satisfy the additional condition $Z_i^2 = \mathbb{1}$. This latter condition requires additional consideration. For instance, given a Pauli monomial q, how do we determine the degree of q? With generic commutative monomials, the degree is the same as the *string length* of the monomial. So, for instance, the degree of $xyzx$ is exactly equal to 4. However, the monomial $q = Z_1Z_2Z_1 = Z_1^2Z_2 = Z_2$ can be written with a string length of 1 or 3. Going too far further in this discussion would require a lengthy description of Gröbner bases and minimal reduced words, but the following lemma sums up the results that we need.

LEMMA 2.1. *Every monic Pauli Z monomial, q, can be written uniquely in the form*

$$q = Z_{i_1}Z_{i_2}\cdots Z_{i_k}, \tag{2.1}$$

$$i_1 < i_2 < \cdots < i_k. \tag{2.2}$$

Moreover, if $q = \prod_{i\in S} Z_i$, then $|S| \geq k$. In particular, the definition $\deg(q) := k$ is well defined.

Henceforth, all monomials will be assumed to be in this standard form. With the issue of degree settled, let $\mathcal{B}_d$ be the ordered list of Pauli Z monomials of degree at most d. We order $\mathcal{B}_d$ first by degree and then, within each degree, lexicographically. So, for instance, $Z_5 < Z_2Z_3 < Z_2Z_4 < Z_3Z_4 < Z_1Z_2Z_3$. This is often called graded lexicographic order. Let $\mathcal{M}_{2d} := \mathcal{B}_d\mathcal{B}_d^\dagger$, by which we mean that $\mathcal{M}_{2d}$ is a matrix indexed by $\mathcal{B}_d$ such that $\mathcal{M}_{2d}(q_1,q_2) = q_1q_2$. So, for example, if $n = 2$, then

$$\mathcal{M}_2 = \begin{pmatrix} \mathbb{1} & Z_1 & Z_2 \\ Z_1 & \mathbb{1} & Z_1Z_2 \\ Z_2 & Z_1Z_2 & \mathbb{1} \end{pmatrix}. \tag{2.3}$$

By an abuse of notation, given a state $|s\rangle$, we denote by $\langle s|\,\mathcal{M}_{2d}\,|s\rangle$ the matrix

$$\langle s|\,\mathcal{M}_{2d}\,|s\rangle\,(q_1,q_2) = \langle s|\,q_1q_2\,|s\rangle\,. \tag{2.4}$$

This is the so-called Gram matrix of the set of vectors $q\,|s\rangle$ for $q \in \mathcal{B}_d$ and it is thus positive semidefinite. We aim to compute a tight outer approximation on $\overline{\mathrm{conv}}(\{\langle s|\,\mathcal{M}_{2d}\,|s\rangle \mid |s\rangle = U(\theta,\gamma)\,|t\rangle\})$.

2.2. Naïve Approach. For the duration of this section, by $|s\rangle$, we mean a possible state $|s\rangle = U(\theta,\gamma)|t\rangle$ resulting from the QAOA circuit. We start with the innocuous observation that any convex combination of matrices of the form $\langle s|\mathcal{M}_{2d}|s\rangle$ can be described elementwise as a convex combination of matrix entries. Thus, we restrict ourselves to studying the values $\langle s|q|s\rangle$ for q some Pauli Z monomial. The simplest possible observation we can make is that $\langle s|\mathbb{1}|s\rangle$ will always take the value 1. Moreover, the diagonal of $\mathcal{M}_{2d}$ is identically $\mathbb{1}$ which implies that the diagonal of $\langle s|\mathcal{M}_{2d}|s\rangle$ is identically 1. Now, suppose that we made no further attempt to exploit the algebraic structure of $|s\rangle$. Then, for $q \neq \mathbb{1}$, the value of $\langle s|q|s\rangle$ would be essentially arbitrary and the tightest approximation to $\overline{\text{conv}}(\{\langle s|\mathcal{M}_{2d}|s\rangle\})$ would be given by the matrix inequality

$$\overline{\mathcal{M}}_{2d}^{\sim triv}(q_1,q_2) = \alpha_{q_1q_2}, \quad \alpha_1 = 1, \quad \overline{\mathcal{M}}_{2d}^{\sim triv} \succeq 0. \tag{2.5}$$

So, for instance, if $n = 3$, we would have

$$\overline{\mathcal{M}}_{2}^{\sim triv} = \begin{pmatrix} 1 & \alpha_{Z_1} & \alpha_{Z_2} & \alpha_{Z_3} \\ \alpha_{Z_1} & 1 & \alpha_{Z_1Z_2} & \alpha_{Z_1Z_3} \\ \alpha_{Z_2} & \alpha_{Z_1Z_2} & 1 & \alpha_{Z_2Z_3} \\ \alpha_{Z_3} & \alpha_{Z_1Z_3} & \alpha_{Z_2Z_3} & 1 \end{pmatrix} \succeq 0. \tag{2.6}$$

This turns out to be equivalent to the Goemans-Williamson relaxation for max-cut, which is actually a big problem. The Goemans-Williamson SDP will always upper bound the value of max-cut, while the output of QAOA will always be a lower bound. So, this relaxation will tell us essentially nothing about the performance of QAOA.

2.3. Tighter Relaxation Via Graph Isomorphisms. Why did this happen? We did not enforce enough of the algebraic structure of $|s\rangle$. To remedy this, we will first consider two different approaches involving graph isomorphisms. Firstly, suppose that we have p-layer QAOA and consider a fixed monomial $q = Z_{i_1}\cdots Z_{i_k}$. An examination of the QAOA unitary reveals that the only two-qubit gates are of the form $\exp(i\theta_\ell(\mathbb{1}+Z_iZ_j)/2)$. The $\mathbb{1}$ only adds a global sign, and can thus be ignored, while the factor of $1/2$ can easily be absorbed into θ_ℓ, so it is equivalent to consider gates of the form $\exp(i\theta_\ell Z_iZ_j) = \cos(\theta_\ell)+i\sin(\theta_\ell)Z_iZ_j$. In particular, this means that the value of $\langle s|q|s\rangle$ depends only on $\text{supp}(q) := \{i_1,i_2,\ldots,i_k\}$ and the p-neighborhood of $\text{supp}(q)$ where the p-neighborhood of a set $P \subset V(G)$ is the graph containing all length at most p paths containing at least one element of P. The dependence on $\text{supp}(q)$ is very important. Consider, for instance, the cycle graph C_{10} and two monomials $q_1 = Z_1Z_2Z_4,\ q_2 = Z_1Z_2Z_3Z_4$. Then, the 1-neighborhood of $\text{supp}(q_1)$ and $\text{supp}(q_2)$ are identical, but $\langle s|q_1|s\rangle$ is always zero and $\langle s|q_2|s\rangle$ need not be. We will explore why $\langle s|q_1|s\rangle = 0$ in a later section. All of this can be summed up in the following lemma.

LEMMA 2.2. *Suppose we have a p-layer QAOA circuit $U(\theta,\gamma)$ on a graph G. Given monic Pauli Z monomials q_1 and q_2, consider $Q_1, Q_2 \subset G$, the p-neighborhood of* $\text{supp}(q_1)$ *and* $\text{supp}(q_2)$ *respectively. Color each vertex i of Q_1 (resp. Q_2) blue if $i \in \text{supp}(q_1)$ (resp. q_2) and red otherwise. If there exists a vertex-color preserving graph isomorphism $\phi : Q_1 \to Q_2$, then $\langle s|q_1|s\rangle = \langle s|q_2|s\rangle$.*

Proof. Given subgraphs $G_1, G_2 \subset G$, any graph isomorphism $\varphi : G_1 \to G_2$ induces a unique permutation P_φ of vertices of G where $P_\varphi(i) = \varphi(i)$ if $i \in V(G_1)$ and $P_\varphi(i) = i$ otherwise. By an abuse of notation, we will also let P_φ denote the corresponding permutation on qubits. Note that $P_\varphi|t\rangle = P_\varphi|+^{\otimes n}\rangle = |t\rangle$. Because of this, we can freely apply any permutation as follows

$$\langle s|q|s\rangle = \langle t|U(\theta,\gamma)^\dagger qU(\theta,\gamma)|t\rangle \tag{2.7}$$

$$= \langle t|P_\varphi^\dagger U_{lc}(\theta,\gamma)^\dagger P_\varphi\, P_\varphi^\dagger qP_\varphi\, P_\varphi^\dagger U_{lc}(\theta,\gamma)P_\varphi|t\rangle, \tag{2.8}$$

Where U_{lc} denotes the unitary restricted to the light cone. Taking $\varphi = \phi$, we obtain the equality

$$\langle s| q_1 |s\rangle = \langle t| P_\phi^\dagger U_{lc}(\theta,\gamma)^\dagger P_\phi \, q_2 \, P_\phi^\dagger U_{lc}(\theta,\gamma) P_\phi |t\rangle . \tag{2.9}$$

Note that $P_\phi^\dagger q_1 P_\phi = q_2$ relies on the fact that ϕ is a color-preserving isomorphism. By using the fact that ϕ is a graph isomorphism and thus is also edge-preserving, we arrive at the desired equality

$$\langle s| q_1 |s\rangle = \langle s| q_2 |s\rangle \tag{2.10}$$

as every $\exp(i\gamma_\ell x_j)$ gate will be mapped to $\exp(i\gamma_\ell x_{\phi(j)}$ and every $\exp(i\theta_\ell Z_i Z_j)$ corresponding to the edge ij will get mapped to $\exp(i\theta_\ell Z_\phi(i) Z_\phi(j)$ corresponding to the edge $\phi(ij)$. □

We can exploit this result to achieve a tighter relaxation. We do this by defining an equivalence relation, $\sim_{iso}$ on monic Pauli Z monomials by $q_1 \sim_{iso} q_2$ if a vertex-color preserving isomorphism ϕ exists between the colored p-neighborhoods Q_1, Q_2 as in the lemma. Denote by $[q]_{iso}$ the equivalence class of q under $\sim_{iso}$. Then, we obtain the relaxation

$$\overline{\mathcal{M}}_{2d}^{\sim_{iso}}(q_1,q_2) = \alpha_{[q_1 q_2]_{iso}}, \quad \alpha_{[1]_{iso}} = 1, \quad \overline{\mathcal{M}}_{2d}^{\sim_{iso}} \succeq 0. \tag{2.11}$$

For instance, if $G = C_3$, the cycle on 3 vertices, and $p = 1$, then

$$\overline{\mathcal{M}}_{2d}^{\sim_{iso}} = \begin{pmatrix} 1 & \alpha_{[Z_1]_{iso}} & \alpha_{[Z_1]_{iso}} & \alpha_{[Z_1]_{iso}} \\ \alpha_{[Z_1]_{iso}} & 1 & \alpha_{[Z_1Z_2]_{iso}} & \alpha_{[Z_1Z_2]_{iso}} \\ \alpha_{[Z_1]_{iso}} & \alpha_{[Z_1Z_2]_{iso}} & 1 & \alpha_{[Z_1Z_2]_{iso}} \\ \alpha_{[Z_1]_{iso}} & \alpha_{[Z_1Z_2]_{iso}} & \alpha_{[Z_1Z_2]_{iso}} & 1 \end{pmatrix} \succeq 0. \tag{2.12}$$

Notably, this method requires solving a number of graph isomorphism problems. While the asymptotic runtime of the graph isomorphism problem is not yet known, the only known algorithms are non-polynomial time. However, for a family of graphs $\mathcal{F}$ with uniformly bounded vertex-degree d, such as the family of all d-regular graphs, the size of the subgraphs in question is bounded by d^p and is independent of n.

The method above can be generalized. In particular, given any equivalence relation $\sim$ on monic Pauli Z monomials such that $q_1 \sim q_2$ implies $\langle s| q_1 |s\rangle = \langle s| q_2 |s\rangle$, we can obtain a relaxation

$$\overline{\mathcal{M}}_{2d}^{\sim}(q_1,q_2) = \alpha_{[q_1 q_2]_\sim}, \quad \alpha_{[1]_\sim} = 1, \quad \overline{\mathcal{M}}_{2d}^{\sim} \succeq 0. \tag{2.13}$$

We have already seen this method applied to the trivial equivalence relation $q_1 \sim_{triv} q_2 \iff q_1 = q_2$ and the p-neighborhood graph isomorphism relation. There is, however, another natural equivalence relation arising from the graph automorphism group $\mathrm{Aut}(G)$.

In particular, given the graph automorphism group $\mathrm{Aut}(G)$, we define $q_1 \sim_{\mathrm{Aut}(G)} q_2$ if there exists $\phi \in \mathrm{Aut}(G)$ such that $P_\phi^\dagger q_1 P_\phi = q_2$. Note that the equivalence classes of this relation are exactly the orbits of the group action $\phi \cdot q = P_\phi^\dagger q P_\phi$. This method has a few notable drawbacks. First of all, this equivalence relation is weaker than $\sim_{iso}$ as any $\phi \in \mathrm{Aut}(G)$ mapping q_1 to q_2 will restrict to a graph isomorphism as in $\sim_{iso}$. Second of all, the graph automorphism problem is harder than the graph isomorphism problem. In light of these two issues, this method should only be applied in cases where the graph automorphism group, or at least subgroup, are well understood. Complete k-partite graphs, polytope graphs, circulant graphs, grid graphs, and many other families of graphs have very well understood automorphism groups, and there often exist fast algorithms for finding a canonical representative of each orbit. The complete graph K_n, for instance, has automorphism group $\mathrm{Aut}(K_n) = S_n$ the symmetric group on n elements. Despite this group containing $n!$ elements, the orbits of $\mathrm{Aut}(K_n)$ are simply the monomials of equal degree.

2.3.1. Numerical Results. Here, we demonstrate the sum of squares relaxation described above applied to max-cut on the cycle graph. On the cycle in particular, both of the graph isomorphism methods will actually yield exactly the same relxation, so we need not worry about the distinction here.

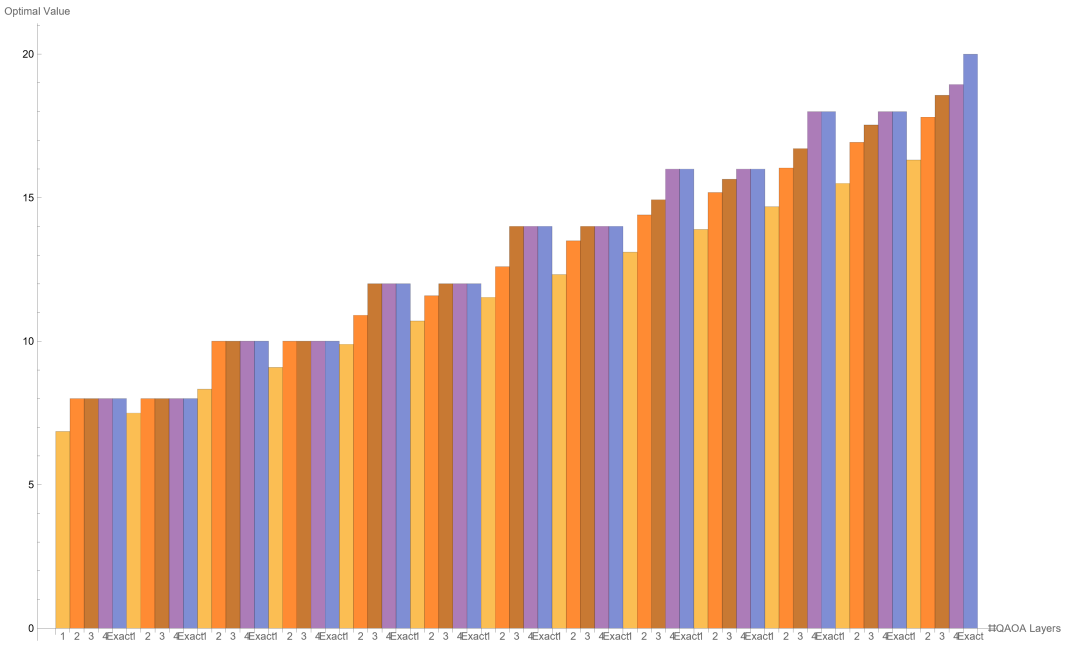


FIG. 2.1. *Degree* $2d = 4$ *SoS Upper Bound on QAOA on the* n*-vertex cycle graph. Each set of* 5 *columns corresponds to one value of* n*, ranging from* 8 *to* 20*, and each column corresponds to a different* p *value. In blue is the exact max-cut value.*

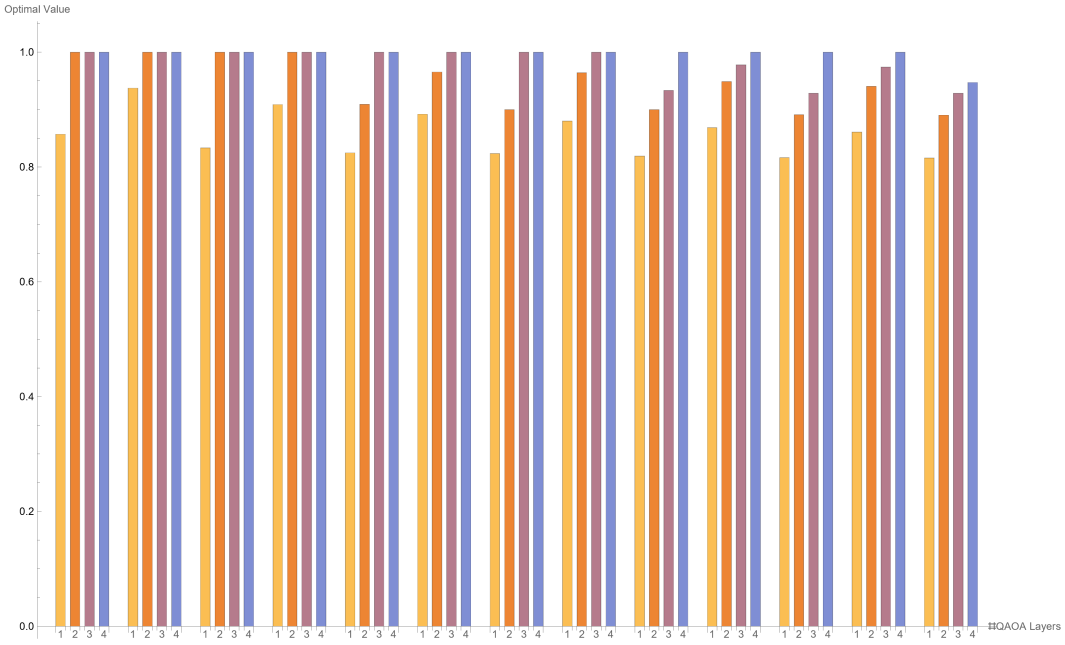


FIG. 2.2. *Degree $2d = 4$ SoS Upper Bound on QAOA approximation ratio on the n-vertex cycle graph. Each set of 4 columns corresponds to one value of n, ranging from 8 to 20, and each column corresponds to a different p value. Note that for all $p > 1$, the approximation ratio is better than 0.878.*

Of importance is that in these tests, a level $d = 2$ relaxation was used. Had we used a higher level relaxation, we would have obtained tighter bounds on QAOA performance. This choice was primarily made out of practical considerations for runtime.

2.4. Polynomial Entries. So far every relaxation we have explored has produced a constraint matrix with monomial entries. In order to get a better approximation, we must generalize to a constraint matrix with polynomial entries. Towards this, we must first establish a number of preliminary results.

LEMMA 2.3. *Suppose q is a non-zero Pauli monomial. Then,*

$$\langle t| q |t\rangle = 0 \iff \deg_Z(q) = 0$$

Proof. Note that $X_i |t\rangle = |t\rangle$. So, without loss of generality, we can assume q has no X degree. Then, note that $\langle t| Z_i |t\rangle = \langle t| |+\cdots+-+\cdots+\rangle = 0$. Thus, $\langle t| q |t\rangle = 0$. For the reverse, suppose $\deg_Z(q) = 0$. Then, $q = \alpha X_{i_1} \cdots X_{i_k}$ for $\alpha, k \neq 0$. Then $\langle t| q |t\rangle = \alpha \neq 0$. ∎

LEMMA 2.4. *Suppose q is a Pauli Z monomial with odd degree. Then, we have*

$$\langle s| q |s\rangle = 0.$$

Proof. Note that while $\deg(q_1 q_2)$ does not necessarily equal $\deg(q_1)\deg(q_2)$, these two values must be equal modulo 2. This is because the only relation that decreases the string length of a monomial is $Z_i^2 = 1$, which maintains parity. Thus, if we expanded $U(\theta, \gamma)$ in terms of X and Z, the Z degree of each monomial in the expansion would be even. So, if we fully expanded $U(\theta, \gamma)^\dagger q U(\theta, \gamma)$ in terms of X and Z, the Z degree of each monomial in the expansion would be odd and thus non-zero. So, by Theorem 2.3, we are done. ∎

LEMMA 2.5. *Given the p-layer QAOA circuit on graph G, for any Pauli Z monomial q, the value $\langle t| U(\theta, \gamma)^\dagger q U(\theta, \gamma) |t\rangle$ is a trigonometric polynomial in θ and γ. Moreover, the*

non-zero terms of this trigonometric polynomial correspond to tuples

$$(H_1^\ell, H_1^r, H_2^\ell, H_2^r, \ldots, H_p^\ell, H_p^r)$$

of subgraphs $H_i^h \subset N_i(\mathrm{supp}(q))$*, the* i*-neighborhoods of* $\mathrm{supp}(q)$*, such that for all* $i \in \mathrm{supp}(q)$*, the sum* $\sum_{j=1}^p (\deg_{H_j^\ell}(i) + \deg_{H_j^r}(i))$ *is odd, but for all* i *not in* $\mathrm{supp}(q)$ *the sum* $\sum_{j=1}^p (\deg_{H_j^\ell}(i) + \deg_{H_j^r}(i))$ *is odd.*

Proof. We first note that $\exp(i\theta_\ell Z_i Z_j) = \cos(\theta_\ell) + i\sin(\theta_e ll) Z_i Z_j$ and $\exp(i\gamma_\ell X_j) = \cos(\gamma_\ell) X_i$. So, expanding $U(\theta,\gamma)^\dagger q U(\theta,\gamma)$ in terms of X and Z will yield a Pauli polynomial with trigonometric coefficients. Call this q^*. Thus, distributing $\langle t| q^* |t\rangle$ will yield only a sum of the trigonometric coefficients, which is a trigonometric polynomial. Note that the coefficients of this trigonometric polynomial must be real integers. To see this, note that

$$\left(U(\theta,\gamma)^\dagger q U(\theta,\gamma)\right)^\dagger = U(\theta,\gamma)^\dagger q^\dagger U(\theta,\gamma) = U(\theta,\gamma)^\dagger q U(\theta,\gamma), \tag{2.14}$$

as each Z_i is self-adjoint and commutes with every other Z_j. Thus, $\langle t| U(\theta,\gamma)^\dagger q U(\theta,\gamma) |t\rangle$ must be real. We will call this real trigonometric polynomial q^{**}.

We now aim to characterize the nonzero terms in q^*. Note by Theorem 2.3, a non-zero monomial term of q^* will contribute a non-zero value to q^{**} if and only if it has no Z degree. Thus, it is sufficient to study which conditions yield zero-Z-degree monomials. Towards this, we follow the classic combinatorialist's proof of the binomial theorem. For simplicity, we will first consider the case $p = 1$. Consider the unexpanded form of q^*

$$\prod_{ij \in E_{lc}(G)} (\cos(\theta_1) - i\sin(\theta_1) Z_i Z_j) \prod_{j \in V_{lc}(G)} \exp(-i\gamma_1 X_j)\ Z_{i_1} \cdots Z_{i_k} \text{h.c.} \tag{2.15}$$

where by E_{lc} and V_{lc} we mean the edges and vertices in the p-neighborhood of $\mathrm{supp}(q)$ and by h.c. we mean the Hermitian conjugate of the left-hand unitary.. Expanding this product can be expressed as a sum over all monomials generated by a given string of choices of either $\cos(\theta_1)$ or $i\sin(\theta_1) Z_i Z_j$. The monomials in this sum with zero-Z-degree, thus, correspond to monomials where each Z_{i_ℓ} appears in an odd number of choices of $\sin(\theta_1) Z_i Z_j$ and all other Z_i terms occur an even number of times. We can encode these choices in a "left" and "right" subgraph of $N(\mathrm{supp}(q))$, H_1^ℓ, H_1^r, where an edge ij appear in H_1^ℓ if we pick $i\sin(\theta_1) Z_i Z_j$ on the left-hand side and the edge ij appearing in H_1^r if we pick $i\sin(\theta_1) Z_i Z_j$ on the right. Thus, the parity condition described above can be checked by checking the parity of $\deg_{H_1^\ell}(i) + \deg_{H_1^r}(i)$.

To generalize to $p > 1$, we simply use a list $(H_1^\ell, H_1^r, H_2^\ell, H_2^r, \ldots, H_p^\ell, H_p^r)$ of left and right subgraphs to encode our decision instead of just one. Each H_i^h is a subgraph of $N_i(\mathrm{supp}(q))$, the i-neighborhood of $\mathrm{supp}(q)$. To check that the resulting monomial with have Z degree of zero, we simply check that for all $i \in \mathrm{supp}(q)$, the sum $\sum_{j=1}^p (\deg_{H_j^\ell}(i) + \deg_{H_j^r}(i))$ is odd, but for all i not in $\mathrm{supp}(q)$ the sum $\sum_{j=1}^p (\deg_{H_j^\ell}(i) + \deg_{H_j^r}(i))$ is odd. □

Given a subgraph tuple as described in Theorem 2.5, it does not take much more work to obtain the corresponding trigonometric monomial exactly in terms of $\cos(\theta_\ell)$, $\sin(\theta_\ell)$ and $\exp(i\gamma_\ell)$. The two observation necessary are that $\exp(i\gamma_\ell X_j) |t\rangle = \exp(i\gamma_\ell) |t\rangle$ and $Z_j \exp(i\gamma_\ell X_j) = \exp(-i\gamma_\ell X_j) Z_j$. We can further expand $\exp(i\gamma_\ell) = \cos(\gamma_\ell) + i\sin(\gamma_\ell)$ to obtain q^{**}. Once we have q^{**} in full, expanded form, we can proceed in a number of ways. Firstly, we could simply apply a preexisting trigonometric sum of squares technique such as the one described [1]. Alternatively, we can simply replace every trigonometric monomial with a corresponding arbitrary commuting variable and proceed as in previous sections. In particular, suppose $\mathcal{C}_d$ is a monomial basis of all the degree-at-most d trigonometric

polynomials in θ and γ, and we associate to every $c \in \mathcal{C}_d$ a variable x_c with $x_1 = 1$. If $x := (x_c)_{c\in\mathcal{C}}$, the resulting relaxation would have the form

$$\overline{\mathcal{M}}_{2d}(q_1, q_2) = b^{\dagger}_{q_1 q_2} x, \quad b_1 = (\delta_{1,c})_{c\in\mathcal{C}_D}, \quad \overline{\mathcal{M}}_{2d} \succeq 0, \tag{2.16}$$

where D is the smallest degree necessary to represent every trigonometric monomial that occurs in q^{**}.

2.4.1. Future Directions. While the method described above is powerful, it is also the most computationally intensive. The restriction to the light cone does mean that the complexity does not grow with n, the size of the graph, the complexity is doubly exponential in d, the degree of the graph, and p. As such, a hybrid method employing both polynomial replacement based on subgraph enumeration and monomial replacement based on graph isomorphisms is a strong candidate for a happy medium between accurate and computationally tractable. We do not currently have any numerical results to demonstrate the accuracy of such a hybrid method, but it is a promising future direction as it should enable us to tractably compute upper bounds for larger graphs.

REFERENCES

[1] F. Bach and A. Rudi, *Exponential convergence of sum-of-squares hierarchies for trigonometric polynomials*, SIAM Journal on Optimization, 33 (2023), pp. 2137–2159.
[2] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, *Variational quantum algorithms*, Nature Reviews Physics, 3 (2021), p. 625–644.
[3] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum approximate optimization algorithm*, 2014.
[4] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM, 42 (1995), pp. 1115–1145.
[5] J. B. Lasserre, *Global optimization with polynomials and the problem of moments*, SIAM Journal on Optimization, 11 (2001), pp. 796–817.
[6] C. Ryan-Anderson, *Quantum algorithms, architecture, and error correction*, UNM Physics and Astronomy ETDs, (2018).
[7] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel, *Quantum approximate optimization algorithm for maxcut: A fermionic view*, Phys. Rev. A, 97 (2018), p. 022304.

EXPERIENCE REPORT ON OBSERVABILITY AND ITS EFFECT ON SECURITY AND USABILITY IN SOFTWARE SYSTEMS

ANAGHA KRISHNA* AND REED MILEWICZ†

Abstract.
This report delves into the concept of observability for software systems, focusing on its role in enhancing security and usability—two fundamental attributes of high-quality software. Through an analysis of two case studies, ECMF and PROVERS, it illustrates how observability facilitates critical insights into software operations and control mechanisms, despite facing challenges in environments where comprehensive oversight of the codebase and infrastructure is not feasible. While observability mechanisms can significantly enhance system security by identifying vulnerabilities and monitoring for anomalies, they also introduce risks related to the potential exposure of sensitive data. Additionally, the report demonstrates that observability is instrumental in measuring usability, highlighting how specific metrics, traces, and logs can help gain a deeper understanding of user interactions with software features. This data-driven approach allows for informed design decisions that enhance user experience.

1. Introduction. Modern software systems are becoming increasingly complex, highly scalable, and are transitioning towards a microservices-based architecture [30]. Multiple cross-functional teams work on different components that are integrated seamlessly. It is difficult for a single person or team to have a deep technical understanding of the entire system. Therefore, it would be extremely beneficial to keep track of everything that is happening in the system internally, and this can be achieved by enabling observability or by having an observable software system. Reflecting on observability in relation to the two projects we worked on during the summer, two other characteristics of software systems that stood out were security and usability.

Software security refers to the degree to which software protects information and system resources, providing access only to authorized users as intended. The goal is to prevent, detect, and recover from attacks and unintended weaknesses that could compromise confidentiality, integrity, and availability of software systems and data. Software security seeks to build in protections proactively to avoid having to react to threats. It involves vigilance across requirements, design, coding, testing, deployment, and maintenance [14].

According to ISO/IEC 25010:2011, usability refers to the ability of users of a software system to "achieve specified goals with effectiveness, efficiency, and satisfaction" within the intended context of use [18]. Usability encompasses not only the user's experience (*i.e.,* "quality in use", including effectiveness, efficiency, satisfaction, freedom from risk, and context coverage) but also how well software is intrinsically designed (*i.e.,* "product quality", including appropriateness recognizability, learnability, operability, use error protection, and user interface aesthetics). For a software application, usability indicates the ease with which users can interact with and navigate through an application to accomplish their tasks effectively and efficiently. It encompasses various aspects such as user interface design, navigation, responsiveness and error handling, thus impacting the overall user experience [16].

Both security and usability are important features for maintaining a high-quality software system. Consequently, we have chosen to address the following research questions:

- **RQ1** How does implementing observability intersect with security in a software system?
- **RQ2** Can enabling observability in a software system help with measuring its usability?

*Sandia National Laboratories, ankrish@sandia.gov
†Sandia National Laboratories, rmilewi@sandia.gov

The remainder of this experience report is as follows. Section 2 provides an overview of observability. In section 3, there is a description of the projects I worked on during the summer, along with a list of tasks I completed for each project. Section 4 summarizes related work. Section 5 provides an analysis for the research questions based on related work. Section 6 is a discussion on observability and lessons learned during the summer internship. Section 7 concludes the report.

2. Background. "Observability" was originally coined by Rudolf E. Kálmán for control theory to refer to how well internal states of a system can be inferred from knowledge of its external outputs. A system is considered "observable" to the extent that the current state can be estimated by only using information from output it generates [12]. Some characteristics of an observable application are: (1) One can understand any system state the application may have gotten itself into, even new ones that weren't encountered before and couldn't have been predicted (2) One can understand the inner workings and system state solely by observing and interrogating with external tools.

Observability offers the tools and practices necessary to comprehend and control complex software systems, providing visibility into how different parts of the system interact. Through this, organizations can gain insights into their applications and infrastructure performance, identifying bottlenecks, inefficient code paths, and resource utilization patterns. When incidents occur, observability tools assist teams in quickly pinpointing the root cause of the problem. The goal of observability is to provide a level of introspection that helps people reason about the internal state of their systems and applications [24].

A necessary component of observability is the creation of useful instrumentation. Instrumentation is process of adding or enhancing code to measure certain aspects of its execution, such as performance, resource usage, or to detect and diagnose complex issues like memory leaks or concurrency problems. In practice, instrumenting code involves emitting traces, metrics, or logs. And, the instrumented data is generally sent to an observability backend, which is a service that receives trace information and metrics from instrumented apps for analysis [13]. Logs, metrics, and traces are often known as the three pillars of observability. Engineers or developers should strive to instrument their code such that they can answer these questions as soon as it's deployed:

- Is your code doing what you expected it to do?
- How does it compare to the previous version?
- Are users actively using your code?
- Are any abnormal conditions emerging?

A popular observability framework used for instrumenting code is OpenTelemetry [13]. It is an open-source observability framework that provides a set of tools, APIs, and SDKs. It is designed to standardize the collection, processing, and transmission of telemetry data (metrics, logs, and traces) across cloud-native applications and services. OpenTelemetry provides two primary ways for instrumenting code: (1) code-based instrumentation and (2) zero-code instrumentation.

- **Code-based** solutions offer the opportunity to get deeper insight and rich telemetry directly from the software application. These solutions make use of the OpenTelemetry API to generate telemetry, which acts as a complement to the telemetry generated by zero-code solutions.
- **Zero-code** solutions are great for getting started, or when there is no access to the application source code, or when the application code can not be modified. These solutions provide rich telemetry from the libraries present in the source code or from the environment used for running the application [6].

Observability helps with getting the complete picture of what is going on in the software

system, hence making it easier to identify performance issues, understand dependencies and interactions between different services and components in the system. Since, DevOps [11] and Site reliability engineering (SRE) [15] teams are required to understand production systems and tame complexity, observability is both a consequence and an integral part of the DevOps, SRE, and cloud native movements [9, 24].

Due to the characteristics and requirements for telemetry data (metrics, traces, logs) discussed below, it is challenging to have one universal observability solution for all kinds of software systems. Telemetry data is immutable and append-heavy. Since, most observable systems run in dynamic distributed environments, the volume of data generated over time is highly variable. Data generated more recently (fresh data), needs to be more accessible and available than older data (historical data), because the former is more likely to be queried than the latter. While fresh data needs better accessibility and availability, you may never know when you might require older historical data. Hence, telemetry data must be managed in such a way that there is low query latency, low operational complexity while maintaining appropriate infrastructural costs [22].

3. Experiential Context.

3.1. The Engineering Common Modeling Framework (ECMF). The ECMF project is a platform for computational model sustainment at Sandia National Laboratories. A model registered on ECMF can be discovered, reused, extended and evaluated over time. This is helpful because current computational models are developed manually, making them difficult to maintain, share, curate and collaborate on. ECMF's architectural design is microservices-based and has Docker containers for its components: database, user interface, API, authentication and scheduler. Some of the tools and technologies used in ECMF are: MongoDB for the database; React and Next.js for the frontend; Starlette and Kubernetes python client for the RESTful APIs [10] that communicate between the frontend and OpenShift (platform where the ECMF containers are deployed). The workflow results and artifacts are stored in a Sandia-hosted instance of Amazon's Simple Storage Service (S3) [19].

During the summer, I worked on backend (server-side) and frontend (client-side) [4] tasks. I had the opportunity to learn about docker compose, yaml files, Kubernetes API, Fast API, Starlette and more. I created a docker compose file that helps with spinning up multiple Docker containers with one command, thereby reducing the time spent on bringing them up individually. I developed two more RESTful API endpoints for the ECMF API component, the first one being an API for downloading a pre-signed URL generated by ECMF's S3 implementation. Since all S3 objects are private by default, the object owner can create a pre-signed URL that provides limited time permission to download the object [8]. I also built a download button on the frontend for calling that API and downloading the pre-signed URL locally. The second RESTful API endpoint is to obtain a streaming download of a file that contains workflow results.

3.2. Pipelined Reasoning of Verifiers Enabling Robust Systems (PROVERS). The objective of the PROVERS program is to:

- Develop automated, scalable formal methods tools that are integrated into traditional software development pipelines.
- Enable traditional software developers to incrementally produce and maintain high-assurance national security systems.

Different entities involved in PROVERS are:

- TA1 (Technical Area 1), Proof Engineering [7]: is responsible for developing Proof engineering tools

- TA2 (Technical Area 2), Platform Development [2, 3]: provide publicly available and national security system platforms and use tools developed by TA1
- TA3 (Technical Area 3), Red Team: perform a security assessment with a high-quality red team that conducts baseline assessments before any modifications and then reexamines each high-assurance version at each consecutive phase of the project
- Quantitative Evaluation and Evidence Curation: perform periodic quantitative technology assessment of TA1 technologies, leveraging TA2 traditional engineer personnel in support of usability assessment

PROVERS tools are intended to achieve the following:

- Provide scalable automation targeted at traditional software developers
- Integrate into standard software workflows
- Enable incremental maintenance processes
- Be refined via a continuous feedback process
- Create demonstrably more secure software based on red-team analysis [1]

To support Quantitative Evaluation and Evidence Curation, during the summer, I explored tools like OpenTelemetry and Jaeger, learned about observability, instrumentation and different telemetry data types. I also instrumented some sample python programs with OpenTelemetry to generate traces and exported them to Jaeger for further analysis.

4. Related Work. The intersection of telemetry data (such as metrics and logs) and security in software systems has been explored across various studies, revealing both its strengths and limitations. Upon considering software systems similar to ECMF which have implemented observability to understand the kind of impact an observable system can have on its security, the following was found: System and application logs present important data such as timestamps, file sizes, and CPU usage [23]. These logs are crucial for both scientific research and industrial analysis. While there's a growing need to share these logs with external analysts to improve system performance, companies are often hesitant due to concerns about information leakage. Logs can inadvertently reveal sensitive details about the systems, applications, or even future products, especially when the logs are linked to real-world production workloads. Long et al. [23] also present a framework can prevent sensitive information leakage while maintaining acceptable levels of information loss and processing time.

Avila et al. [33] talk about using logs for detecting data leaks, their critical role in identifying cyber attacks, and proposes a novel classification of information leaks that aligns with GDPR [5] principles, emphasizing the importance of data privacy and protection. The paper mentions that logs are a crucial tool for detecting and responding to threats in real-time, although they also present challenges such as large data volumes and heterogeneity. Medeiros et al. [25] provide an analysis on software metrics in different architectural levels of five different software projects; they state that there is a strong correlation between several project-level metrics and the number of reported vulnerabilities, and that it is possible to use a group of metrics to distinguish vulnerable and non-vulnerable units of code with a high level of accuracy. Medeiros et al. [26] presented a comprehensive study on the use of software metrics and machine learning algorithms for the detection/prediction of vulnerable code. The most important observation is that using machine learning algorithms on top of software metrics helps identifying vulnerable code units with relatively high level of confidence for security-critical software systems but they are not helpful for low-critical or non-critical systems due to the relatively high number of false positive alarms reported when compared to the number of true positive cases.

Zhi et al. [31] provide a study on sensitive information exposure through logging, they state that, about two-third vulnerabilities can be exploited via the network, and a signifi-

cant number of vulnerabilities can be exploited with low efforts. About half of the exploits can be used by malicious users and insiders without any expertise to launch attacks. The authors further claim that, the top 3 common root causes for the vulnerabilities are insecure whole-object logging, incorrect permission assignment, and improper implementation of sanitization. Inadequate logging practices can significantly compromise security by leaking sensitive data. In mobile applications, improper logging often leads to the exposure of sensitive information, such as passwords or location data, through log files; 72 percent of apps with identified taint flows were affected by poor logging, resulting in substantial data leakage [32]. Vulnerabilities like local file inclusion (LFI) can allow attackers to inject malicious code into log files, which can be executed if these logs have improper permissions. This can lead to unauthorized access or persistent threats, such as backdoors, compromising the security of the system. Consequently, without appropriate safeguards, logs can become a significant security risk rather than a protective measure [27]. The reviewed papers collectively highlight that logs and metrics can both support and undermine security, depending on their management and implementation. While logs provide critical insights for security, they also present risks if not adequately protected. Metrics are valuable for monitoring and detecting anomalies but require careful management to be effective. Hence, a balanced approach that involves taking proactive measures and secured telemetry data management is essential for leveraging observability to enhance security.

For use cases similar to PROVERS where the system is made observable to further evaluate another attribute like usability, a brief high-level summary of related work is discussed below. Grossman et al. [20] provide a comprehensive survey of the definitions, metrics and evaluation methodologies used for software learnability, which is an important aspect of usability. The authors conducted a survey of existing research, developing a taxonomy to better understand how learnability can be assessed. The paper introduces a new question-suggestion protocol for usability testing, comparing it with the traditional think-aloud method. Seffah et al. [28] highlight the challenges in applying usability measurement standards due to inconsistent definitions and frameworks across different models; and proposes the Quality in Use Integrated Measurement (QUIM) model, which combines various usability metrics and criteria, supports the idea that observability data can be systematically analyzed to assess usability. By consolidating metrics and providing a structured approach to their application, the QUIM model demonstrates how observability can help in measuring usability effectively. Burton et al. [17] demonstrate the value of integrating various data collection methods, such as server logs and client-side logs and usability testing to enhance Web design. While these approaches can be resource intensive for professional designeers, they provide a solid understanding of user interactions and can lead to design improvements, which can in-turn make the system more usable. User Interface (UI) events, naturally generated during software usage can give valuable insights into usability by capturing end-user interactions with an application. These events, which include actions like mouse clicks, keyboard presses and menu selections, can be analyzed to identify usage patterns, detect usability issues and understand user behavior. Usability teams in large software development organizations are often approached by design and development team members with questions like “how often do users do X?” or “how often does Y happen?” [21].

5. Analysis.

5.1. RQ1: How does implementing observability intersect with security in a software system?. Here, the objective is to identify whether observability has a positive/negative/neutral effect on security for software appliactions. Evidently, from the related work, there are software systems where metrics and logs have been found to be both fa-

vorable and detrimental. For a system like ECMF, there is a trade-off between the risk of possibly exposing sensitive data and the benefit of obtaining internal system data for understanding the system better or detecting vulnerabilities. From section 2 and section 4, it is understood that there is no "one size fits all" solution for observability; and the security implications of an observable system is subjective and depends on each particular use case. The same feature (log or metric) that poses as a potential security risk can be used to identify vulnerabilities when used properly by taking necessary precautions.

5.2. RQ2: Can enabling observability in a software system help with measuring its usability?. The aim here is to understand if one can determine how usable a software system is, from its internal telemetry data. From section 4, it can be summarized that the integration of server and client-side logging, along with automated event analysis and traditional usability testing methods can contribute towards a robust framework for measuring usability in software systems. Telemetry data can be used to obtain ordinal data such as the number of times a particular action is performed by the user, it can also be used to measure performance metrics such as the degree to which users can accomplish a task successfully [29]. Enabling observability involves continuous monitoring, real-time data collection and provides insights into user behaviours, hence, offering a more comprehensive, data-driven approach to understanding how users interact with the system.

6. Discussion. In the context of an in-house software system like ECMF, where developers have access to and can modify the source code, (after making necessary considerations and getting necessary approvals from management), an implementation strategy for observability would probably involve using appropriate APIs and SDKs (provided by an observability framework) to instrument code, gathering the telemetry data and exporting it to other supporting tools (which serve as observability backends) for additional analysis and visualization. However, in the case of PROVERS, specifically for Sandia's role in the project, the tools will be developed and instrumented by other parties. There is no control over the source code and telemetry data will be given for analysis. Formulating a strategy for enabling observability in contexts where there is no software ownership, brings out a different kind of challenge and clearly, the previously mentioned approach for in-house projects will not hold good. Therefore, there is a need for more tailored guidelines to address observability in different contexts. A future direction would be to investigate how such guidelines can be developed to effectively navigate the complexities of observability across various software environments.

7. Conclusion. In this paper, an overview on observability has been provided, observability has been looked at, in the perspective of two real-world projects: ECMF and PROVERS, and with respect to "security" and "usability", which are two prominent aspects of high-quality software systems. Observability, while offering profound insights and control over software environments, is not without its challenges. It thrives in scenarios where there is comprehensive control over the codebase and infrastructure, highlighting a limitation in environments where such control is not feasible. Despite these constraints, the implementation of observability mechanisms remains invaluable. The extent to which, an application must be made observable is a key factor to consider. The ability to collect telemetry data, even with potential risks of exposing sensitive information, plays a pivotal role in enhancing system security when managed with due diligence. Many usability studies involve understanding how frequently a user interacts with certain features in a software application or how much time a user spends on a certain feature, and such information can be obtained from telemetry data.

REFERENCES

[1] https://www.darpa.mil/program/pipelined-reasoning-of-verifiers-enabling-robust-systems.

[2] *Computing platform - Wikipedia — en.wikipedia.org.* https://en.wikipedia.org/wiki/Computing_platform.

[3] *Development Platform - an overview — ScienceDirect Topics — sciencedirect.com.* https://www.sciencedirect.com/topics/computer-science/development-platform.

[4] *Frontend and backend - Wikipedia — en.wikipedia.org.* https://en.wikipedia.org/wiki/Frontend_and_backend.

[5] *General Data Protection Regulation - Wikipedia — en.wikipedia.org.* https://en.wikipedia.org/wiki/General_Data_Protection_Regulation.

[6] *Instrumentation — opentelemetry.io.* https://opentelemetry.io/docs/concepts/instrumentation/.

[7] *Proof Engineering — proofengineering.org.* https://proofengineering.org/.

[8] *Sharing objects with presigned URLs - Amazon Simple Storage Service — docs.aws.amazon.com.* https://docs.aws.amazon.com/AmazonS3/latest/userguide/ShareObjectPreSignedURL.html.

[9] *The Cloud Native Movement — linkedin.com.* https://www.linkedin.com/pulse/cloud-native-movement-ryan-perkins.

[10] *What is a REST API? — redhat.com.* https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[11] *What is DevOps?? — about.gitlab.com.* https://about.gitlab.com/topics/devops/.

[12] *What is Observability? An Introduction — Splunk — splunk.com.* https://www.splunk.com/en_us/blog/learn/observability.html.

[13] *What is OpenTelemetry? — opentelemetry.io.* https://opentelemetry.io/docs/what-is-opentelemetry/.

[14] *What is software security and why is it important? — computer.org.* https://www.computer.org/resources/software-security.

[15] *What is SRE? — redhat.com.* https://www.redhat.com/en/topics/devops/what-is-sre.

[16] *What Is Usability And Why Does It Matter In App Development? — ux4sight.com.* https://ux4sight.com/blog/what-is-usability-and-why-does-it-matter-in-app-development.

[17] M. C. Burton and J. B. Walther, *The value of web log data in use-based design and testing*, Journal of Computer-Mediated Communication, 6 (2006), pp. 0–0.

[18] I. O. for Standardization, *ISO/IEC 25010:2011, systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models*, Tech. Rep. ISO/IEC Standard No. 25010:2011, 2011.

[19] C. Gilbertson, R. Milewicz, E. Berquist, A. Brundage, J. Engelmann, B. Evans, N. Francis, E. Fridman-Hill, S. Grayson, E. Harvey, E. Ho, E. Hoffman, K. Irick, A. Krishna, A. Moreno, and J. Teves, *Towards long-term scientific model sustainment at sandia national laboratories (under review)*, 2024.

[20] T. Grossman and R. Fitzmaurice, George amd Attar, *A survey of software learnability: metrics, methodologies and guidelines*, in Proceedings of the SIGCHI COnference on Human Factors in Computing Systems, CHI '09, Association for Computing Machinery, 2009, pp. 649–658.

[21] D. M. Hilbert and D. F. Redmiles, *Extracting usability information from user interface events*, ACM Computing Surveys, 32 (2000), pp. 384–421.

[22] S. Karumuri, F. Solleza, S. Zdonik, and N. Tatbul, *Towards observability data management at scale*, SIGMOD Rec., 49 (2021), p. 18–23.

[23] Y. Long, L. Xu, and C. A. Gunter, *A hypothesis testing approach to sharing logs with confidence*, in Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy, CODASPY '20, New York, NY, USA, 2020, Association for Computing Machinery, p. 307–318.

[24] C. Majors, L. Fong-Jones, and G. Miranda, *Observability Engineering*, " O'Reilly Media, Inc.", 2022.

[25] N. Medeiros, N. Ivaki, P. Costa, and M. Vieira, *Software metrics as indicators of security vulnerabilities*, 10 2017.

[26] N. Medeiros, N. Ivaki, P. Costa, and M. Vieira, *Vulnerable code detection using software metrics and machine learning*, IEEE Access, 8 (2020), pp. 219174–219198.

[27] H. Noman, O. Abu-Sharkh, and S. Noman, *Log poisoning attacks in iot: Methodologies, evasion, detection, mitigation, and criticality analysis*, IEEE Access, PP (2024), pp. 1–1.

[28] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, *Usability measurement and metrics: A consolidated model*, Software Quality Journal, 14 (2006), pp. 159–178.

[29] T. Tullis and W. Albert, *Measuring the User Experience, Second Edition: Collecting, Analyzing, and Presenting Usability Metrics*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd ed., 2013.

[30] Y. Wang, H. Kadiyala, and J. Rubin, *Promises and challenges of microservices: an exploratory*

study, Empirical Software Engineering, 26 (2021).

[31] C. Zhi, J. Yin, J. Han, and S. Deng, *A preliminary study on sensitive information exposure through logging*, in 2020 27th Asia-Pacific Software Engineering Conference (APSEC), IEEE, Dec. 2020.

[32] R. Zhou, M. Hamdaqa, H. Cai, and A. Hamou-Lhadj, *Mobilogleak: A preliminary study on data leakage caused by poor logging practices*, in 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2020, pp. 577–581.

[33] R. Ávila, R. Khoury, R. Khoury, and F. Petrillo, *Use of security logs for data leak detection: A systematic literature review*, Security and Communication Networks, 2021 (2021), p. 6615899.

ANALYZING QUBIT-RUNTIME TRADEOFFS IN PARALLELIZING UNARY ITERATION

C. O'NEIL,*, M. D. PORTER,†, AND S. K. SERITAN‡

Abstract.
As the development of scalable, fault-tolerant quantum computers progresses, optimizing quantum resources in circuit design remains a critical challenge. Traditional efforts have focused on minimizing qubit count and the number of resource-intensive T gates (T count). Unary iteration is a critical subroutine in several state-of-the-art quantum algorithms. In this work, we demonstrate how to parallelize unary iteration to reduce T depth at the cost of more qubits and analyze the resulting qubit-runtime tradeoffs. Preliminary results demonstrate constant factor speedups over the serial algorithm with minimal additional qubit overhead while preserving linear T count. We also study several partial parallelization techniques and show in which regimes each strategy yields a better speedup. Our ongoing research explores the full potential of employing dirty qubits and seeks to refine qubit routing and scheduling techniques for further speed improvements.

1. Introduction. Quantum computers are heralded for their potential to surpass classical computers in solving specific, complex problems, particularly in the realm of quantum simulation. A prime example within material science is the calculation of the ground state energy of an atomic ensemble. On classical computers, this task becomes increasingly untenable as the size of the ensemble grows, primarily due to the general exponential growth of entanglement between electrons. This challenge, first put forth by Richard Feynman [4, 5], suggested that quantum computers could offer a viable solution, a notion that has since captivated the interest of researchers across multiple fields including physics, chemistry, and material science. Feynman's theoretical proposition laid the groundwork for what would later be formalized by Lloyd as "Universal Quantum Simulators" in his seminal 1996 article [2, 10]. This foundational concept underscores the transformative potential of quantum computing in simulating complex quantum systems, a capability that extends across multiple scientific disciplines.

Circuit-based quantum computers perform computations through sequences of quantum gates, with the Clifford + T gate set being a popular choice for enabling universal quantum computing [14]. This particular set is known for its compatibility with error detection and correction codes—a critical component for realizing fault-tolerant quantum computing [1, 6, 14–16]. The implementation of the Clifford + T gates allows for versatile and reliable quantum computing, addressing a significant challenge in the field.

Circuit-based quantum computers will require certain physical resources to carry out a given computation. In the literature, resources are typically quantified in terms of the number of qubits and the T count (number of T gates in the circuit) [2]. Previously, T gates have been considered excessively resource-intensive, both in terms of their demand on physical resources and their impact on computational runtime. However, some recent studies have indicated that the costs may not be as exorbitant as once thought [9, 15]. Despite this reassessment, Ref. [15] notes that T gates will still be more expensive than other gates and should therefore be used as little as possible.

In addition to qubit number and T count, runtime considerations are also essential. For quantum circuits employing the Clifford + T gate set, the runtime is primarily determined by the T depth, a metric quantifying the number of T gates that can be implemented sequentially. Thus, optimizing quantum circuit efficiency involves minimizing both the T

*New Mexico State University, coneil@nmsu.edu
†Sandia National Laboratories, mdport@sandia.gov
‡Sandia National Laboratories, sserita@sandia.gov

count and the T depth wherever feasible.

In 2017, Babbush et al. [2] introduced a method known as"unary iteration," a subroutine within the qubitization protocol [11] to construct quantum circuits capable of encoding the spectra of correlated electrons with linear T complexity. Qubitization requires a block encoding of the Hamiltonian, which is often achieved through the linear combination of unitaries (LCU) technique [3]. The unary iteration subroutine achieves linear T count and requires only logarithmic auxiliary qubits. The method executes the multicontrolled Hamiltonian terms by first computing the multicontrolled Toffoli onto an auxiliary qubit, performing a singly-controlled operation for the Hamiltonian term, and then iterating to the next multicontrolled term. Through clever storage of partial AND computations on $\log_2(N) - 1$ auxiliary qubits, the iterative computation of all multicontrolled Toffolis the auxiliary qubit that controls the singly-controlled operations can be done with only $N - 1$ Toffoli computations (where each Toffoli requires 4 T gates, as shown in Fig. 2.1a). Although the possibility of parallelizing the unary iteration method was noted by Babbush et al. [2], such an approach has yet to be thoroughly explored or analyzed.

In this study, we explore the prospect of enhancing quantum circuit efficiency through both full and partial parallelization of the unary iteration method. While achieving full parallelization results in the most significant reduction in T depth, leading to optimal speedup, it also incurs a substantial increase in the number of required auxiliary qubits, especially as the number of terms in the Hamiltonian grows. To address this issue, we propose a balanced approach via partial parallelization, aiming to optimize the trade-off between computational speedup and qubit economy.

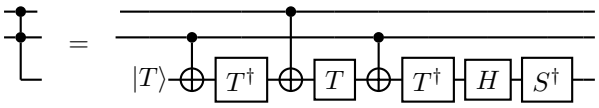
Our investigation delves into two distinct strategies for circuit organization: a serial-then-parallel approach and a parallel-then-serial approach to unary iteration. We carefully calculate the necessary resources for each, focusing on both the T depth and the auxiliary qubit count. Here, "serial" refers to the unary iteration in its conventional form, as initially presented by Babbush et al. [2]. Our findings reveal that partial parallelization not only achieves consistent speed improvements but also effectively curtails the exponential increase in auxiliary qubit requirements, presenting a viable path forward for the efficient design of quantum circuits.

Our analysis further reveals that the effectiveness of each parallelization technique in achieving speedup is influenced by several factors: N, the number of terms in the Hamiltonian, N_p a parameter representing the degree of parallelization within the circuit; and the total number of available qubits. Specifically, we observe that the serial-then-parallel approach yields greater speedup when the qubit count is low, whereas the parallel-then-serial strategy becomes more advantageous as the number of qubits increases. This relationship defines a "crossover" point that shifts with the size of N, beyond which the preferred method may oscillate between the two approaches before the parallel-then-serial method consistently emerges as the more effective strategy for larger systems. This dynamic underscores the importance of carefully selecting the parallelization technique based on the specific computational and resource context of the quantum circuit in question.

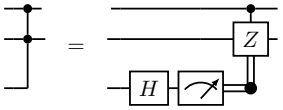
In the Methods section, we first overview the unary iteration method, and then move on to determining the resources for full circuit parallelization. Next, we will examine the partial parallelization methods: serial-then-parallel and parallel-then-serial circuit organizations mentioned in the previous paragraph. The Results section will include a discussion on T depth and speedup. Additionally, we will also examine the "crossover" behavior between the two methods, as well as emphasize the importance of choosing the optimal degree of parallelization, i.e. choosing a good N_p, needed to get the best speedup for the number of available auxiliary qubits.

2. Methods. In this study, we delve into the parallelization strategies of the unary iteration technique, a method initially introduced by Babbush et al. [2]. Unary iteration efficiently executes controlled operations based on an index register, boasting a linear T gate count of $4(N-1)$, where N represents the number of terms in the Hamiltonian [2].

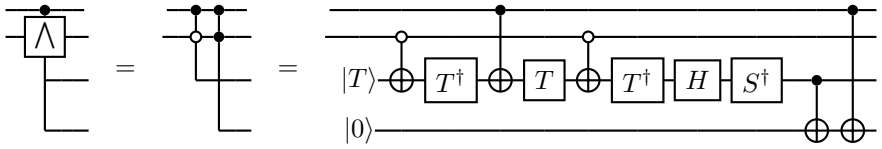
Additionally, we introduce a segment tree representation to further elucidate the structure of unary iteration. Moving forward to Subsection 2.2, we explore the implications of adopting a fully parallel approach to unary iteration, focusing on the associated resource requirements and potential speedup benefits. Subsection 2.3 is dedicated to examining two distinct strategies for partial parallelization: serial-then-parallel and parallel-then-serial. For each strategy, we calculate the requisite number of auxiliary qubits, the T depth, and the resultant speedup, with these findings systematically compiled in Table 3.1.



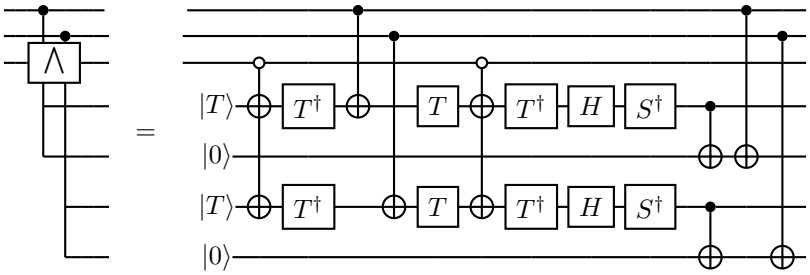
(a) AND computation with T-count/depth 4.



(b) AND uncomputation with 0 T-count.



(c) 1-to-2 AND computation with T-count and T-depth 4.



(d) 2-to-4 AND computation with T-count 8 but T-depth 4. Offset control lines link input qubits to their respective output qubits.

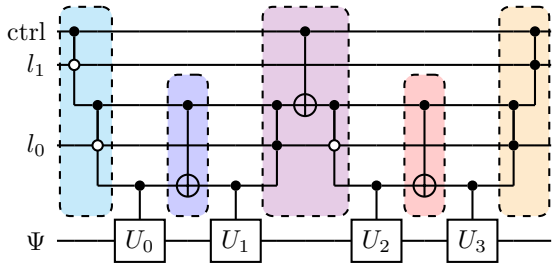
Figure 2.1: Circuits for AND computations/uncomputations. Note that T gates can be stacked in the parallel case, leading to a T-depth reduction with no additional T-count. Parallel AND uncomputation can be done by using CNOTs to uncompute the second auxiliary qubit of each pair, and then performing (b) on the remaining auxiliary qubits.

2.1. Unary Iteration. Unary iteration is a quantum algorithmic technique designed to efficiently perform controlled operations on an index register. The core idea behind unary iteration is to sequentially apply controlled operations corresponding to each term in the Hamiltonian, using a minimal number of auxiliary qubits while avoiding using any multicontrolled gates besides Toffolis. Specifically, for a Hamiltonian with N terms, unary

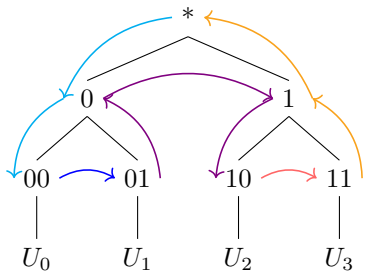
iteration achieves a linear T gate count of $4(N-1)$ and $\log_2(N)$ auxiliary qubits, making it a resource-efficient approach in terms of gate complexity and qubits [2]. In the unary iteration method, Toffolis are used to perform partial AND computations and storing these values on auxiliary qubits (one per qubit in the index register). The final auxiliary qubit is then "on" for a single state of the index register and can be used to do a singly-controlled operation (in our case, a term of the Hamiltonian) on the system register. One can then uncompute and recompute some of the auxiliary qubits to "iterate" to the next Hamiltonian term without have to redo *all* of the AND computations. This means we have to perform fewer non-Clifford Toffoli gates, resulting in a lower overall T count.

Fig. 2.1 illustrates how unary iteration is applied to controlled operations. In the context of quantum computing, the AND operation can be implemented with a Toffoli gate. In Fig. 2.1, the AND operation controlled by the top two qubits is represented as a corner shape. The AND computation, shown in Fig. 2.1a has a T count of 4 and since the operations are done sequentially, the T depth is also 4. On the other hand, the uncomputation depicted in Fig. 2.1b requires no T gates. Fig, 2.1c shows a 1-2 AND computation, and again the T count and T depth is 4. However, in the 2-4 AND computation in Fig. 2.1d, the T depth is 8, but the T depth is now halved due to the parallelization in the circuit.

To further illustrate the structure of unary iteration, we refer to a segment tree representation which has been used previously in the literature [7]. This representation provides a visual framework for understanding how the algorithm iterates through the terms of the Hamiltonian. The segment tree in Fig. 2.2b corresponds to the circuit in Fig. 2.2a. Following the color-coded arrows, we see that in serial unary iteration, the segment tree is traversed in a depth-first-search fashion. Only arrows traveling to a left child adds 4 T gates, while arrows going to the right or up the tree add no T gates. The leaves (bottom row) correspond to the final auxiliary qubit that will control our Hamiltonian terms.



(a) Serial unary iteration with $N = 4$.



(b) Representation of serial unary iteration as depth-first traversal of a segment tree.

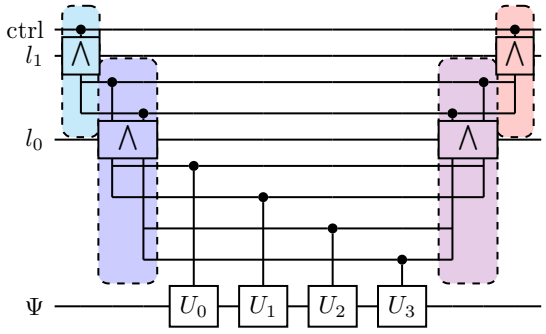
Figure 2.2: Serial unary iteration circuit and segment tree. Each arrow in a colored path corresponds to a gate in the highlighted section of matching color in the circuit, and the value of each node represents the partially computed bitstring of the $|l\rangle$ register stored on the targeted auxiliary qubit at that time in the circuit.

This visualization is also useful for resource estimation. For estimating T count, recall that only the first traversal to a child results in a Toffoli. The number of times we have to perform one of these traversals is equal to the number of internal nodes, i.e. excluding the leaves, of the tree. In a complete binary tree with N leaves, there are $N-1$ internal nodes. If we also include the cost of 4 T gates for each Toffoli from Fig. 2.1a, then we recover the

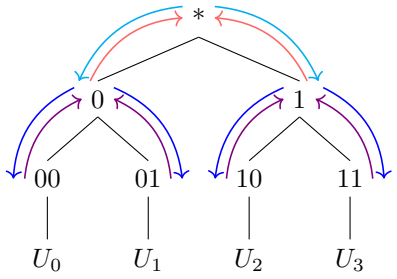
$4(N-1)$ T count. For estimating qubit resources, we see that no two nodes on the same level are ever visited at the same time. This means we only need one auxiliary qubit to store that partial AND computation per level. For a complete binary tree with N leaves, there are $\log_2(N)$ levels. We also don't need a qubit for the root node of the tree, so we recover the $\log_2(N)-1$ auxiliary qubit cost of serial unary iteration.

2.2. Fully Parallel Unary Iteration. In contrast with the depth-first-search traversal of the serial unary iteration tree, one could carry out a parallel traversal of the tree by level. This is depicted in Fig. 2.3b with the corresponding parallel unary iteration circuit shown in Fig. 2.3a. This time, the 0 and 1 branches are explored simultaneously, and the tree is traversed layer by layer. Thus we achieve logarithmic T depth: $4\log_2(N)$. Although our exponential decrease in T depth is great, we pay the price by the increase in auxiliary qubits. We now require an auxiliary qubit for every internal node. Excluding the root node, we obtain $2(N-1)$. So while parallel unary iteration offers great speedup and maintains the linear T count from [2], the number of auxiliary qubits has grown exponentially.

This motivates one to embrace a trade-off strategy in which we parallize part of the circuit to capitalize on speed gains, but also limit the number of auxiliary qubits needed. In the next subsection, we will examine two distinct strategies: serial-then-parallel and parallel-then-serial circuit organizations.



(a) Parallel unary iteration with $N = N_p = 4$.

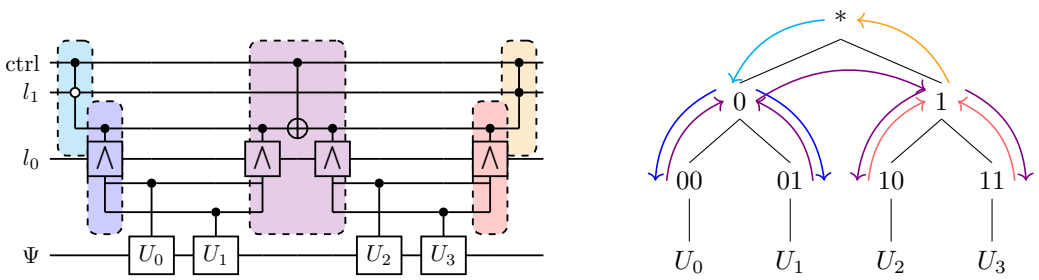


(b) Representation of parallel unary iteration as breadth-first-like traversal of a segment tree.

Figure 2.3: Fully parallel unary iteration circuit and segment tree. All arrows of the same color going to the left child will have overlapping T gates, and all arrows going to right children have no T gates.

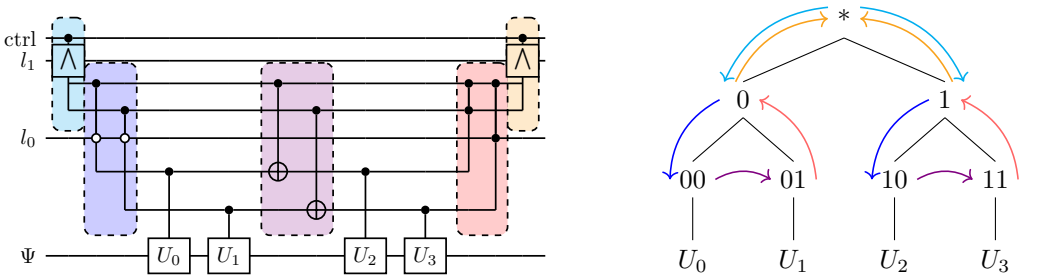
2.3. Partial Parallelization. Instead of settling for linear order T depth and logarithmic order qubits in serial unary iteration or for linear order qubits and logarithmic order T depth in parallel unary iteration, partial parallelization strategies offer a suitable balance between qubit numbers and T depth (and therefore the speedup). We consider two tree traversal orderings: serial-then-parallel (Fig. 2.4) and parallel-then-serial (Fig. 2.5).

In the case of serial-then-parallel unary iteration in Fig. 2.4b, we can imagine breaking up the segment tree into a serial subtree at the top starting with the node giving $N_s = 2$ serial-type leaves. Each of these leaves acts as roots for the parallel subtrees with $N_p = 2$ parallel-type leaves each. We can use the formulas already found for serial and parallel unary iteration T depth and qubit number and sum them up for each of the serial and parallel subtrees. We will define N_p – the number of leaves in a single parallel subtree – as



(a) Serial-then-parallel circuit with $N = 4$ and $N_p = 2$. (b) Serial-then-parallel segment tree traversal.

Figure 2.4: Serial-then-parallel unary iteration circuit and tree. Each parallel "subtree" is done serially.



(a) Parallel-then-serial circuit with $N = 4$ and $N_p = 2$. (b) Parallel-then-serial segment tree traversal.

Figure 2.5: Parallel-then-serial unary iteration circuit and tree. Each serial "subtree" is done completely in parallel.

our parallelization factor, and note that the total number of leaves is given as $N = N_s N_p$; therefore, we will often just use $N_s = N/N_p$ in our final formulas so that they are in terms of number of terms and parallelization factor.

The T depth of the serial-then-parallel cases can be computed as follows: the T depth of the serial subtree $(4(N_s - 1) = 4(N/N_p - 1))$ and N_s copies of the T depth of the parallel subtree $(4\log_2(N_p))$. Note that we need N_s copies because we are parallelizing *within* a subtree, only one parallel subtree is being done at a time. This gives us a total T depth cost of $4\left[N/N_p + N/N_p \log_2(N) - 1\right]$. Similar analysis can be done for the auxiliary qubits. We need the number of auxiliary qubits for the serial subtree $(\log_2(N_s) - 1 = \log_2(N/N_p) - 1)$ and one copy of the auxiliary qubits for a parallel subtree $(2(N_p - 1))$. The fact that we are doing one parallel subtree at a time hurt us in terms of T depth (the N_s factor above), but helps us here because we can reuse auxiliary qubits between parallel subtrees.

We carry out a similar approach for the parallel-then-serial unary iteration in Fig. 2.5. We compute T depth as follows: the T depth for the parallel subtree $(4\log_2(N_p))$ and the T depth for *one* serial subtree $(4(N_s - 1) = 4(N/N_p - 1)$. We compute the number of qubits as follows: the qubits for the parallel subtree $(2(N_p - 1))$ and N_p copies of the qubits for a serial subtree $(\log_2(N_s) - 1 = \log_2(N/N_p) - 1)$. Note that this has the reverse trend as the

serial-than-parallel case: we get the T depth of a single serial subtree because we are doing them all in parallel, but conversely we need enough auxiliary qubits for each copy to run independently.

3. Results. In this section, we explore the empirical findings that highlight the trade-offs and efficiencies achieved through the parallelization of the unary iteration subroutine in quantum circuits. A key focus is on understanding how these strategies affect T depth speedup — our critical metric for quantum computation efficiency.

Table 3.1: Unary iteration resource estimation comparison.

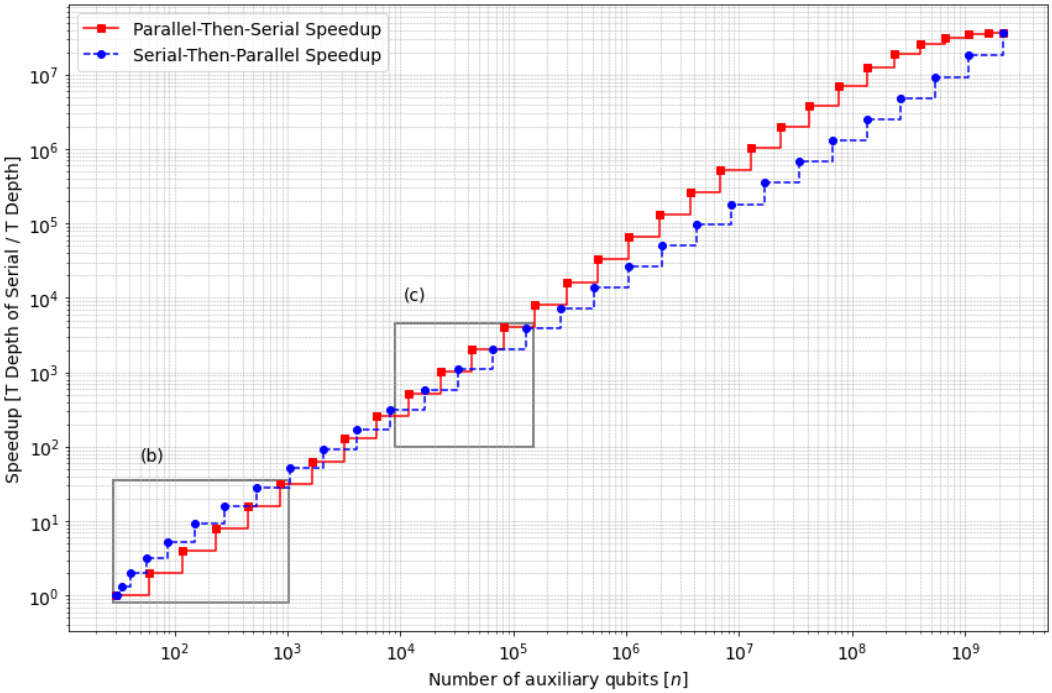
	No. Auxiliary Qubits	T Depth	Speedup ($N_p << N$)
Serial[a]	$\log_2(N)$	$4(N-1)$	–
Parallel	$2(N-1)$	$4\log_2(N)$	$\mathcal{O}\left(\frac{N}{\log_2(N)}\right)$
Serial-then-parallel	$2(N_p-1)+\log_2\left(\frac{N}{N_p}\right)$	$4\left[\frac{N}{N_p}+\frac{N}{N_p}\log_2(N_p)-1\right]$	$\mathcal{O}\left(\frac{N_p}{1+\log_2(N_p)}\right)$
Parallel-then-serial	$2(N_p-1)+N_p\log_2\left(\frac{N}{N_p}\right)$	$4\left[\frac{N}{N_p}+\log_2(N_p)-1\right]$	$\mathcal{O}(N_p)$

[a]Unary iteration method as introduced by Babbush et al. (2018) [2].

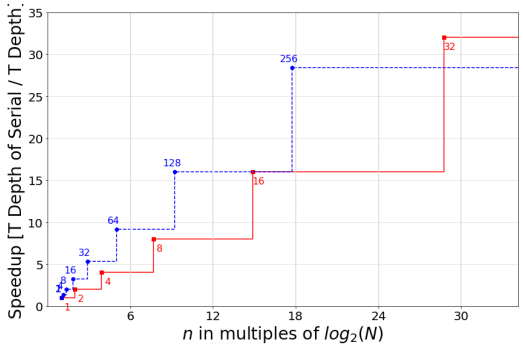
3.1. T Depth Speedup. Our initial examination of full parallelization, detailed in Table 3.1, demonstrates a significant reduction in T depth from linear to logarithmic, a transformation that greatly enhances computational speed. However, this advantage comes at the cost of increased qubit requirements, transitioning from a logarithmic to a linear scale. Specifically, while fully serial unary iteration maintains linear T depth with logarithmic auxiliary qubit needs [2], fully parallel unary iteration achieves logarithmic T depth but necessitates a linear number of qubits. According to the results in Table 3.1, the speedup achieved through full parallelization is quantified as $(N-1)/\log_2(N)$, yet the prohibitive increase in auxiliary qubits renders this approach impractical for large-scale applications.

Instead we consider a balanced approach in which the circuit benefits from the speedup gains of partial parallelization while constraining the expansion of auxiliary qubit requirements. We analyze two distinct configurations: serial-then-parallel and parallel-then serial circuits, with their respective T depth and qubit numbers summarized in Table 3.1. The speedup offered by serial-then-parallel unary iteration over fully serial is calculated as $(N-1)/\left[N/N_p + N/N_p\log_2(N_p) - 1\right]$, while the parallel-then-serial speedup is $(N-1)/\left[N/N_p + \log_2(N_p) - 1\right]$. The presence of the N/N_p factor multiplied on the logarithmic term in the denominator of the serial-then-parallel speedup, might lead one to naively conclude that the parallel-then-serial approach is optimal for all scenarios. However, as we will make clear in the subsequent subsection, the optimal strategy is contingent upon the interplay between the number of available qubits n, the number of terms in the Hamiltonian N, and the degree of parallelization N_p, revealing a more complex decision landscape than is initially apparent.

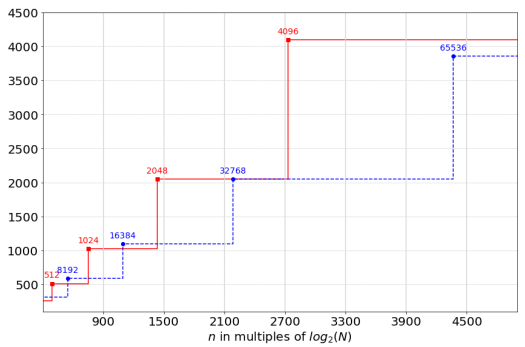
3.2. Crossover. The formulas from Table 3.1 may lead one to erroneously assume that parallel-then-serial unary iteration invariably offers the best trade-off between speedup and qubit number. A careful study of Fig. 3.1 tells a different story. The main Fig. 3.1a depicts the speedup for a system size of $N = 2^{30}$ for increasing numbers of auxiliary qubits. This analysis, depicted through the dashed blue line with circle markers for the serial-then-parallel



(a) Speedups for parallel-then-serial (red) and serial-then-parallel (blue) unary iteration. Fully serial ($N_p = 1$) is the bottom-left point, while fully parallel ($N_p = N$) is the top-right point.



(b) Inset focusing on low numbers of auxiliary qubits, where serial-then-parallel outperforms parallel-then-serial.



(c) Inset focusing on the crossover region between serial-then-parallel and parallel-then-serial.

Figure 3.1: Speedup as a function of auxiliary qubits for parallel unary iteration for a system size of $N = 2^{30} \approx 10^9$. Insets label points with their corresponding N_p values and show auxiliary qubits in terms of multiples of $\log_2(N)$, i.e. how many copies of the LCU coefficient register would be needed.

method and the solid red line with square markers for the parallel-then-serial method, considers escalating degrees of parallelization (N_p). Contrary to initial expectations, this visual comparison reveals that the optimal strategy for maximizing speedup while efficiently managing qubit resources is not so straightforward.

On the lower left portion of the plot, highlighted in the inset Fig. 3.1b the serial-then-parallel technique initially demonstrates a higher speedup. Towards the middle portion of the plot, detailed in inset Fig. 3.1c, a pivatol shift occurs. Between 10^4 and 10^5 qubits, we observe the parallel-then-serial method overtake serial-then-parallel in terms of speedup, marking a significant crossover in efficiency between the two strategies. Beyond this point, the plot reveals a pattern of oscillation where the lead in speedup advantage shifts back and forth between the two methods, until reaching approximately 10^5 qubits. Beyond this threshold, parallel-then-serial unary iteration consistently maintains an edge over the serial-then-parallel approach.

What underlies the observed crossover and oscillation phenomena between the two partial parallelization methods? The answer lies within the detailed annotations of the N_p values in the inset Fig. 3.1b and Fig. 3.1c. Intriguingly, these annotations reveal that for a given number of qubits, the N_p values associated with the serial-then-parallel method consistently exceed those of the parallel-then-serial method. The discrepancy between N_p values accounts for the observed crossover behavior and highlights the nuanced balance between qubit availability and the amount of parallelization in the circuit, offering deeper insights into the strategic deployment of quantum resources for optimizing circuit performance.

The complex dynamics of the crossover behavior are further illustrated in Fig. 3.2, which marks the initial point at which the parallel-then-serial method surpasses the serial-then-parallel method across varying system sizes (N). This point is annotated for each method, with the serial-then-parallel approach represented by a blue dashed line and circle marker, and the parallel-then-serial method by a red solid line and square marker. Notably, these annotations include the corresponding N_p values at the crossover point for both strategies.

When visualized on a $\log_2$-scale, a clear pattern emerges: the serial-then-parallel unary iteration consistently necessitates larger N_p values to achieve optimal speedup for the same number of qubits. Moreover, as N increases, the disparity in the required degree of parallelization (N_p) at the crossover point becomes more pronounced. This observation is not merely a quantitative detail but a qualitative insight into how the effectiveness of parallelization strategies scales with system size. This analysis reveals a critical consideration for the strategic deployment of quantum resources in optimizing circuit performance.

Contrary to what was initially anticipated by the formulas in Table 3.1, the serial-then-parallel unary iteration strategy outperforms its counterpart at lower auxiliary qubit count. This counterintuitive finding is captured in Fig. 3.1 where the performance advantage of the serial-then-parallel approach becomes apparent for lower qubit count. The underlying reason for this unexpected outcome is traced back to the disparities in N_p values between the two methods, as detailed in the inset Figs. 3.1b and 3.1c. Further, Fig. 3.2 exemplifies that this crossover behavior stems from the larger N_p values required for the serial-then-parallel method and that it is system size-dependent. This insight not only unveils the dynamics between these partial parallelization strategies but also underscores the importance of considering the degree of parallelization in optimizing quantum circuit designs for efficiency.

4. Discussion/Conclusion. In this study, we explored the use of various strategies for parallelizing unary iteration as initially described in Babbush et al. [2]. Our investigation found that full parallelization offers significant speedup over serial unary iteration - transitioning from linear T depth to logarithmic, but at the cost of transitioning from logarithmic order auxiliary qubits to linear order. We discovered that partial parallelization methods:

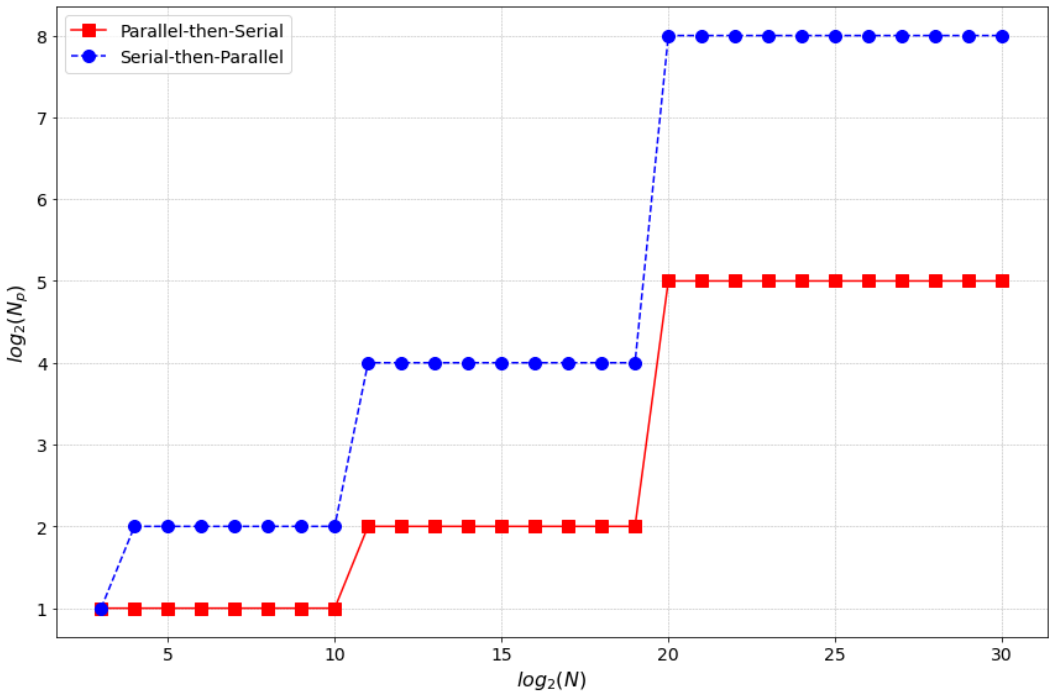


Figure 3.2: Values of N_p where parallel-then-serial (red) first outperforms serial-then-parallel (blue) for various system sizes. Note that the values of N_p get larger, i.e. serial-then-parallel is favorable for longer and the crossover point shifts to the right on plots like Fig. 3.1a, as N increases.

serial-then-parallel and parallel-then-serial offer a suitable trade-off between T depth and qubit costs as is quantified in Table 3.1.

While the T depth formulas in Table 3.1 mght suggest the parallel-then-serial method as universally superior, the speedup outcomes prove to be more complex as is apparent from Fig. 3.1. The serial-then-parallel approach offers better speedup for low qubit count while parallel-then-serial unary iteration is best for higher qubit numbers. This can be traced back to the disparity between the N_p values of the two methods. The serial-then-parallel approach consistently requires larger N_p values for the same number of qubits. At some threshold point which depends on the system's size, the parallel-then-serial approach first overtakes the serial-then-parallel method as can be seen from Fig. 3.2. Beyond this point the optimal speedup shifts back and forth between the two methods until the parallel-then-serial approach eventually triumphs for larger qubit numbers. Thus, the N_p value is a key factor which must be considered when assessing resource requirements in the quantum circuit.

For future work, we will investigate additional avenues for developing efficient quantum circuits. This includes optimizing the organization, routing, and scheduling of the auxiliary qubits similar to the resource estimates in Lee et al. [8]. We are also interested in making use of dirty [12] and conditionally clean qubits [7, 13] to realize additional circuit efficiency. In its current form, unary iteration makes use of "clean" auxiliary qubits meaning that they have been initialized to a known state. "Dirty" qubits are those which are initialized in unknown states which must be restored upon completion of the computation [7]. To

summarize Khattar and Gidney [7], a "conditionally clean" qubit on the other hand, is one whose state is either dirty or clean conditioned by a subset of other system qubits.

In conclusion, this study contributes to the ongoing discourse on optimizing quantum circuits in terms of physical resources and speedup. Our findings highlight the intricate balance required in the parallelization of unary iteration. As this field continues to evolve, this study will undoubtedly contribute to the refinement of strategies aimed at harnessing the full potential of quantum computing.

References.

[1] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, *A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 32 (2013), pp. 818–830.

[2] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, *Encoding electronic spectra in quantum circuits with linear t complexity*, Phys. Rev. X, 8 (2018), p. 041015.

[3] A. M. Childs and N. Wiebe, *Hamiltonian simulation using linear combinations of unitary operations*, Quantum Info. Comput., 12 (2012), p. 901–924.

[4] R. Feynman, *Quantum mechanical computers*, Foundations of Physics, 16 (1986), pp. 507–531.

[5] R. P. Feynman, *Simulating physics with computers*, International Journal of Theoretical Physics, 21 (1982).

[6] C. Jones, *Low-overhead constructions for the fault-tolerant toffoli gate*, Phys. Rev. A, 87 (2013), p. 022328.

[7] T. Khattar and C. Gidney, *Rise of conditionally clean ancillae for optimizing quantum circuits*, 2024.

[8] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, *Even more efficient quantum computations of chemistry through tensor hypercontraction*, PRX Quantum, 2 (2021), p. 030305.

[9] D. Litinski, *Magic State Distillation: Not as Costly as You Think*, Quantum, 3 (2019), p. 205.

[10] S. Lloyd, *Universal quantum simulators*, Science, 273 (1996), pp. 1073–1078.

[11] G. H. Low and I. L. Chuang, *Hamiltonian Simulation by Qubitization*, Quantum, 3 (2019), p. 163.

[12] G. H. Low, V. Kliuchnikov, and L. Schaeffer, *Trading T gates for dirty qubits in state preparation and unitary synthesis*, Quantum, 8 (2024), p. 1375.

[13] J. Nie, W. Zi, and X. Sun, *Quantum circuit for multi-qubit toffoli gate with optimal resource*, 2024.

[14] M. A. Nielsen and I. L. Chuang, *Quantum Computing and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 2011.

[15] F. Orts, E. Filatovas, G. Ortega, J. F. SanJuan-Estrada, and E. M. Garzón, *Improving the number of t gates and their spread in integer multipliers on quantum computing*, Phys. Rev. A, 107 (2023), p. 042621.

[16] H. Thapliyal, T. S. S. Varun, E. Munoz-Coreas, K. A. Britt, and T. S. Humble, *Quantum circuit designs of integer division optimizing t-count and t-depth*, in 2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS), 2017, pp. 123–128.

STORAGE SYSTEM CHARACTERIZATION IN VIRTUALIZED TESTBED

JOHN SHAWGER* AND MATTHEW L. CURRY†

Abstract. Emerging workloads have broadened the requirements for data I/O systems. Prior work has focused on the characterization of workloads and their performance on production systems. We develop a set of tools for characterizing the behavior of storage systems in a virtual testbed environment, and compare the behavior of a parallel file system, BeeGFS, and an object store, Minio, on a synthetic deep learning workload.

1. Introduction. Data I/O is a challenge for large-scale systems. Researchers have long seen the need to properly characterize application I/O demands and develop realistic benchmarking tools to evaluate I/O performance. As a result, storage systems for highly parallel high performance clusters have evolved over the past several decades in response to application needs. Parallel file systems are understood by the HPC systems community to have good support for typical application I/O patterns such as checkpoint and restart.

Within the past decade, a new class of data-intensive applications has emerged. In particular, data analytics and deep learning applications have become important in many scientific domains and are recognized as a growing class of workloads in HPC systems [26]. Over the same time period, cloud-based [17] storage offerings have expanded to include object stores [22, 35] and key-value stores [29, 30]. These storage systems have simple interfaces and do not support all of the guarantees of the POSIX file interface, but application developers have found them attractive [8, 13, 33] for data intensive and ML applications.

Recent HPC storage research has attempted to characterize the I/O patterns of deep learning workloads [10, 27]. Workload characterization studies have yielded valuable insights into the I/O usage of HPC applications on existing flagship clusters and storage systems. However, any performance evaluation performed on existing systems is colored by the bespoke nature of each system – clusters have unique hardware, networking, and scale. Work of this nature does not answer the question of whether an application would perform better on a different type of storage system.

Using archetypal workloads identified in characterization studies, we propose characterizing storage system daemons. We consider a daemon running on a storage node to be an agent that processes messages sent by clients and generates actions in the form of system calls which perform I/O on local storage. By measuring the input messages and output actions of the daemon, we can gain understanding of the workings of the storage system. This type of measurement is unaffected by optimizations which do not fundamentally change the nature of the storage system, such as RDMA [37, 36], accelerator offload [18, 20], or increased asynchrony [19]. We approach our work with several research questions.

- Are certain workloads fundamentally suitable or unsuitable for storage system types?
- Can we infer storage system policies by measuring the behavior of storage daemons?
- Are there properties of storage systems which are not discernable solely by looking at the inputs and outputs of their daemons?

In this paper, we create a virtual machine testbed and use it to compare BeeGFS, a parallel file system, and Minio, an S3-compatible object store. For each system, we wrote an instrumentation framework using eBPF to log the activity generated by the storage system on each member node. Finally we compare the behavior of the two systems while running a synthetic deep learning workload.

*University of Wisconsin-Madison, shawgerj@cs.wisc.edu
†Sandia National Labs, mlcurry@sandia.gov

2. Background.

2.1. Parallel File Systems. Parallel file systems stripe data across collections of storage nodes – the number of storage targets for a file is known as the *stripe width*, the amount of data written in each chunk is known as the *block size*. By striping data across multiple targets, the file system uses the aggregate bandwidth of the storage targets for reads and writes. Parallel file systems generally provide POSIX guarantees, such as strict consistency amongst processes interacting with files. To ensure global consistency, parallel file systems have a centralized metadata service, which controls placement of data and write ordering. Clients of the file system communicate directly with the storage nodes for the data path. Storage nodes run user-level daemons which store chunk data atop a local file system, commonly XFS ldiskfs, a fork of ext4.

2.2. Object Stores. Object stores organize data into buckets, where each bucket is a flat namespace for objects stored in it. Objects are addressed via their bucket name and a unique identifier. While complex POSIX-style file system may be implemented on top of an object store [35], object stores have a much simpler interface (GET and PUT) for accessing an objects. Commonly, objects are accessed atomically, although a GET command may take a byte offset to support reading a partial object. The absence of a strongly consistent POSIX model alleviates the need for a standalone metadata service. Due to their simplicity, object stores have become widely used for data-intensive workloads where data can easily be mapped to objects, just as training a deep learning model on an image dataset.

2.3. HPC Workload Characterization. Designers of I/O systems have a strong interest in understanding the workloads running on top of them. Traditionally [31], HPC applications where believed to be write-intensive, perform infrequent random I/O, and perform N-N I/O using the POSIX interface. More recent work [9] has used the Darshan tool to profile applications directly, and has found that read-intensive workloads are more common than previously thought. Recent years have seen an explosion of activity in training deep learning models, which requires reading a dataset over a number of epochs. Tensorflow [2], a popular machine learning library, has been characterized [10] while running the ImageNet training workload. While reads are random over the entire dataset, Tensorflow reads sample files within the dataset sequentially. We believe the demands being placed on I/O systems have evolved substantially in recent years, and storage systems should be evaluated in light of these new workloads.

3. Experimental Setup.

Distributed storage systems are complex pieces of software with many system requirements necessary for their deployment. They may require unique build environments, direct access to storage devices, kernel modules, specialized dependencies, and other environmental peculiarities. Creating the right environment on production HPC systems is a difficult problem, as it is important that the environment built for the storage system not interfere with the HPC system environment. Furthermore, the storage system may require privileged access to install and run, that may not be feasible to grant for exploratory research.

We have developed a method for deploying storage systems for research purposes which avoids these difficulties. Sandia has developed a set of tools for large-scale cyber experimentation, known within Sandia as Emulytics [15] (a portmanteau of emulation and analytics). Emulytics tools support emulating large-scale networks using virtual machines (VMs) and software-defined network topologies. In particular, we use the Carnac [16] cluster, which provides bare-metal-as-a-service for Emulytics workloads. Carnac nodes run a minimal instance of Ubuntu 18.04 and have ample memory (512GB) for VMs. The Minimega [24] project is a Sandia tool which can orchestrate VMs, providing an API to start and stop

individual VMs and run executables within them. Minimega balances VM allocation over a set of physical nodes. Lastly, the Firewheel system manages the experimental runtime as a set of Python scripts, using Minimega to control VMs. Our experiments are defined as a set of resources scheduled and executed by Firewheel – (1) VM images, (2) software source and packages, (3) package configuration and runtime scripts, (4) workload runtime scripts, and (5) the Firewheel configuration script, which defines the experiment topology and schedules the execution of all resources within the experiment. Firewheel can push resources to running VMs, run arbitrary executables, and pull data from VMs at the conclusion of a workload.

Here is a list of requirements for our experimental setup and a short description of how Carnac and Firewheel satisfy each requirement.

1. *Administrative flexibility.* We launch custom VMs with Firewheel and have administrative access to the VMs once they are booted. Firewheel can also create complex network topologies, although for our experiment, a single switch with no network delay has been sufficient.
2. *Scale experiment up and down easily.* Firewheel orchestrates VMs across a set of physical nodes. Creating more or fewer VMs is trivial within Firewheel.
3. *No source code modification.* Storage system codebases are large and complex, and we sought to avoid direct modification of system code to introduce instrumentation. Root privileges on Firewheel VMs allow us to use eBPF uprobes, tracepoints, and load custom libraries with `LD_PRELOAD` to instrument our target storage systems.
4. *Automated and repeatable experimentation.* We define our entire experiment, including VM setup and configuration, using Firewheel scripts, which are written in Python. Experiments are thus easily run and repeated.

While the Emulytics platform satisfies our administrative goals for our experiments, one lingering question is whether or not the virtualized environment in which we run our experiments meaningfully alters the results compared to a bare-metal approach. In this work, we are interested in relative access patterns of data in a parallel file system and object store, not the absolute performance of each system. While a thorough performance evaluation of each system would necessitate quantifying the overhead, if any, of the virtualized environment, the traces of data accesses on each storage node remain unchanged.

3.1. Firewheel-Beegfs. A BeeGFS installation consists of a set of storage nodes, a manager node, and one or more client nodes. Using Firewheel, we launch a separate Ubuntu 22.04 VM for each node, and connect them using a software-defined switch. Our BeeGFS experiment is shown in Figure 3.1. The manager node is given IP `10.0.0.1`, clients are given IPs `10.0.0.2` through `10.0.0.9`, and storage nodes are given IPs `10.0.0.10` and higher. We can launch up to nine client nodes and an arbitrary number of storage nodes. For ease of setup and teardown, we give each storage node 128GB of memory and create a 96GB ramdisk which is assigned to the storage daemon. We can scale up the capacity of BeeGFS by creating more storage nodes – the only limitation here is the amount of physical RAM available to Firewheel. Carnac nodes have 512GB of memory each, thus we can comfortably scale BeeGFS by 288GB of storage per physical node.

Client nodes are equipped with the BeeGFS client kernel module, which is used to mount the BeeGFS file system. We use BeeGFS 7.4.3. We have configured OpenMPI on the clients in order to run our benchmark programs, IOR 4.0 [31] and DLIO [12].

3.1.1. eBPF and uprobes. Recall that one of our goals in instrumenting storage services was to avoid direct source code modification of the service. We use eBPF [14] and uprobes [11] to interpose methods in the BeeGFS storage daemon which handle incoming messages. Our library, which is written in Rust and uses aya-rs [6], has three components

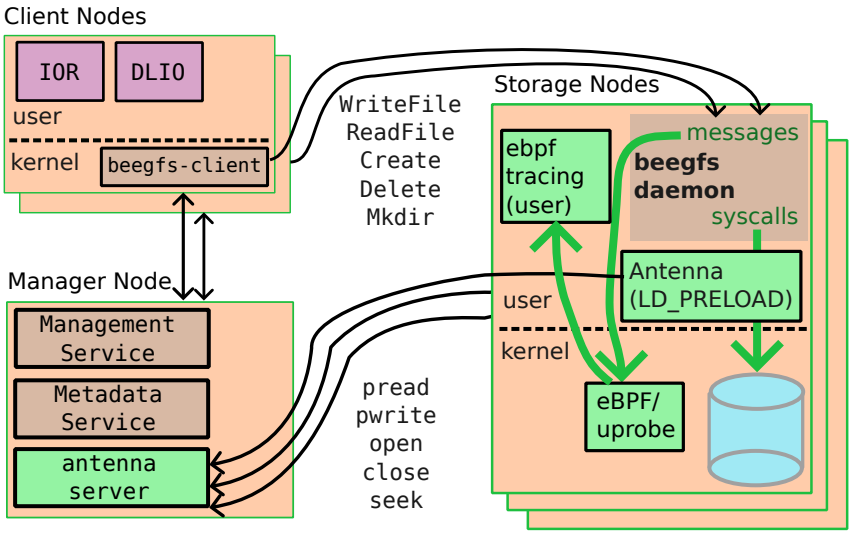


FIG. 3.1. *Firewheel-BeeGFS. Purple denotes user-level benchmark programs, yellow denotes S3-compatibility layers, gray denotes BeeGFS services, green denotes instrumentation services.*

– (1) eBPF programs which are attached to uprobes and run in the kernel, (2) a user-space program which logs data captured by eBPF, and (3) a library defining common data structures used by both the kernel and user-space components.

Uprobes are a tracing capability included in the Linux kernel. Given an offset within an executable file, the kernel creates a special copy of the page containing the offset. At the offset, the kernel inserts a breakpoint instruction. When the breakpoint is reached, the uprobe handler within the kernel launches the eBPF program associated with the offset. As eBPF allows arbitrary code to run within the kernel, eBPF programs are subject to many constraints and must undergo static verification [25] prior to execution. In particular, eBPF programs are limited to a 512 byte stack. In order to manipulate larger amounts of data, eBPF provides different types of maps, which are mmap-ed regions of memory which can be accessed within the kernel or in userspace.

In particular, we use uprobes to trace incoming messages to each storage daemon. Tracing at the application level allows us to introspect the exact file name, offset, and length of reads and writes requested from storage daemons, but we must know precisely where this data is stored in memory in the instrumented function in order to read it with the uprobe. In BeeGFS, each message type has an associated C++ class. We ran BeeGFS under `gdb` separately to record the offsets of data fields within the class data hierarchy using the `ptype /o` command. eBPF programs have access to probed function's arguments, because those arguments are placed on the execution stack. In C++ programs, class methods have an implicit first argument, a pointer to the instance of the class known as the `this` pointer. Our eBPF programs access data members for each class by following the `this` pointer for traced methods. Given the data gathered from `gdb`, we were able to access all the relevant class data for tracing storage requests in BeeGFS.

3.1.2. Antenna. Inspired by Darshan [32], we wrote a library to intercept system calls being made by the storage daemon. The BeeGFS storage daemon makes system calls, as do most C/C++ programs compiled in a Linux environment, by calling wrapper functions in libc. We use `LD_PRELOAD` to wrap the libc functions with our own functions which capture relevant data about the system call being made. We focus our library on system calls which

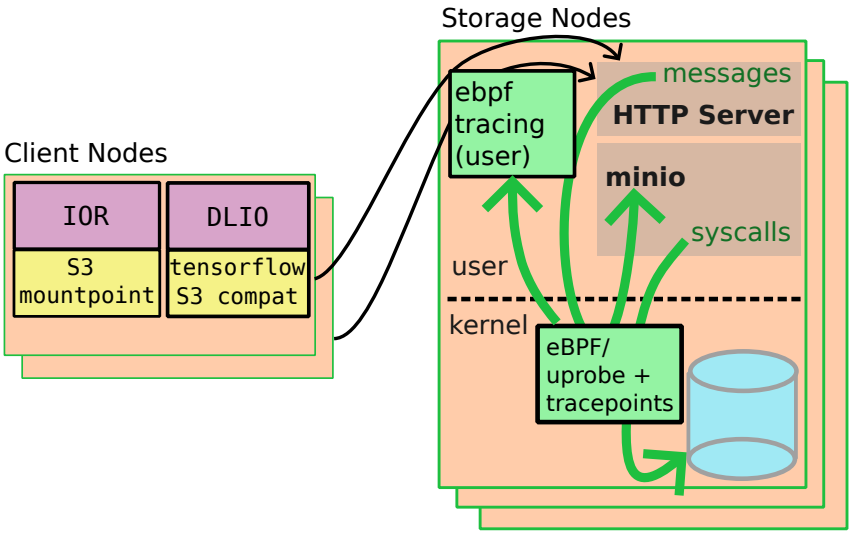


FIG. 3.2. *Firewheel-Minio. Purple denotes user-level benchmark programs, gray denotes Minio services, green denotes instrumentation services.*

interact with the file system – `open`, `close`, `read`, `write`, `pread`, `pwrite`, and `seek`. Using `strace` [1], we verified that the BeeGFS storage daemon uses these system calls for its file system operations. For each system call, we capture it's arguments and record the time at which it was called. We then send that data via UDP to the `antenna-server` process running on the manager node. UDP was chosen to improve the scalability of the Antenna, as it could be used for large-scale experiments in the future. Sending messages over the network to a long-running server process avoids a problem we observed when trying to log the messages locally. When logging locally, we found that output buffering was causing incomplete log output – the BeeGFS storage daemon process would end prior to all the output being flushed from the buffer, causing a loss in logging data. We have not observed the same problem after logging the messages remotely.

3.2. Firewheel-Minio. Minio is an S3-compatible object store. Clients communicate with Minio nodes using HTTP, as shown in Figure 3.2.

3.2.1. eBPF tracing HTTP REST. Clients communicate with Minio via HTTP, using an S3-compatible REST protocol. For example, a client might check for the existance of an object using `HEAD`, upload an object to the object store using `PUT`, and download an object using `GET`, with each command followed by a uniform resource locator (URI), which corresponds to the bucket and object name.

Because HTTP requests are sent via the network, we can trace them within the kernel by monitoring sockets opened by Minio rather than probing Minio code directly. We used `strace` to verify that Minio opens sockets using `accept4`, which returns a file descriptor. All data read and written to the file descriptor between `accept4` and `close` are HTTP messages. To capture HTTP data read and written on the socket, we use an eBPF tracepoint to record data on file descriptors opened by `accept4`. When data is read from the socket, we send a small amount (1KB) of the data to userspace via a circular buffer. A process in userspace polls the buffer and parses the HTTP headers within the data read from the socket. The headers of each HTTP request contain all the necessary information for tracing S3 storage requests (request type, URI, content-length).

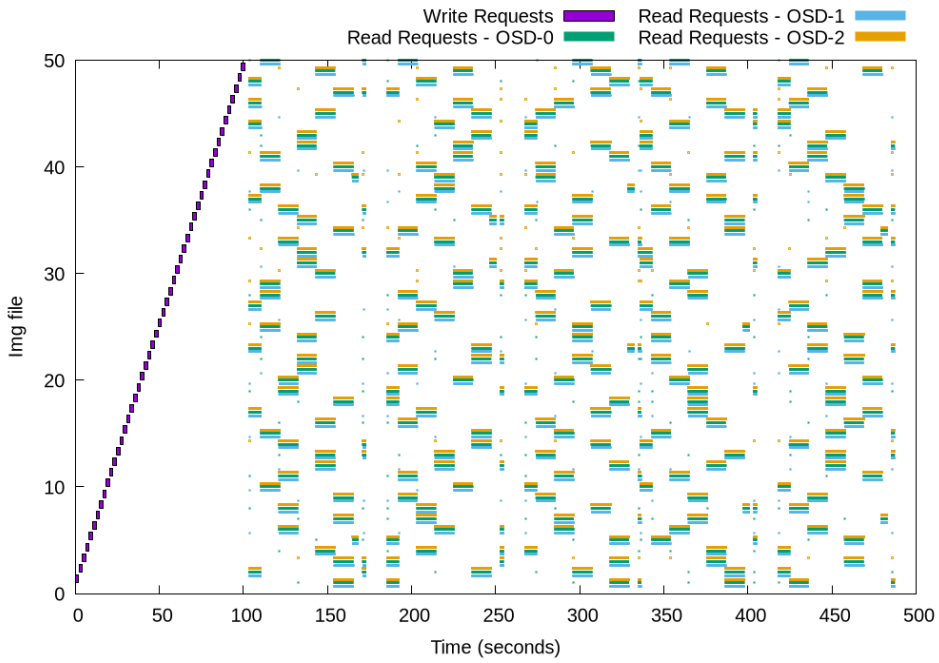


FIG. 4.1. *DLIO workload running on Beegfs. BeeGFS client requests.*

3.2.2. eBPF tracepoints. Minio is written in Go, and discovered that Go applications running on Linux do not follow the typically convention of linking to libc and using libc system call wrappers. Since our Antenna framework relies on `LD_PRELOAD` to interpose libc functions, it unfortunately does not work with Minio. Therefore, we must instrument system calls inside the kernel. To do this, we use the tracepoint functionality within the Linux kernel. There are tracepoints defined for each system call enter and exit. Using eBPF, we record the arguments passed to each system call we are interested in tracing on enter, and the return value from each system call on exit. Similarly to the data intercepted on sockets, we pass the data back to userspace for logging.

4. Experimental Results.

4.1. ML Training. We use the DLIO benchmark to simulate a machine learning training workload. We run DLIO with the resnet50 tensorflow workload, which generates synthetic data similar to an imagenet dataset, and then "runs" the workload over a course of five epochs. DLIO simply sets a sleep timer for the training phase to simulate computation. We generate 50 data files, each 137MB large, for a total size of 6.8GB. The DLIO client runs as a single process with eight threads. Tensorflow has both a POSIX interface and an S3 interface. We use these interfaces unmodified – there is no extra connector used for S3.

4.1.1. Beegfs. Using our Ebpf-uprobe tracing system, we trace the requests made from the DLIO client process to BeeGFS storage nodes (OSDs) during the DLIO resnet50 workload. Figure 4.1 shows the results. It is clear from the graph that there are two distinct phases in the workload - a write (data generation) phase, and a read (training) phase. The resnet50 workload generates 50 sample files – we plot each file by it's number on the y-axis of the graph. Reads to different OSDs are slightly vertically offset so that parallel reads to different OSDs are visible on the graph. All reads appear to be done in parallel from the

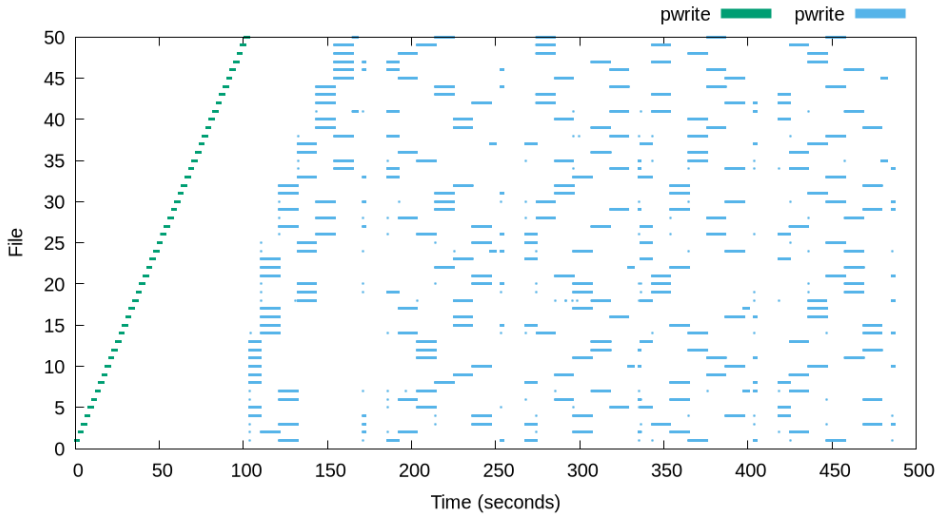


FIG. 4.2. *DLIO workload running on Beegfs. BeeGFS write and read system calls on OSD-1.*

Total Data Written	OSD-0	OSD-1	OSD-2
multipart	38MB	141MB	190MB
tmp	2.9GB	1.6GB	1.8GB
resnet50	2.8GB	2.4GB	3GB

FIG. 4.3. *Total data written to Minio OSDs, by file type*

three OSDs, and eight files are read concurrently, which corresponds to the eight threads in the DLIO client process.

We plot the system calls generated on a single BeeGFS OSD in Figure 4.2. The system calls match closely with the requests from the client, and are similar on each OSD.

4.1.2. Minio. Figure 4.4 shows the write phase of our DLIO workload on the Minio object store. We find that Minio writes to three types of files – "multipart" files, "tmp" files, and "resnet50" files. Summing the total amount of data written to each type of file gives some clues as to how they are used. Comparatively little data is written to "multipart" files, so we hypothesize these files are used for internal metadata relating to the objects and their placement. Large amounts of data corresponding to the size of the dataset are written to both "tmp" and "resnet50" files, and the read phase (Figure 4.5) operates exclusively on "resnet50" files.

In Figure 4.4, we plot write operations occurring on each type of file over time. There is an initial phase of writes, when only "multipart" and "tmp" files are written to (the activity is similar on each OSD; we only plot the writes for OSD-0). Surprisingly, writes to "resnet50" happen during the read phase of the workload, following the initial phase in which all workload data is written to "tmp" files. More study of Minio's internals is needed to understand this behavior, but the presence of heavy write activity during the read phase is unexpected and shows the utility of characterizing storage daemons when seeking to understanding storage system behavior.

Figure 4.5 shows the read (training) phase of the DLIO workload running on Minio. While all OSDs are utilized during the read phase, even without using a load balancer, this graph shows a different access pattern than the same workload on BeeGFS. The client

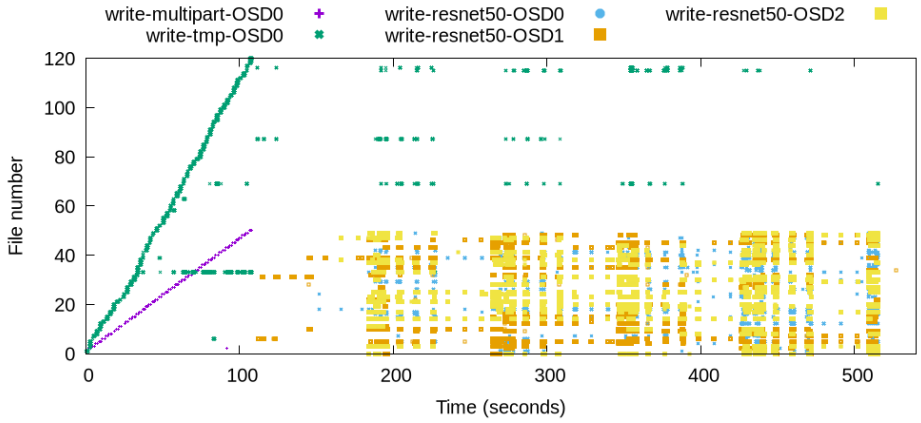


FIG. 4.4. *DLIO workload running on Minio. Write system calls over time separated by file type.*

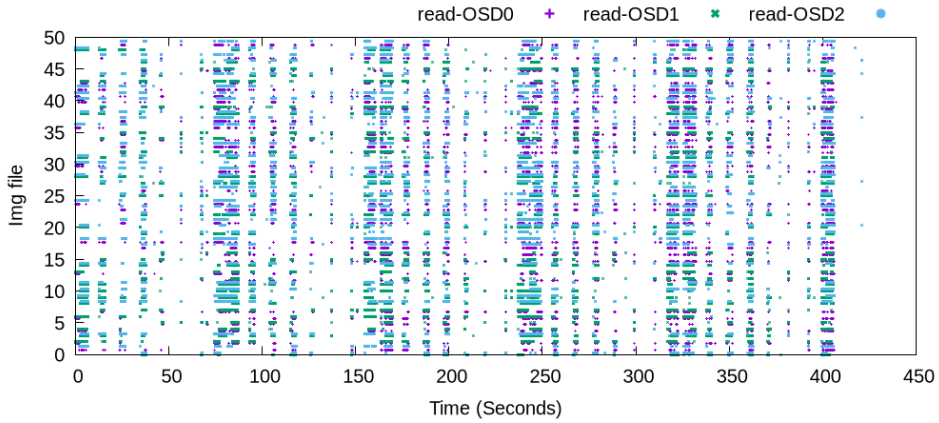


FIG. 4.5. *DLIO workload running on Minio. Read system calls over time on resnet50 image files.*

accesses resnet50 files from OSDs, but reads each file from a single OSD, rather than in parallel from several OSDs.

5. Related Work. Storage systems are notoriously difficult to evaluate at scale. The David emulator [4] makes the observation that data blocks may be compressed out of a storage workload within losing information about the storage system access pattern. Exalt [34] compresses away application data and was used to find bugs in the HDFS metadata server at large scale. We use an Emulytics platform to evaluate our target systems at scale.

An abundance of prior work has utilized `strace` to gain information about program correctness [28] and record traces for performance and stress testing [3, 7]. Our work uses `strace` and eBPF to trace read/write patterns. We analyze the traces to correlate storage daemon inputs with system call outputs, but we do not use the traces to evaluate raw performance.

Application tracing has long been an active field of research. Two earlier systems, TraceFS [5] and TRACE [23] have been used to trace applications, and make different tradeoffs in performance overhead vs. the amount of tracing data collected. More recently, Darshan [32] has been widely [9, 10, 21] used to characterize application I/O access patterns.

Darshan instruments applications and generates traces of POSIX, HDF5, and MPI-IO activity. Our Antenna system generates traces of POSIX system calls, but we use it to instrument storage daemons rather than applications.

6. Conclusion. High performance I/O requirements have diversified in recent years. It is important to continually evalute the design space of I/O systems with an eye on emerging workloads, such as deep learning. We have developed a testbed and instrumentation framework which runs BeeGFS, a parallel file system, and Minio, an object store. Our choices in developing a set of instrumentation tools reflect a desire to balance general approaches with the need to capture application-specific information. By instrumenting at the source-code level with eBPF, we can capture information with a high degree of fidelity compared to more generic approaches at the cost of higher development complexity.

Examining the data from our experiments reveals differences between the two systems. We observe that BeeGFS uses storage nodes in parallel for reads and writes, while Minio does not, although separate client threads may read from more than one storage node concurrently. Instrumentation can reveal system policies which are not clearly documented – for example, we find that Minio copies data from temporary files during read operations, possible as part of read compaction operation. We are optimistic that further work could provide insight into other system policies such as write coalescing and caching behaviors.

REFERENCES

[1] *strace*. `http://strace.io`. Accessed: Aug 28, 2024.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., *Tensorflow: a system for large-scale machine learning*, in 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016, pp. 265–283.

[3] N. Agrawal, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, *Generating realistic impressions for file-system benchmarking*, ACM Transactions on Storage (TOS), 5 (2009), pp. 1–30.

[4] N. Agrawal, L. Arulraj, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, *Emulating goliath storage systems with david*, ACM Transactions on Storage (TOS), 7 (2012), pp. 1–21.

[5] A. Aranya, C. P. Wright, and E. Zadok, *Tracefs: A file system to trace them all.*, in FAST, 2004, pp. 129–145.

[6] *aya-rs*. `https://github.com/aya-rs/aya`, 2024.

[7] E. F. Boza, C. San-Lucas, C. L. Abad, and J. A. Viteri, *Benchmarking key-value stores via trace replay*, in 2017 IEEE International Conference on Cloud Engineering (IC2E), IEEE, 2017, pp. 183–189.

[8] Z. Cao, S. Dong, S. Vemuri, and D. H. Du, *Characterizing, modeling, and benchmarking rocksdb key-value workloads at facebook*, in 18th USENIX Conference on File and Storage Technologies (FAST 20), 2020, pp. 209–223.

[9] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross, *Understanding and improving computational science storage access through continuous characterization*, ACM Transactions on Storage (TOS), 7 (2011), pp. 1–26.

[10] S. W. Chien, A. Podobas, I. B. Peng, and S. Markidis, *tf-darshan: Understanding fine-grained i/o performance in machine learning workloads*, in 2020 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, 2020, pp. 359–370.

[11] J. Corbet, *Uprobes in 3.5*. `https://lwn.net/Articles/499190/`, May 2012.

[12] H. Devarajan, H. Zheng, A. Kougkas, X.-H. Sun, and V. Vishwanath, *Dlio: A data-centric benchmark for scientific deep learning applications*, in 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), IEEE, 2021, pp. 81–91.

[13] S. Dong, A. Kryczka, Y. Jin, and M. Stumm, *Rocksdb: Evolution of development priorities in a key-value store serving large-scale applications*, ACM Transactions on Storage (TOS), 17 (2021), pp. 1–32.

[14] *ebpf*. `http://ebpf.io`.

[15] *Sandia emulytics*. `https://www.sandia.gov/emulytics`. Accessed: Aug 28, 2024.

[16] J. F. Floren, J. A. Friesen, C. D. Ulmer, and S. T. Jones, *A reference architecture for emulyticstm clusters*, tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2017.

[17] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, et al., *Cloud programming simplified: A berkeley view on serverless computing*, arXiv preprint arXiv:1902.03383, (2019).

[18] J. Kim, I. Jang, W. Reda, J. Im, M. Canini, D. Kostić, Y. Kwon, S. Peter, and E. Witchel, *Linefs: Efficient smartnic offload of a distributed file system with pipeline parallelism*, in Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, 2021, pp. 756–771.

[19] Y. Kwon, H. Fingler, T. Hunt, S. Peter, E. Witchel, and T. Anderson, *Strata: A cross media file system*, in Proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 460–477.

[20] B. Li, Z. Ruan, W. Xiao, Y. Lu, Y. Xiong, A. Putnam, E. Chen, and L. Zhang, *Kv-direct: High-performance in-memory key-value store with programmable nic*, in Proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 137–152.

[21] Z. Liu, R. Lewis, R. Kettimuthu, K. Harms, P. Carns, N. Rao, I. Foster, and M. E. Papka, *Characterization and identification of hpc applications at leadership computing facility*, in Proceedings of the 34th ACM International Conference on Supercomputing, 2020, pp. 1–12.

[22] M. Mesnier, G. R. Ganger, and E. Riedel, *Object-based storage*, IEEE Communications Magazine, 41 (2003), pp. 84–90.

[23] M. P. Mesnier, M. Wachs, R. R. Simbasivan, J. Lopez, J. Hendricks, G. R. Ganger, and D. R. O'hallaron, *//trace: parallel trace replay with approximate causal events*, (2007).

[24] *Minimega*. `https://www.sandia.gov/minimega`. Accessed: Aug 28, 2024.

[25] L. Nelson, J. Van Geffen, E. Torlak, and X. Wang, *Specification and verification in the field: Applying formal methods to {BPF} just-in-time compilers in the linux kernel*, in 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), 2020, pp. 41–61.

[26] S. Neuwirth and A. K. Paul, *Parallel i/o evaluation techniques and emerging hpc workloads: A perspective*, in 2021 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, 2021, pp. 671–679.

[27] A. K. Paul, A. M. Karimi, and F. Wang, *Characterizing machine learning i/o workloads on leadership scale hpc systems*, in 2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, 2021, pp. 1–8.

[28] T. S. Pillai, V. Chidambaram, R. Alagappan, S. Al-Kiswany, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, *All file systems are not created equal: On the complexity of crafting {Crash-Consistent} applications*, in 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), 2014, pp. 433–448.

[29] *Redis*. `http://redis.io`.

[30] *Rocksdb*. `http://rocksdb.org`.

[31] H. Shan, K. Antypas, and J. Shalf, *Characterizing and predicting the i/o performance of hpc applications using a parameterized synthetic benchmark*, in SC'08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, IEEE, 2008, pp. 1–12.

[32] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright, *Modular hpc i/o characterization with darshan*, in 2016 5th workshop on extreme-scale programming tools (ESPT), IEEE, 2016, pp. 9–17.

[33] I. Stoica, D. Song, R. A. Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. E. Gonzalez, et al., *A berkeley view of systems challenges for ai*, arXiv preprint arXiv:1712.05855, (2017).

[34] Y. Wang, M. Kapritsos, L. Schmidt, L. Alvisi, and M. Dahlin, *Exalt: Empowering researchers to evaluate large-scale storage systems*, in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), 2014, pp. 129–141.

[35] S. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, *Ceph: A scalable, high-performance distributed file system*, in Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06), 2006, pp. 307–320.

[36] J. Yang, J. Izraelevitz, and S. Swanson, *Orion: A distributed file system for {Non-Volatile} main memory and {RDMA-Capable} networks*, in 17th USENIX Conference on File and Storage Technologies (FAST 19), 2019, pp. 221–234.

[37] B. Zhu, Y. Chen, Q. Wang, Y. Lu, and J. Shu, *Octopus+: An rdma-enabled distributed persistent memory file system*, ACM Transactions on Storage (TOS), 17 (2021), pp. 1–25.

SCALABLE APPLICATION-ORIENTED BENCHMARKING OF QUANTUM COMPUTERS

NOAH D. SIEKIERSKI*, AIDAN Q. WILBER-GAUTHIER†, AND STEFAN K. SERITAN‡

Abstract. Benchmarking quantum applications is an essential part of tracking progress towards quantum advantage and comparing different quantum devices on real-world problems. However, current application-oriented benchmarking approaches often lack scalability due to the need for expensive classical computation or are limited to testing full problem instances that may be too large to run on current devices. Here, we show how subcircuit volumetric benchmarking using mirror circuit fidelity estimation can be used to design scalable benchmarks from any quantum algorithm that uses only unitary operations. We demonstrate how to generate subcircuit volumetric benchmarks from an existing application-oriented benchmarking suite and show how a real quantum device performs on one such benchmark.

1. Introduction. Quantum computers promise to dramatically reduce the resources, both spatial and temporal, required to accomplish certain tasks of practical interest [6, 20, 1]. This promise is hampered in the present noisy intermediate-scale quantum (NISQ) era [14] by complex errors that make it infeasible to reliably run these tasks on a quantum computer. It is critical to utilize well-defined, accurate protocols to assess the performance of quantum hardware to track progress and compare between different devices and technologies. We call such assessment schemes *benchmarks*, and many have been developed. IBM's quantum volume benchmark [3] and randomized benchmarking (RB) [8, 12] have been used to capture the general performance of a quantum device by running classes of random circuits. It is also desirable to run *application-oriented* benchmarks that interrogate the performance of a device on circuits that represent real-world programs that users are likely to use a quantum computer for. Various benchmarking suites have been designed with these programs in mind [4, 13, 18, 5, 9, 11, 17]. A difficulty faced by both classes of benchmark is their scalability to many qubits. This inability to scale beyond several qubits is due to expensive classical pre-processing for circuit generation or expensive classical simulation to verify the circuit output. Some application benchmark generation schemes circumvent the scalability challenge by leveraging properties unique to the application that allow the creation of circuits with efficiently predictable outcomes or the usage of an application-specific success metric. However, since these approaches are application-specific, they do not provide a framework that is easily adaptable to any application one may wish to benchmark on a quantum computer.

Here, we show how subcircuit sampling and circuit mirroring [15] combined can be used to create volumetric benchmarks (VBs) [2] for any application. We refer to these as subcircuit volumetric benchmarks (SVBs), and our technique is amenable to any application circuit that represents a unitary evolution. We demonstrate its efficacy using selected circuits from the application-oriented benchmarking suite created by the Quantum Economic Development Consortium (QED-C).

2. Benchmark methodology. In this section, we describe our method for taking a high-level application circuit $\tilde{C}$ and transforming it into a circuit ensemble on which mirror circuit fidelity estimation (MCFE) can be performed. We begin by performing a compilation that maps $\tilde{C}$ onto the target quantum processor's qubit graph and transpiles the circuit to a U3-CX gate set, yielding the circuit C. C can then be expressed as a product of layers, i.e. $C = L_d L_{d-1} ... L_1$, where d is the virtual depth (see Section 2.2 for an explanation of

*Sandia National Laboratories, ndsieki@sandia.gov
†Sandia National Laboratories, aqwilbe@sandia.gov
‡Sandia National Laboratories, sserita@sandia.gov

the difference between virtual and physical depth) of the full circuit and each L_i contains either only U3 gates (a U3 layer) or only CX gates (a CX layer).

To generate the subcircuits $\{S_C\}$, we first specify a list of width-depth pairs (w_i, d_i). For each pair, we generate subcircuits via the strategy described in Section 2.3. For each subcircuit in $\{S_C\}$, we generate a pair of compilations: an exact "reference" compilation in the U3-CX gate set, which in our case is simply the subcircuit itself; and a test compilation whose process fidelity we want to assess when executed on the target quantum device. These compilations are further discussed in Sections 2.4 and 2.5. Each reference-test compilation pair is then used to generate three sets of mirror circuits, which are then executed on the target device to perform MCFE. These mirror circuit ensembles are further described in Section 2.7.

2.1. Initial compilation. The initial compilation of $\tilde{C}$ into C consists of two parts: mapping the high-level circuit onto the qubit connectivity graph of the device, and standardizing the gate set. A high-level circuit is specified in terms of *virtual* qubits, which are typically assumed to be directly connected to all other qubits and admit any unitary to act on them. When mapping to *hardware* qubits, neither of these assumptions typically holds. Most current quantum devices have limited connectivity and only support a particular set of native gates that must be combined in order to synthesize any unitary that is not directly supported. To overcome connectivity limitations, hardware qubits use SWAP operations, which are noisy. In order to limit the number of SWAP operations needed, the classical compiler selects a layout and routing for the circuit that minimizes the number of SWAPs needed and utilizes the highest fidelity qubits of the device as quantified by noise calibration data. The layout and routing induce a permutation of the virtual qubits that is easily correctable in classical post-processing. As an example, a circuit described with three virtual qubits that is compiled for a ten-qubit device may in the initial layout map virtual qubit 0 to physical qubit 3, virtual qubit 1 to physical qubit 5, and virtual qubit 2 to physical qubit 1. Then, over the course of the circuit, physical qubit 1 is swapped with physical qubit 5. By measuring first physical qubit 3, then physical qubit 1, and finally physical qubit 5, the order of the measured result exactly corresponds to the virtual qubit ordering.

The quantum device we run on, ibmq_brisbane, does not natively support U3 and CX operations. However, they are easily re-expressed in terms of native operations:

$$U3(\theta, \phi, \lambda) = e^{-i\frac{\pi+\theta}{2}} R_Z(\phi+\pi)\sqrt{X}R_Z(\theta+\pi)\sqrt{X}R_Z(\lambda), \tag{2.1}$$

$$CX_{a,b} = (I \otimes X)ECR_{a,b}([R_Z(-\pi)\sqrt{X}R_Z(-\pi)] \otimes R_Z(-\pi/2)), \tag{2.2}$$

where a and b are the control and target qubits of the CX operation, respectively.

2.2. Physical versus virtual depth. The circuit C is composed of U3 and CX layers of virtual depth 1. When these layers are compiled for execution on a quantum device, these U3 and CX operations must be transpiled to the native gate set of the target device. The transpiled depth of a circuit layer is referred to as its *physical* depth. In the case of ibmq_brisbane, a U3 gate requires five operations when transpiled. Only the $\sqrt{X}$ operations are physical, and since there are two, a U3 layer has physical depth 2. An analogous analysis shows that a CX layer has physical depth 3. Note that the physical depth of a given unitary operation will in general be device-dependent.

In this work, we use a modified depth metric to which each U3 layer contributes 2 and each CX layer contributes 1. This modified depth metric is based upon the physical depth of the U3 and CX operations on the 27-qubit generation of IBM quantum processors, and we plan to replace it with the appropriate physical depth metric in the future. Note that when the word "depth" is used in this paper, though, it refers to the modified depth metric.

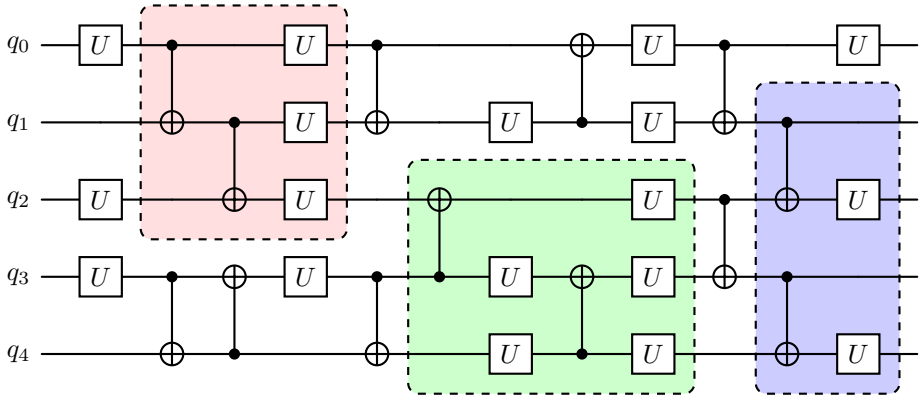


FIG. 2.1. *Subcircuit selection example. Each section of the full circuit captured in a colored box is a subcircuit sample.*

2.3. Subcircuit selection. The aim of subcircuit selection is to sample a set of subcircuits $\{S_C\}$ from the full circuit C such that $\{S_C\}$ is representative of C. Figure 2.1 provides shows what these subcircuits look like. Each subcircuit consists of a connected set of qubits $\{q_i\}$ that are a subset of the hardware qubits utilized by C, along with a contiguous set of layers $\{L_i\}$ from C that act on those qubits. Given a width-depth pair (w_i, d_i), we randomly select a starting layer L_{start} to obtain a contiguous set of layers that has depth d_i. With L_{start} identified, we randomly select a $\{q_i\}$ of size w_i. The operations in each layer that do not act on $\{q_i\}$ are removed. This yields a subcircuit of width w_i and depth d_i.

A complication arises when a subcircuit layer contains a CX gate where only one of the qubits acted on by the gate is included in $\{q_i\}$. We refer to these gates as "dangling" and discard them from the sampled subcircuit.

2.4. Reference compilation. A reference compilation r of a circuit s in $\{S_C\}$ is created by transpiling s to the gate set needed for randomized compilation, which is discussed in Section 2.6. The circuit r is an exact compilation, i.e. it is logically equivalent to c. Quantum computing SDKs like IBM's Qiskit and Quantinuum's TKET offer approximate compilation passes that trade off logical accuracy for less noisy operations, and these passes are forbidden. This work uses a randomized compilation protocol that requires r to be expressed in a U3-CX gate set, and since s is already expressed in a U3-CX gate set, we take $r = s$.

2.5. Test compilation. A test compilation t of a circuit s in $\{S_C\}$ is created by passing s through a full-stack quantum computer's classical pre-processor. In this step, we forbid certain optimizations that the classical pre-processor could make if c were being run in isolation as a high-level circuit. These include the layout and routing steps described in Section 2.1, along with the removal of idle operations, with the rationale that such optimizations cannot be realized when the $\{S_C\}$ are executed *inside* C. Any layout and routing optimization to a subcircuit would require additional SWAPs to be set up and undone in the context of its execution inside C, which would reduce the overall quality of the execution of C. If a qubit is idling for some layers in C and a subcircuit is selected where the idle operations can be removed from the subcircuit, the idle operations are nonetheless maintained because they are necessary when the subcircuit is executed inside of C. This work uses $t = s$, i.e. we use the sampled subcircuit with no additional optimizations.

2.6. Randomized compilation. Our goal is to estimate the process fidelity of each test compilation t. If t experiences coherent errors, then embedding it in a larger circuit – as is done in MCFE – can cause these coherent errors inside t to interact coherently with errors in other parts of the larger circuit. Randomized compilation [19, 7] can be applied to parts of the larger circuit to ensure they have only stochastic errors. We follow the randomized compilation protocol of Reference [16]. Consider a circuit $c = l_{\tilde{d}} e_{\tilde{d}-1} l_{\tilde{d}-1} ... e_2 l_2 e_1 l_1$ on n qubits, where the l_i are U3 layers and the e_i are CX layers. The randomized compilation protocol requires the sampling of $\tilde{d}$ n-qubit Pauli operators P_i for $i = 1, 2, ..., \tilde{d}$. Each l_i in c is then replaced by a new layer l'_i that satisfies $l'_i = P_i l_i e_{i-1} P_{i-1} e_{i-1}^\dagger$, where P_0 is the identity operator and e_0 is an empty layer. This new circuit, denoted by $f_{rc}(c)$, is logically equivalent to c up to post-multiplication by a Pauli.

2.7. Mirror circuit fidelity estimation. The goal of a quantum processor tasked with implementing an n-qubit unitary U with superoperator representation $\mathcal{U}[\rho] = U\rho U^\dagger$ is to implement a quantum process Λ that is as close as possible to $\mathcal{U}$. We measure how close Λ is to $\mathcal{U}$ with the *process fidelity*, F:

$$F(\Lambda, \mathcal{U}) = \frac{1}{4^n} \operatorname{tr}\left(\mathcal{U}^\dagger \Lambda\right). \tag{2.3}$$

It is convenient to work with a rescaling of the process fidelity known as the *effective polarization* γ:

$$\gamma(\Lambda, \mathcal{U}) = \frac{4^n}{4^n - 1} F(\Lambda, \mathcal{U}) - \frac{1}{4^n - 1}. \tag{2.4}$$

The MCFE protocol [16] is a robust fidelity estimation technique based on the Loschmidt echo [10] that estimates the process fidelity. It combines motion reversal with randomized compiling and multiple circuit ensembles to separate state preparation and measurement (SPAM) errors and reference compilation errors from test compilation errors and provides an estimate of the process fidelity of the test compilation.

Given the test compilation t and reference compilation r that correspond to a given c in $\{S_C\}$, we construct three types of mirror circuits M_1, M_2, and M_3:

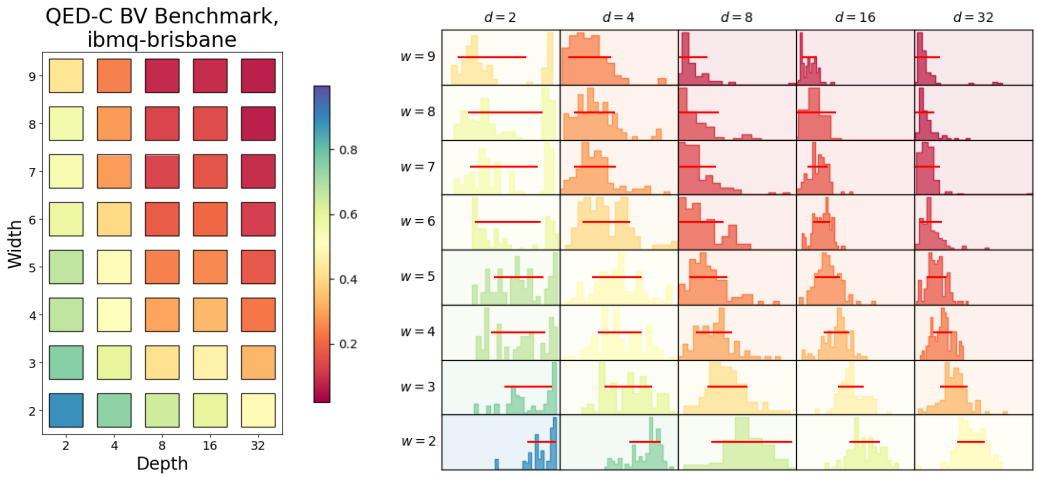
$$M_1 = f_{rc}(p_{\text{rev}} r_{\text{rev}})\, tp, \tag{2.5}$$
$$M_2 = f_{rc}(p_{\text{rev}} r_{\text{rev}} rp), \tag{2.6}$$
$$M_3 = f_{rc}(p_{\text{rev}} p). \tag{2.7}$$

Here, the "rev" subscript refers to a layer-by-layer inverse of the circuit. The circuit p is a state preparation circuit that implements a local unitary 2-design. We generate $k \gg 1$ of each type of mirror circuit. When implemented perfectly, each mirror circuit will output a single efficiently predictable bitstring, which is what makes MCFE scalable. To estimate F, we compute the *observed polarization* of every mirror circuit M_i:

$$\gamma(M_i) = \frac{4^n}{4^n - 1} \left[\sum_{k=0}^{n} \left(-\frac{1}{2}\right)^k h_k\right] - \frac{1}{4^n - 1}, \tag{2.8}$$

where h_k is the probability the circuit outputs a bitstring that is Hamming distance k from the target bitstring. The process fidelity for a circuit c is then estimated as

$$F \approx 1 - \frac{4^n - 1}{4^n} \left(1 - \frac{\text{avg}[\gamma(M_1)]}{\text{avg}[\gamma(M_2)]\, \text{avg}[\gamma(M_3)]}\right). \tag{2.9}$$



(a) BV fidelities

(b) Subcircuit fidelity histograms

FIG. 3.1. *Mirror circuit process fidelities by width and depth. The VB plot in (a) shows the mean process fidelities while (b) shows the histograms of the process fidelity estimates for the 100 subcircuits per width-depth pair. The red horizontal bar indicates* $\pm 1\sigma$ *from the mean.*

3. Results and Discussion. We applied our SVB creation framework to the Bernstein-Vazirani (BV) benchmark developed by the Quantum Economic Development Consortium [11]. We generated 10 20-qubit BV circuits and generated 100 subcircuits per width-depth pair. The width ranges linearly from 1 qubit to 9 qubits, and the depth ranges exponentially from 2 to 32. We used 20 mirror circuit samples per subcircuit and 100 SPAM reference circuits (these correspond to the M_3 circuits) per qubit subset. We ran these circuits on ibmq_brisbane, a 127-qubit quantum processor. Figure 3 shows the process fidelity for each width-depth pair. A fidelity of 1.0 indicates that the circuit performed perfectly, and decreasing fidelity toward 0.0 indicates decreasing accuracy in bitstring output. We observe that the process fidelity decreases with increasing width and depth, which is reasonable. A greater width means there are more qubits and operations that errors can occur on. Circuits of greater depth have more gates or idle time, both of which contribute to errors.

We notice that many of the histograms in Figure 3.1(b) are not sharply peaked. The depth 2, width 6 through 9 histograms appear bimodal, and many of the subcircuit shapes with intermediate mean process fidelity have a broad process fidelity distribution. This behavior suggests that circuit shape alone may not always be sufficient to predict process fidelity. For instance, a depth 2 subcircuit could consist of either a single U3 layer or two CX layers, and the error rates of the CX gates are an order of magnitude greater than their U3 counterparts. Despite having the same shape, these two cases would likely have very different process fidelities. We can connect the broad process fidelity distributions for the subcircuit shapes with intermediate mean process fidelity to the capability regions described in Reference [15]. In that work, a VB plot is partitioned into three capability regions: a high fidelity region where all circuits succeed, a low fidelity region where no circuits succeed, and an intermediate region where the success of a circuit depends on its structure in addition to its shape. Our results reinforce the interpretation of the intermediate region as one where knowledge of circuit characteristics beyond shape is required to predict success.

4. Conclusion. In this work, we described how to create scalable benchmarks from any application. We demonstrated the creation of such a benchmark and executed it on current hardware. Our results agree with the effect that increasing the number of qubits or increasing the circuit depth has on the likelihood of errors. We also observed that the shape of a subcircuit is not sufficient to understand its process fidelity, with a more pronounced effect for particular subcircuit shapes.

Running the same benchmark on another quantum device would provide insight into the comparative performance of the two. Additionally, running a benchmark created from a different application would highlight performance differences across applications on the same device and would allow us to understand if there are cases where certain devices are superior to others for only certain applications. Another open question is if these SVBs are effective predictors of success on untested full problem instances. Such predictive power would allow users to know the limits of the quantum hardware for their application without needing to run the problem first.

REFERENCES

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, *Quantum supremacy using a programmable superconducting processor*, Nature, 574 (2019), pp. 505–510.

[2] R. Blume-Kohout and K. C. Young, *A volumetric framework for quantum computer benchmarks*, Quantum, 4 (2020), p. 362.

[3] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, *Validating quantum computers using randomized model circuits*, Phys. Rev. A, 100 (2019), p. 032328.

[4] Y. Dong and L. Lin, *Random circuit block-encoded matrix and a proposal of quantum LINPACK benchmark*, Phys. Rev. A, 103 (2021), p. 062412.

[5] H. Donkers, K. Mesman, Z. Al-Ars, and M. Möller, *QPack Scores: Quantitative performance metrics for application-oriented quantum computer benchmarking*, arXiv, (2022).

[6] J. Kallaugher, O. Parekh, and N. Voronova, *Exponential Quantum Space Advantage for Approximating Maximum Directed Cut in the Streaming Model*, arXiv, (2023).

[7] E. Knill, *Quantum computing with realistically noisy devices*, Nature, 434 (2005), pp. 39–44.

[8] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, *Randomized benchmarking of quantum gates*, Phys. Rev. A, 77 (2008), p. 012307.

[9] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, *QASMBench: A Low-Level Quantum Benchmark Suite for NISQ Evaluation and Simulation*, ACM Trans. Quantum Comput., 4 (2023).

[10] J. Loschmidt, *Uber den zustand des warmegleichgewichts eines systems von korpern mit rucksicht auf die schwerkraft; ii-73*, Sitzungsber. Akad. Wiss., (1876), p. 128.

[11] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, and T. Proctor, *Application-Oriented Performance Benchmarks for Quantum Computing*, IEEE Trans. Quantum Eng., 4 (2023), pp. 1–32.

[12] E. Magesan, J. M. Gambetta, and J. Emerson, *Scalable and Robust Randomized Benchmarking of Quantum Processes*, Phys. Rev. Lett., 106 (2011), p. 180504.

[13] S. Martiel, T. Ayral, and C. Allouche, *Benchmarking Quantum Coprocessors in an Application-Centric, Hardware-Agnostic, and Scalable Way*, IEEE Trans. Quantum Eng., 2 (2021), pp. 1–11.

[14] J. Preskill, *Quantum Computing in the NISQ era and beyond*, Quantum, 2 (2018), p. 79.

[15] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, *Measuring the capabilities of quantum computers*, Nat. Phys., 18 (2022), pp. 75–79.

[16] T. Proctor, S. Seritan, E. Nielsen, K. Rudinger, K. Young, R. Blume-Kohout, and M. Sarovar, *Establishing trust in quantum computations*, arXiv, (2022).

[17] N. Quetschlich, L. Burgholzer, and R. Wille, *MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing*, Quantum, 7 (2023), p. 1062.

[18] T. Tomesh, P. Gokhale, V. Omole, G. Ravi, K. N. Smith, J. Viszlai, X. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, *Supermarq: A scalable quantum benchmark suite*, in 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), IEEE Computer Society, apr 2022, pp. 587–603.

[19] J. J. Wallman and J. Emerson, *Noise tailoring for scalable quantum computation via randomized compiling*, Phys. Rev. A, 94 (2016), p. 052325.

[20] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, P. Hu, X.-Y. Yang, W.-J. Zhang, H. Li, Y. Li, X. Jiang, L. Gan, G. Yang, L. You, Z. Wang, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, *Quantum computational advantage using photons.*, Science, 370 (2020), pp. 1460–1463.

III. Machine Learning

Articles in this section discuss the application of machine learning techniques to problems in computational physics and mathematics. This includes the design of neural network architectures, the efficient use of data for training and validation, or the use of machine learning to provide insights into materials, high-energy applications, atmospheric phenomena, and structural mechanics.

1. *H. Bayat, M.A. Cusentino* and *J.M. Goff* Charge Dependent Machine Learned Models for Atomistic Simulations of Divertor Materials
2. *A.K. Boahen* and *W.L. Davis IV* In Situ Machine Learning for Intelligent Data Capture and Event Detection
3. *M.C. Gaitan-Cardenas, C. Siefert* and *S.W. Tsai* Large Language Model Accuracy on Post-processed AI-generated Code
4. *D. Deighan, J. Actor* and *R. Patel* Mixture of Neural Operator Experts for Non-trivial Boundary Conditions and Model Selection
5. *A. Feeney* and *S. Rajamanickam* Exploring Machine Learning Surrogates for Molecular Dynamics Simulations
6. *I. Furrick, M. Wood,* and *A. Hensley* Designing a Machine-learned Interatomic Potential for Gold-promoted Nickel Catalysts Utilizing Magnetic Training Data
7. *M. Gahl, W. Chapman, S. Agarwal,* and *F.S. Chance* Event Detection Using Neural Networks Robust to Statistically Similar Distractors
8. *J.D. Gonzales-Pasion, M. Wood,* and *A.J. Hensley* Machine Learned Interatomic Potential Development Accelerated via Large-language Models for Nickel-gold
9. *Q. Mason* and *K.A. Maupin* Decision Tree Machine Learning Model Construction for Particle Simulation
10. *C. Mullen, E. Salas,* and *J. Goff* Breaking Bad Structure Generation: Methods For Systematic, Data-driven Atomistic Structures for ML Model Training
11. *P. Mutia* and *J. Davis* Ollama-Assisted Function Calls in Leap
12. *K.A. Ohene-Obeng* and *K. Maupin* Scientific Machine Learning for Surrogate Modeling
13. *J. Paez* and *R. Patel* Quantifying Aleatoric Uncertainty in Operator Learning using Generative Networks
14. *D. Rodriguez* and *M. Perego* Coupled Deep Neural Operators as a Surrogate Model for Ice-sheet Dynamics

M. Adams
T. Casey
B.W. Reuter

December 17, 2024

CHARGE DEPENDENT MACHINE LEARNED MODELS FOR ATOMISTIC SIMULATIONS OF DIVERTOR MATERIALS

HADIA BAYAT*, MARY ALICE CUSENTINO†, AND JAMES MICHAEL GOFF‡

Abstract. We study tungsten (W) divertors in fusion reactors, focusing on the impact of implanted atoms such as nitrogen (N), helium (He), and hydrogen (H). These elements implant in the divertor surface, displace W atoms into interstitial sites, and form a 'fuzz' that drives nanostructure growth, increasing brittleness. We developed a charge-dependent machine-learned potential for W-N interactions, trained using FitSNAP and implemented in LAMMPS, to assess the differences between a charge-independent and charge-dependent interatomic potential in providing a more accurate physical representation. Our charge-dependent model demonstrated stable dynamics without needing the Ziegler-Biersack-Littmark (ZBL) potential. Uniquely, it incorporates long-range interactions into the charge-dependent model, an approach not previously explored for this system. Our charge-dependent model exhibits significantly greater stability than a charge-independent model, displaying the importance of including charge-dependent interactions with long-range effects. Parameters will be optimized using DAKOTA, and future work will focus on refining the model by optimizing hyperparameters and incorporating objective functions such as formation energies, defects, and adsorption energies. This work represents a significant step towards evaluating the impact of charge on atomistic simulations of materials.

1. Introduction. Fossil fuels contribute significantly to global greenhouse gas emissions, leading to climate change, altered weather patterns, and disruptions in natural ecosystems. In the pursuit of sustainable and clean energy sources, nuclear fusion reactors present a promising solution. Fusion technology, leveraging deuterium-tritium reactions, promises a long-term energy solution free from the environmental drawbacks associated with greenhouse gas emissions and long-lived radioactive waste. However, the deployment of fusion technology on a large scale is contingent upon overcoming several technical challenges, particularly in enhancing the durability and performance of reactor components such as the divertor[18, 19].

The divertor plays a crucial role in maintaining reactor integrity by facilitating heat and ash extraction, minimizing plasma contamination, and protecting reactor walls from thermal and neutronic loads. A critical issue affecting divertor performance is the interaction with nitrogen, introduced within fusion reactors to improve plasma confinement and reduce core radiation. These interactions lead to the formation of a tungsten 'fuzz,' characterized by nanostructure growth and increased brittleness, which significantly compromises the divertor's functionality[18][13].

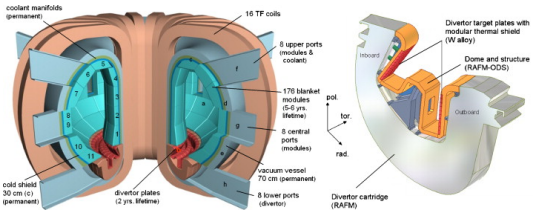


Fig. 1.1: Diagram of a nuclear fusion reactor, highlighting the divertor (right side), which plays a crucial role in heat extraction (reproduced from Ref. [13])

Addressing the complexities of these interactions necessitates a robust simulation ap-

*University of California San Diego, habayat@ucsd.edu
†Sandia National Laboratories, mcusent@sandia.gov
‡Sandia National Laboratories, jmgoff@sandia.gov

proach. While Density Functional Theory (DFT) offers precise electronic structure calculations, its application is inherently limited by computational demands, restricting its utility to smaller systems and shorter temporal scales. This limitation is particularly pronounced in studying divertor surface phenomena, where the interactions span extensive spatial domains and evolve over prolonged periods[11]. In conjunction with DFT, machine-learned interatomic potentials (MLIP) can be used to model larger systems over longer timescales with near ab-initio accuracy, offering a more comprehensive understanding of these complex processes. Our research introduces a polarizable MLIP tailored for W-N interactions. The

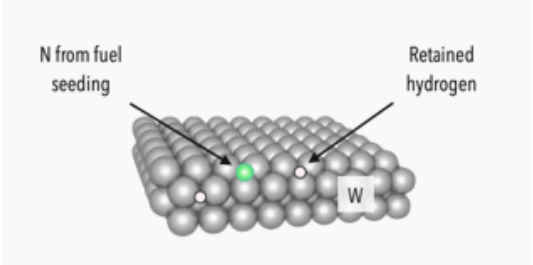


Fig. 1.2: Visualization of nitrogen seeding on tungsten divetor surface. Nitrogen is introduced to help effectively cool the edge plasma by radiating heat away from the reactor walls, protecting the plasma-facing components from excessive heat loads

model significantly advances the field by incorporating variable atomic charges and dynamic charge transfer processes. Given the limited atomic screening at the W divertor surface, our approach is designed to capture both short-range and long-range interactions. This nuanced approach is essential for accurately modeling nitrogen implantation on tungsten, a process where conventional short-range focused MLIPs often fail to capture the complex interactions and charge transfer dynamics. We will incorporate MD simulations to observe and verify the variations between our charge-independent and charge-dependent models in displaying accurate atomistic physics.

2. Methods. To rigorously evaluate the efficacy and accuracy of our charge-dependent model, we will conduct a comparative analysis with two additional models: a shadow molecular dynamics (SMD) approach combined with the Atomic Cluster Expansion (ACE) method for flexible charge models, and an independent charge model (ICM). This comparative study not only validates the effectiveness of our range-separated MLIP but also illuminates the strengths and limitations of each modeling approach. The introduction of this polarizable MLIP not only addresses the limitations inherent in previous simulation models but also represents a substantial methodological leap forward. By enabling a more accurate and comprehensive simulation of charge dynamics, our approach facilitates a deeper understanding of the material degradation processes that compromise divertor functionality. This enhanced understanding is instrumental in exploring alternative materials and strategies to improve divertor durability, thereby contributing to the advancement of fusion technology as a sustainable energy solution.

Incorporating charge dynamics into our machine-learned interatomic potentials (MLIPs) necessitates a refined, range-separated approach, particularly to account for the complexities of long-range charge interactions. For the short-range domain, we employ a machine-learned interatomic potential (MLIP) utilizing the Atomic Cluster Expansion (ACE), as detailed in work conducted by Drautz (2019), Dusson et al. (2019), and Goff et al. (2024) [3, 4, 6]. To complement our handling of short-range interactions, we have modeled the long-range

Coulombic forces using the Streitz-Mintmire potential [17]. Such interactions are pivotal for accurately capturing the charge-dependent behavior of the system, a critical aspect in systems where electrostatic forces are a dominant influence, including metals, ionic crystals, and polarizable materials. Furthermore, to accurately simulate dynamic charge equilibration, which is critical for understanding the charge transfer mechanisms at play, we implement the QEq method with Streitz parameters[17] This methodological framework allows for the dynamic adjustment of atomic charges, reflecting the charge redistribution that occurs during the interaction processes. The contributions of each component to the overall interaction model can be visualized in Fig.2.1 offering a clear depiction of how these elements synergize to accurately represent the complex dynamics of the W-N system. The phenomenon

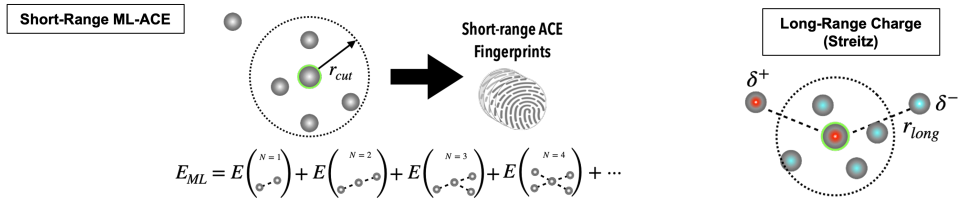


Fig. 2.1: Short range interactions will be incorporated within our system through the machine-learned ACE potential, whereas long-range interactions will be accounted for through the streitz-mintmire potential

of screening plays a significant role in our modeling approach. Screening describes how the effective interaction between charged particles is moderated by the presence of other charges, such as electrons or ions, thereby reducing the force experienced by each particle. This effect is paramount in systems characterized by significant electrostatic interactions, as it fundamentally alters the behavior of charged particles within the material.

To dynamically manage the redistribution of atomic charges, we integrate the QEq charge equilibration method alongside the Streitz-Mintmire potential. This approach ensures that at every timestep, atomic charges are adjusted to minimize the system's electrostatic energy while conserving the total charge. By doing so, we can accurately simulate the dynamic nature of charge distribution and its impact on the system's overall behavior[9, 17].

Moreover, the inclusion of screening in our model allows us to more accurately represent the potential experienced by an atom. It acknowledges that this potential is not solely the result of direct long-range interactions but is also significantly influenced by the surrounding environmental charges. This comprehensive approach to modeling charge interactions and screening effects underscores our commitment to capturing the intricate dynamics of the W-N system, ensuring that our simulations reflect the complex interplay of forces that govern material behavior at the atomic level.

The combination of short-range MLIP and the Streitz-Mintmire potential facilitates a thorough analysis of interactions at the atomic level. Most machine-learned interatomic potentials are developed with only short-range forces because they are the dominant contributor to the total energy and dynamics of atoms. To account for long-range charge interactions, we use the following range-separated potential:

$$E_{pot}(R, Q) = E_{ACE}(R) + E_{streitz}(R, Q) \tag{2.1}$$

where $E_{ACE}(R)$ represents the short-range energy. In this work, $E_{ACE}(R)$ is a linear ACE model, which solely depends on interatomic distance R [6, 5] . In Eq. (2.1), $E_{streitz}(R, Q)$ is the long-range electrostatic potential from Ref. [17], where Q represents the atomic charges.

The first term in Eq. (2.1) shows the traditional form of machine-learned potentials, focusing on short-range interactions through a linear ACE model. While this is suitable for some materials systems, it is expected that the long-range charge interaction term will help reproduce more accurate predictions of Nitrogen embedding at tungsten surfaces. We extend this model to include the $E_{streitz}$ term, to ensure that both short-range and long-range interactions are considered in our potential energy calculations.

2.1. Atomic Cluster Expansion. The ACE potentials are a class of ML potentials that can be used for arbitrary body-ordered interactions and chemistries. The ACE descriptors encode detailed information about atomic interactions locally around a central atom[3]. This, along with the ability to systematically improve ACE models with higher-order descriptors makes them an excellent method to use for the short-range term in the range-separated potential, Eq. (2.1). The local environment around an atom is described by the positions of its neighboring atoms R_{ij}. The total potential energy of a system of X atoms can be expressed as a sum of contributions from atoms, where E_i is the energy contribution from the i_{th} atom, calculated according to its surrounding environment. Each E_i can be expanded in terms of cluster functions, which account for interactions using contributions from clusters of atoms. In practice, rotation and permutation invariant ACE descriptors (cluster functions) are used. The general potential energy function is expressed as a sum of contributions from different atomic clusters:

$$E_i = \sum_{\boldsymbol{\nu}} B_{i\boldsymbol{\nu}}(\boldsymbol{\sigma}) c_{\boldsymbol{\nu}} \tag{2.2}$$

Where $B_{\boldsymbol{\nu}}$ represents the rotation and permutation invariant ACE descriptors, $\boldsymbol{\sigma}$ collects the variable the atomic neighbor positions and chemistries, while $c_{\boldsymbol{\nu}}$ are linear model coefficients to be determined. The linear model coefficients are determined within FitSNAP using linear regression methods. We employ a ridge regression model, which calculates the coefficients in FitSNAP by fitting against DFT energies and forces.

2.2. Streitz-Mintmire Potential. In our exploration of the W-N system, the Streitz-Mintmire potential emerges as a pivotal component, specifically engineered to adeptly manage charge transfer and polarization effects prevalent in ionic systems. This empirical potential is seamlessly integrated with charge equilibration methods, notably the QEq charge equilibration technique, to dynamically modulate atomic charges. The QEq method, a computational strategy, assigns partial charges to atoms within a molecular framework through a second-order Taylor expansion, focusing on charge-dependent energy. This assignment is crucial not only for the charge equilibration process but also serves as a foundational step in the iterative refinement of our potential's parameters.

Through extensive simulations conducted with LAMMPS, this iterative process has been instrumental in identifying the sensitivity of parameters that are vital for both the QEq calculations and the fitting of our potential. Establishing an optimal range for these parameters is essential for ensuring the accuracy and reliability of our model, particularly in capturing the intricate dynamics of the W-N system. We have displayed our results for the slater exponent in Fig 2.2. These results underscore the importance of carefully selecting ζvalues to avoid spuriously large charges from charge equilibration. This analysis has been instrumental in determining our initial set of parameters for both fitting our model and during charge equilibration, as displayed in table 2.1[12] . Applying the Streitz potential, which aims to conserve the total charge within the system, results in a series of linear equations. Solving these equations allows for determining equilibrium charges, ensuring that the system's charge distribution is accurately represented. The total energy $E_{Streitz}$ is

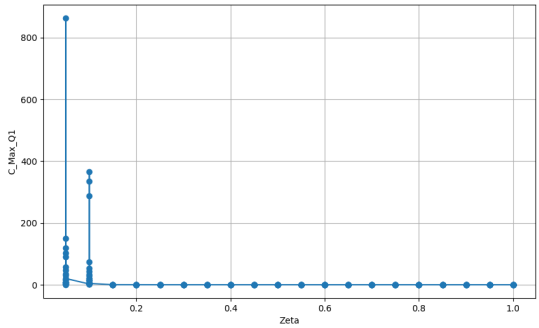


Fig. 2.2: ζ vs Q_{max} plot

Table 2.1: Set of Streitz Parameters

Type	χ **(eV)**	J **(eV)**	$\gamma\left(\frac{1}{\mathring{A}}\right)$	$\zeta\left(\frac{1}{\mathring{A}}\right)$	Z
N	6.57	7.98	0.00	0.70	0.60
n	6.57	7.98	0.00	0.70	0.60
W	4.34	3.52	0.00	0.60	0.50
w	4.34	3.52	0.00	0.60	0.50

the sum of long-range interactions as expressed in the following equation:

$$E_{Streitz} = \frac{1}{2}\sum_{i\neq j}\frac{(q_i + Z_i)(q_j + Z_j)}{r_{ij}f(r_{ij})} + \sum_i \left(\chi_i q_i + \frac{1}{2}J_i q_i^2\right) \tag{2.3}$$

Here $f(r_{ij}) = \exp(-\gamma r_{ij}) + \frac{1}{r_{ij}}\exp(-\zeta r_{ij})$. $f(r_{ij})$ represents a screening function that modifies the effective distance between charges to account for any screening effects. The screening function parameter γ represents the degree to which the Coulomb interactions between charged particles are screened by the presence of other charges or the medium, ζ is the Slater exponent, an exponential decay parameter for charge distribution, and Z is the effective core charge of the atom of interest[12]. The Streitz potential further includes χ_i, the Mulliken electronegativity of atom i, which drives the charge transfer force, and J_i is the self-Coulomb repulsion parameter, representing the energy penalty associated with deviating from the neutral state[3]. The functional form of the Streitz potential can be simplified further:

$$E_{Streitz} = ((\chi_i + \sum_J J_{ij}q_j)q_i) \tag{2.4}$$

The long-range nature of Coulomb forces means that interactions between distant particles and their periodic images cannot be ignored, but directly summing them leads to poor convergence and inefficiencies. The Ewald summation method was implemented within our model to efficiently compute the electrostatic energy. The method evaluates the problem by splitting contributions into short-range interactions handled in real space and long-range interactions managed in reciprocal space. The decomposition allows for the efficient calculation of electrostatic interactions in systems with periodic boundary conditions. This allows

for the potential to focus on short-range forces, while Ewald summation handles the long-range electrostatics, ensuring that all relevant interactions, including any screened effects, are properly accounted for[20][1].

2.3. ACE Hyperparemter Sampling with DAKOTA. To optimize our model further, we employed Dakota, an open-source toolkit that supports a variety of optimization algorithms. Utilizing the Single Objective Genetic Algorithm (SOGA) and Efficient Global Optimization (EGO) methods, we've been able to refine our model parameters efficiently. SOGA and EGO's capabilities in handling complex optimization problems and computationally expensive objective function evaluations have been instrumental in enhancing our model's accuracy and reliability[15][16].

EGO stands out for its integration of surrogate modeling with a global search strategy, proving exceptionally effective for scenarios where objective function evaluations are notably expensive. By incorporating noise into the surrogate model and utilizing an acquisition function to pinpoint unexplored yet promising regions of the search space, EGO facilitates an efficient exploration and identification of optimal regions[8]. This method's probabilistic approach is invaluable, especially when quantifying the uncertainty in objective function evaluations and minimizing the number of necessary assessments[10][7].

Our optimization efforts using DAKOTA are not merely confined to the present but extend into future work aimed at refining the model through hyperparameter optimization and the integration of objective functions such as formation energies, defects, and adsorption energies. A preliminary range of values for our ACE hyperparameters, determined through sensitivity tests with EGO within DAKOTA, is outlined in Table 2.2. These parameter ranges were determined after a thorough charge equilibration process and the analysis of approximately 4,500 candidate runs.

Table 2.2: EGO Hyperparameters (hyperparameter meanings illustrated in Fig. 2.3)

Rcutfac N-N	**Rcutfac N-W**	**Rcutfac W-W**	**Lambda N-N**	**Lambda N-W**	**Lambda W-W**	α
$3.052 - 3.169$	$4.140 - 4.240$	$4.666 - 5.306$	$0.988 - 0.989$	$0.808 - 0.838$	$0.431 - 0.794$	-4.93

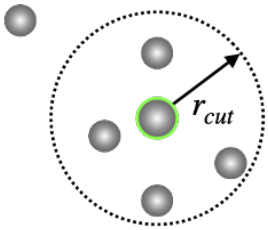
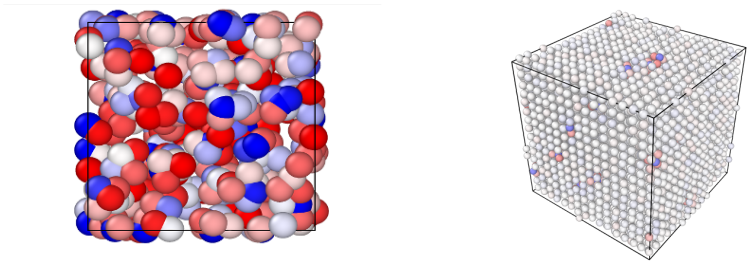


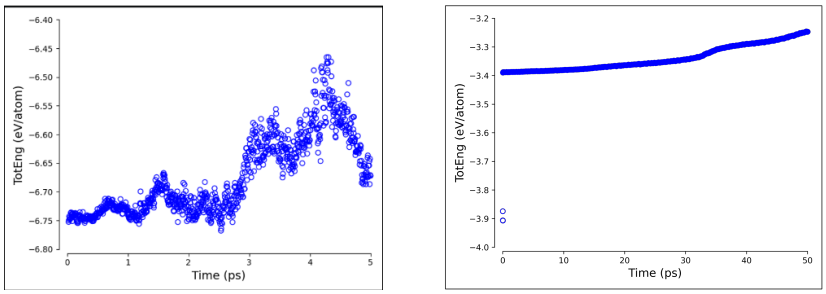
Fig. 2.3: Illustration of the cutoff radius used for Atomic Cluster Expansion (ACE) fingerprints, which represents the radial distance considered for atomic neighbors. The lambda parameters in the ACE functions apply importance sampling, giving more weight to closer atomic neighbors to enhance the accuracy of the radial ACE functions for a given training dataset.

Having established the methodologies for modeling the short-range and long-range interactions and the optimization techniques employed, we now present the results of our simulations. The following section discusses the accuracy, efficiency, and applicability of our machine-learned model in capturing the interactions within the tungsten-nitrogen system.



(a) Charge Independent Model (b) Charge Dependent Model

Fig. 3.1: Unstable versus Stable LAMMPS NVE Simulations



(a) Charge Independent Model (b) Charge Dependent Model

Fig. 3.2: Graphs of total energy versus time for our stable and unstable dynamics systems. We are observing energy drift; the top graph experiencing at a rate of 5 meV/ps, and the bottom at a rate of 3 meV/ps. Current optimization is working on addressing drift and predicting properties (such as defects with charge)

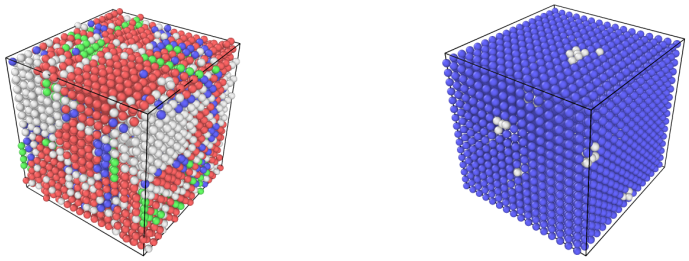
.

3. Results. We have developed a machine-learned model for the W-N system that exhibits stable dynamics. A stable range-separated W-N potential was trained. The hyperparameters for ACE and Streitz terms in this model were within the range of viable hyperparameters determined by sensitivity tests and Dakota searches. To demonstrate this potential along with other candidates, MD simulations were performed on BCC W with N defects in NVE ensembles at 2,000 K. Snapshots of these MD simulations were visualized using Ovito and colored according to charge[14]. These results are presented in Figure 3.1.

The total energy of both systems was graphed as a function of time, and these plots are displayed in Figure 3.2.

Notably, we observe that we were able to obtain a stable potential with qualitatively accurate physics. The charge around the negative N defects is positive. Not all candidate potentials yield realistic physics and chemistry, as seen in Fig. 3.1a. To study W-N fuzz formation, the stable candidate will be further refined. One feature of these simulations is that there is nonphysical drift in the energy of the system over long integration periods. Energy drift is a phenomenon in molecular dynamics simulations where the average total energy of the system drifts away from its initial value. While several factors can contribute to this, it is likely due to the QEq charge equilibration in this case.

The unstable candidates experience larger drifts along with the unrealistic physics and



(a) Charge Independent Model (b) Charge Dependent Model

Fig. 3.3: Charge Independent versus Dependent W-N models, highlighting incorrect and correct (blue) crystal structure predictions

chemistry. The drift in Fig. 3.3 is 5 meV/ps with a large amount of fluctuation, while the more stable potential, Fig. 3.2b with realistic physics is steady with a drift of 4 meV/ps. To mitigate energy drift, it is crucial to carefully select and tune the simulation parameters, such as using a more accurate integrator, reducing the time step, and ensuring proper energy conservation techniques. In the future, shadow MD schemes will be employed to more rigorously conserve energy during charge equilibration.[6] Despite the presence of energy drift, the simulations remain largely stable, and the models can still produce some accurate predictions for physical and chemical phenomena [2].

The model depicted in Figure 3.1b accurately represents physical properties, with positively charged tungsten atoms encircling negatively charged nitrogen atoms. This arrangement aligns with expectations, given nitrogen's significantly higher electronegativity compared to tungsten. Furthermore, the charges observed are within a plausible range. Tungsten possesses 6 electrons in its valence shell, whereas nitrogen has 5, allowing tungsten atoms to carry a charge of up to +6 and nitrogen atoms up to -3. These characteristics are evident in the model.

We examined the crystal structure to verify its adherence to the body-centered cubic (BCC) orientation defined in our LAMMPS simulation. The charge-dependent model consistently maintained this structure, while the charge-independent model deviated, transforming into a mix of face-centered cubic (FCC), BCC, and other crystal structures. The comparative analysis of these two models is presented in Fig. 3.3.

4. Conclusions. We have developed a stable machine-learned charge-dependent potential for a W-N system. Our results demonstrate that including charge produces a model that predicts correct charge transfer characteristics for W with N defects in it. This model was obtained by making a range-separated potential with hyperparameters within reasonable ranges. Charge-dependent potentials were obtained with more correct physical and chemical properties compared to charge-independent potentials (before optimization). We highlight how rapidly potentials with stable dynamics, can be produced once charge is included. This is often a challenge with charge-independent MLIPs.

Future work will focus on incorporating additional objective functions to further fine-tune our charge-dependent model while comparing this model to a shadow Born-Oppenheimer potential scheme for flexible charge models, as well as conducting further comparisons to charge-dependent simulations. Given the stable nature of our charge-dependent potential, we anticipate observing more consistent properties, such as vacancy and adsorption energies, in subsequent simulations.

Despite the promising results, some challenges and limitations need to be addressed. For instance, the current model may require further refinement to effectively handle our target system, which includes Hydrogen and Helium, due to the complexity of these elements and the extreme conditions involved. Our developed model has potential applications in the divertor of fusion reactors, where accurate simulations of interactions are crucial for optimizing materials that can withstand harsh conditions while improving the efficiency and longevity of the reactor components.

Current project efforts are focused on developing objective functions to be used within DAKOTA to improve the behavior of our system. To achieve this, we have undertaken detailed calculations of surface, adsorption, and formation energies. These calculations are crucial for understanding the thermodynamic properties and stability of our system, as well as for optimizing the parameters that govern its behavior. By accurately determining these energies, we aim to refine our models and enhance the predictive capabilities of our simulations, ultimately leading to more reliable and efficient designs.

5. Acknowledgments. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly-owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] A. Brodka, *Ewald type summation method for electrostatic interactions in computer simulations of a three-dimensional system periodic in one direction*, Chemical physics letters, 363 (2002), pp. 604–609.
[2] D. Cottrell and P. Tupper, *Energy drift in molecular dynamics simulations*, BIT Numerical Mathematics, 47 (2007), pp. 507–523.
[3] R. Drautz, *Atomic cluster expansion for accurate and transferable interatomic potentials*, Physical Review B, 99 (2019), p. 014104.
[4] G. Dusson, M. Bachmayr, G. Csányi, R. Drautz, S. Etter, C. van Der Oord, and C. Ortner, *Atomic cluster expansion: Completeness, efficiency and stability*, Journal of Computational Physics, 454 (2022), p. 110946.
[5] FitSNAP Development Team, *Fitsnap: A flexible interatomic potential training and application package*, 2023. Available online: `https://fitsnap.github.io/`.
[6] J. M. Goff, C. Sievers, M. A. Wood, and A. P. Thompson, *Permutation-adapted complete and independent basis for atomic cluster expansion descriptors*, Journal of Computational Physics, 510 (2024), p. 113073.
[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
[8] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, 1992.
[9] J. D. Jackson, *Classical Electrodynamics*, Wiley, New York, 3rd ed., 1999.
[10] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient global optimization of expensive black-box functions*, Journal of Global Optimization, 13 (1998), pp. 455–492.
[11] J. W. Lee, R. H. Nilson, J. A. Templeton, S. K. Griffiths, A. Kung, and B. M. Wong, *Comparison of molecular dynamics with classical density functional and poisson–boltzmann theories of the electric double layer in nanochannels*, Journal of chemical theory and computation, 8 (2012), pp. 2012–2022.
[12] Mendeleev, *Mendeleev: A Python package for the periodic table of the elements.* `https://pypi.org/project/mendeleev/0.3.1/`, 2017. Version 0.3.1.
[13] P. Norajitra, S. I. Abdel-Khalik, L. M. Giancarli, T. Ihli, G. Janeschitz, S. Malang, I. V. Mazul, and P. Sardain, *Divertor conceptual designs for a fusion power plant*, Fusion Engineering and Design, 83 (2008), pp. 893–902.

[14] OVITO Team, *OVITO Manual*. `https://www.ovito.org/manual/python/introduction/installation.html`, 2023. Accessed: 2023-09-28.
[15] Sandia National Laboratories, *About dakota*. `https://dakota.sandia.gov/about-dakota/`, 2024. Accessed: 2024-08-08.
[16] ScienceDirect, *Single-objective optimization problem*, ScienceDirect, (2024). Accessed: 2024-08-08.
[17] F. Streitz and J. Mintmire, *Electrostatic potentials for metal-oxide surfaces and interfaces*, Physical Review B, 50 (1994), p. 11996.
[18] United Nations, *Causes and effects of climate change*, 2023. Accessed: 2024-08-08.
[19] U.S. Department of Energy, *Doe explains: Deuterium-tritium fusion fuel*, 2023. Accessed: 2024-08-08.
[20] H. Yu, L. Hong, S. Chen, X. Gong, and H. Xiang, *Capturing long-range interaction with reciprocal space neural network*, arXiv preprint arXiv:2211.16684, (2022).

IN SITU MACHINE LEARNING FOR INTELLIGENT DATA CAPTURE AND EVENT DETECTION

ANDREWS KWASI BOAHEN* AND WARREN L. DAVIS IV†

Abstract. Capturing significant events in detailed, high-fidelity HPC simulations for scientific investigations is becoming more and more challenging because exporting the entire simulation state at each timestep is often impractical. Important phases of events may be overlooked between checkpoints, and transient events might be entirely missed, complicating the detection of these events afterward. In this study, we concentrate on the in situ event detection use case of machine learning to optimize the amount of data saved to disk and the number of events captured. We present the In situ Machine Learning (ISML) framework which is composed from signature, measure and decision building blocks with well defined semantics. We show the capabilities of our framework using a synthetic data with well-known anomalies. We use manifold learning techniques such as Multidimensional Scaling (MDS), Isometric Mapping (Isomap), Locally Linear Embedding (LLE) and t-distributed Stochastic Neighbor Embedding (t-SNE) to create the signatures. Conveniently chosen measure and decision functions were used to detect the anomalies at various timesteps. The anomaly recall and the total percentage of flagged analysis partitions were then compared for the various methods.

1. Introduction. In the era of big data, characterised by the generation and handling of enormous amount of data, scientific research is consistently challenged with the creation of effective tools to handle the complexities inherent to such datasets. According to [6], there is a notable shift from post-simulation analysis to in situ analysis in high performance computing modelling and simulation. Unlike traditional methods that analyze data after a simulation is complete, in situ analysis occurs concurrently with the simulation, utilizing the same resources. This shift is driven by the impracticality of exporting the entire simulation state at each timestep, which often leads to the loss of critical event details between checkpoints and the omission of transient events. While in situ analysis offers a solution, it presents its own challenges, particularly because conventional algorithms that rely on global data access demand excessive communication bandwidth.

We build upon the framework presented by [6] for applying machine learning (ML) to detect events of interest in situ within High-Performance Computing (HPC) simulation data. We define, similar to [6], "events of interest" as local activities in a region that differ significantly from the activities of other regions or timesteps. According to Shead et al (2023, p.56),"ISML is fashioned for parallel and distributed computing environments, where the data represents a space-time domain of interest, with the spatial domain distributed across computing resources and data along the time dimension arriving in a streaming manner."

Consider a scenario where a part of the simulation domain handled by a single processor exhibits behavior that differs significantly from the rest of the simulation domain on other processors. The region that exhibits this difference could be deemed interesting. These differences may persist over time or could just occur between particular timesteps. When changes occur from one timestep to the other at a particular location in the simulation, they are called temporal changes and when these differences are observed between the various regions of the simulation at a particular timestep, they are called spatial events.

At one end, an example of local dynamics persisting over time and different from the remaining simulation domain could be a tropical cyclone that persists over many timesteps in a weather simulation but is geographically localized. At the other end, an unanticipated change across all processors from one timestep to the next could also be considered interesting. An example of this type of event could be simultaneous ignition across an entire

*Michigan State University, boahenan@msu.edu
†Sandia National Laboratories, wldavis@sandia.gov

domain in a combustion simulation.

The in situ detection of these events is crucial for optimizing data capture and analysis in large-scale simulations, where traditional post-hoc analysis becomes impractical due to the sheer volume of data generated. Our contribution to the ISML framework consists of using manifold learning techniques in representing the different analysis blocks formed from the simulation domain and taking advantage of their underlying structure in deciding which events should be considered interesting as they occur during the simulation run.

The paper is outlined in the following sequence. Section 2 presents the Machine Learning framework for intelligent data capture and event detection. In Section 3, a brief overview of manifold learning and its various approaches used in this work to reduce the dimension of the signatures is provided. We then present simulation results and discussion in Section 4, on our synthetic data. Finally, Section 5 contains our conclusion and we added Section 6 to provide suggestions for future work.

2. Machine Learning (ISML) Framework . ISML is designed for automatically detecting spatial and temporal events of interest while running high performance computing simulations [6]. The framework consists of three main buildings blocks: *signatures, measures* and *decisions* as shown in Figure (2.2).

At the initial stage, we consider a simulation domain $\mathcal{S}$ with any number of dimensions. We then assume that $\mathcal{S}$ is split into a set of P analysis partitions. The analysis partitions $p_i, i = 0, ..., P-1$ are a spatially-adjacent subset of mesh points of $\mathcal{S}$ as shown in Figure (2.1).

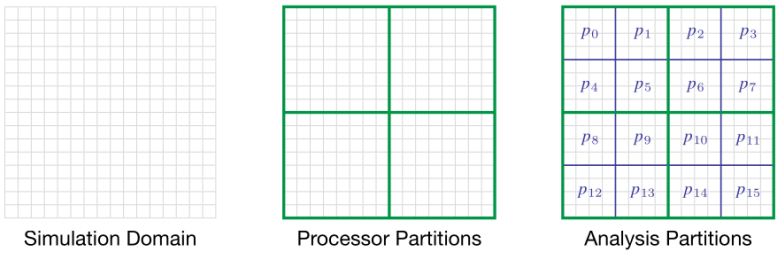


Fig. 2.1: Simulation domain (gray), split across processors (green), and divided into analysis partitions (blue)[6]

The first building block is established by computing a low-dimensional representation of the analysis partitions' content. These representations are called signatures and should contain critical aspects of the observations in the respective partitions. It is worthy to note that signatures are stored in such a way that spatial and temporal changes can be detected by subsequent analysis. The main objective in dividing $\mathcal{S}$ into analysis partitions is in harnessing the power of parallel computing since the analysis partitions can then be allocated to different processors, with the size and number of partitions handled by a processor depending on the nature of the problem being solved[6].

Let $\mathcal{M}$ be the space of all signatures. After forming the signatures, we then define a function $f : \mathcal{M} \to \mathbb{R}$ on the signatures to compute the measures. These measures can be spatial or temporal and are used to detect either changes in each partition across different timesteps or interesting events in $\mathcal{S}$ at a particular timestep. As discussed in [6], comparing measures of a particular partition across various timesteps is a local operation and hence does not require communication among the processors but spatial measures do require com-

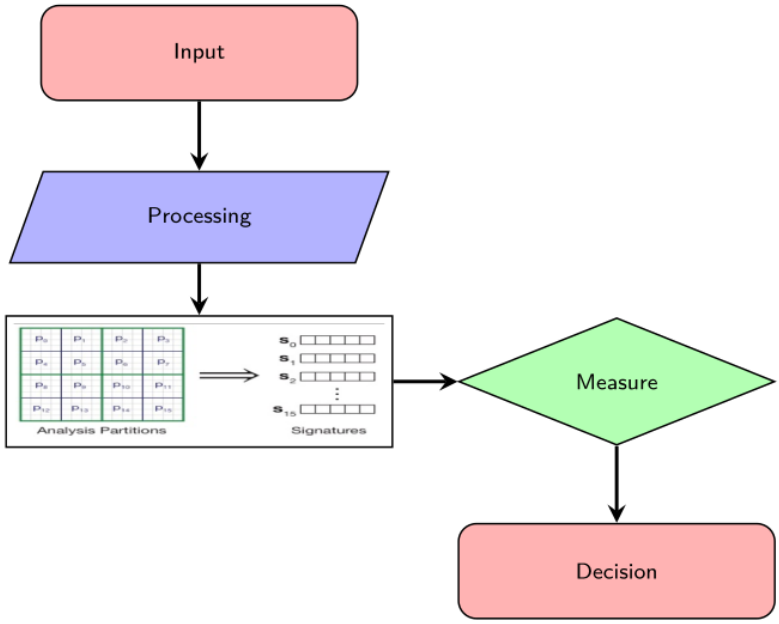


Fig. 2.2: ISML algorithm for event detection

munication among the processors as they compare signatures across analysis partitions to spot spatial events.

Lastly, we define a decision function that maps the measures into the set $\{0, 1\}$, where a value of 1 attributed to an analysis partition indicates the presence of an event of interest and hence the partition will be flagged and saved to disk. It is worth mentioning that, a particular way of defining decision functions is to flag and hence save partitions which value are above a particular threshold.

In this study, we will use a particular class of machine learning techniques for non-linear dimensionality reduction problems called manifold learning to form the signatures. These methods are described in the next section.

3. Manifold Learning. Handling high dimensional data can be quite intensive. From inference to modelling, these tasks can be extremely complex and almost impossible in some cases. One way to deal with such issues is to project the data into a lower dimensional space which preserves the relevant structure of the data since high dimensionality of data is mostly assumed to be artificial and data intrinsically lie on a low dimensional space manifold embedded in a high-dimensional space.

Manifold learning is a non-linear dimensionality reduction technique used to understand and represent these complex, high-dimensional data by mapping it to a lower-dimensional space for visualization, analysis and interpretation. In our framework, we use manifold learning to reduce the dimensionality of the analysis partitions by projecting the data unto a space of reduced dimensions, while aiming to preserve its intrinsic structure. These newly formed partitions, now become the data points for the next step of the algorithm.

There is a wide range of manifold learning techniques which is still being expanded. We then present the manifold learning methods used in this work.

Suppose $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{Y} = \{y_1, ..., y_n\}$ are respectively, a data set on the N-dimensional manifold $\mathcal{M}$ and its projection on a lower dimensional space $\mathcal{D}$. Let d_{ij} be the euclidean distance between y_i and y_j, respective projections of x_i and x_j unto $\mathcal{D}$. The above notations are maintained throughout the paper unless otherwise stated.

3.1. Multidimensional Scaling. Multidimensional scaling (MDS) is a type of manifold learning technique that reduces the dimensionality of data non-linearly by preserving pairwise distances (euclidean) as much as possible on the lower dimensional space.

MDS aims to minimize the stress function defined as:

$$Stress(\mathcal{X}) = \sqrt{\sum_{i \neq j = 1,2,...,n} (d_{ij} - ||x_i - x_j||)^2} \tag{3.1}$$

which incorporates the difference between the distances in the original space (mostly manifold) and their corresponding distances in the lower dimensional space for all possible pairs of points.

3.2. Isomap. Isometric mapping as an unsupervised learning technique is similar to MDS, as it also seeks to discover underlying dimensions that could help describe differences among features by preserving the pairwise geodesic distances as much as possible on the lower dimensional space.[8] It also minimizes a stress function defined as :

$$Stress(\mathcal{X}) = ||g(y_i, y_j) - d(x_i, x_j)||^2, \tag{3.2}$$

where $g : \mathcal{M} \to \mathbb{R}$ and $d : \mathcal{D} \to \mathbb{R}$ are geodesic distance functions defined respectively on the higher and lower dimensional spaces.

3.3. Locally Linear Embedding. Various manifold learning approaches differ by the structure of the high-dimensional manifold being preserved, reflected in the objective function being optimized which reflects the properties of the space being learned. Locally Linear Embedding (LLE) is an unsupervised manifold learning algorithm that computes low dimensional, neighborhood preserving embedding of high dimensional data. LLE uses the local symmetries of linear reconstructions to find nonlinear structure in high dimensional data. [5]

The main assumption of this approach is that each data point and its neighbors lie on or close to a locally linear patch of the manifold. There are two main ways to determine the nearest neighbors:

1. We can use the k nearest neighbors per data points based on the Euclidean distance
2. Neighbors can be defined as all the points in a ball of fixed radius centered around x_i.

This method aims to minimize the reconstruction error :

$$\epsilon(W) = \sum_i ||x_i - \sum_j w_{ij} x_j||^2, \tag{3.3}$$

subject to

$$\sum_j w_{ij} = 1. \tag{3.4}$$

The linear coefficients in W constituting the solution to equation (3.3) are then used to reconstruct each data point from its neighbors in the low dimensional space $\mathcal{D}$ by minimizing the embedding cost function:

$$C_{\mathcal{D}}(Y) = \sum_i ||y_i - \sum_j w_{ij} y_j||^2. \tag{3.5}$$

It is important to note that the optimization in equation (3.3) and (3.5) is over W and Y respectively.

The steps of this algorithm are as follows:

LLE algorithm

1. Compute the neighbors of each data point, x_i
2. Compute the weights w_{ij} that best reconstruct each data point x_i from its neighbors, minimizing the cost in eq. (3.3) by constrained linear fits.
3. Compute the coordinate of each y_i best reconstructed by the linear coefficients w_{ij} minimizing the cost function in (3.5)

Table 3.1: Summary of the LLE algorithm

3.4. t-distributed Stochastic Neighbor Embedding . The final manifold learning method used is the t-distributed Stochastic Neighbor Embedding (t-SNE). This approach very well captures much of the local structures and reveals global structures (such as the presence of clusters at several scales) of the high-dimensional data and is mostly used for visualization [7].

t-SNE's overall aim is to preserve local similarity structure of the data by modelling each high-dimensional data point x_i by a point in $\mathcal{D}$ such that if x_i and x_j are close in $\mathcal{M}$, then their projections should be close in $\mathcal{D}$ and if they are distant in $\mathcal{M}$, then their projections should be distant in $\mathcal{D}$ with high probability.

This approach starts by transforming the euclidean distance between the data points in $\mathcal{X}$ into conditional probabilities as modelled by (3.6).

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}, \tag{3.6}$$

where σ_i is the variance of the Gaussian distribution with mean x_i and $p_{i|i} = 0$.
These $p_{j|i}$'s called similarities represent the conditional probability that x_i would pick x_j as its neighbor if neighbors were chosen proportionately to their probability density under a normal distribution centered at x_i. [7]

Since $p_{j|i}$ and $p_{i|j}$ represent different distributions, their values can be different so we define (3.7), which is symmetric to make the optimization faster.[7]

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2} \tag{3.7}$$

We then model the conditional probability that y_i would pick y_j as its neighbor if neighbors were chosen proportionately to their probability density under a Student t distribution with one degree of freedom via (3.8)

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \tag{3.8}$$

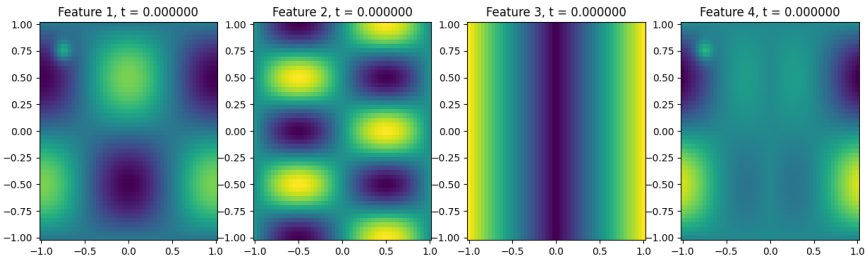
Finally, the low dimension representation $\mathcal{Y}$ in $\mathcal{D}$ is found by minimizing the sum of Kullback-Leibler divergences in (3.9)

$$C = \sum_i KL(P_i||Q_i) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{3.9}$$

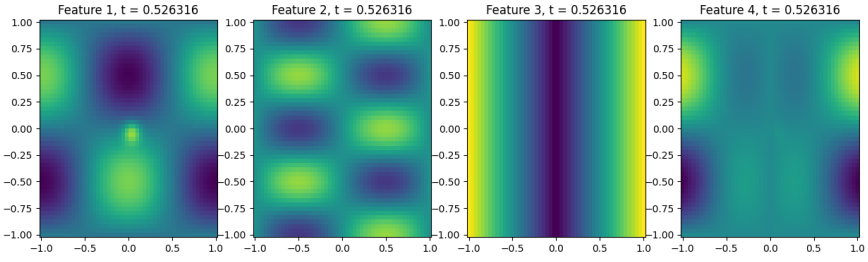
where P_i and Q_i are the respective conditional probability distribution over all other points in $\mathcal{X}$ given x_i and their projections in $\mathcal{Y}$ given y_i.[2]

The signatures formed as functions of the analysis partitions are then projected onto a lower-dimensional space using the non-linear dimensionality reduction methods described in the previous section.

4. Simulation Results & Discussion. In this section, we present the result of our experiments on the simulated data.



(a) t=0s



(b) t=0.526s

Fig. 4.1: Contour map of features 1 through 4 at $t = 0s$ and $t = .526s$.

For the first use of our framework, we created, using Python, a synthetic dataset sitting on a $t \times x \times y$ grid of dimension $20 \times 50 \times 50$, where $t \in (0, 1)$ is the number of timesteps for which the simulation runs and $(x, y) \in [-1, 1]^2$. Our dataset, generated as a NetCDF file, had 4 features represented by a product of elementary functions as follows:

- **Feature 1:** $y_1(t,x,y) = \cos(2\pi t)\cos(\pi x)\sin(\pi y) + \exp\left\{-\frac{[x-(1.5t-0.75)]^2+[y+(1.5t-0.75)]^2}{0.01}\right\}$
- **Feature 2:** $y_2(t,x,y) = \cos(1/2\pi t)\sin(\pi x)\cos(2\pi y)$
- **Feature 3:** $y_3(t,x,y) = |x|$
- **Feature 4:** $y_4(t,x,y) = y_1(t,x,y)y_3(t,x,y)$

The difficulty in event detection problems is usually in the definition of what an interesting event constitutes. Since there is mostly no ground truth or pre-defined anomaly, it is quite arduous to determine which aspects of a simulation should be flagged as anomalous.

We added an anomaly to feature 1 to set the ground truth for the event detection in the context of our framework. This anomaly was modeled by a negative exponential function that depends on the position of a data point on the grid. We decided that when the exponential function is greater than .4, it will indicate the position of an anomaly in the feature space. Feature 4 via its dependency on feature 1 also contains an anomaly. Features 2 and 3 were respectively a plane wave and a linear function independent of time t that did not have any injected anomaly. Figure (4.1) shows a contour map of the features at timestep 0s and 0.526s describing how the anomaly moves across the simulation region between those two timesteps; Figure (4.2) shows an anomalous region detected in the simulation domain at t= 0.158s.

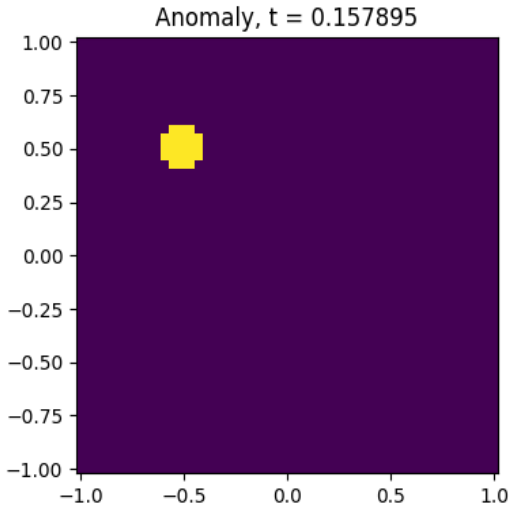


Fig. 4.2: Plot showing an anomaly at time t=0.158s.

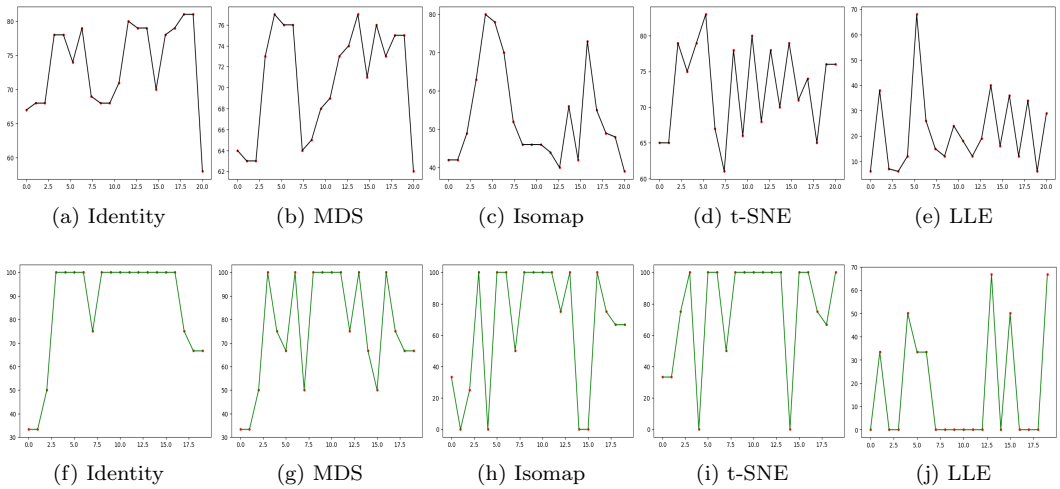
To form the analysis partitions, we use the framework to divide the simulation domain into 5×5 contiguous rectangles. We then have 100 analysis partitions at each of the 20 timesteps. The MinMax scaling function in ISML was used to pre-process the data points in each partition to avoid any scaling discrepancy that may mislead our analysis. Next, we explored with different functions such as the mean per feature and percentile functions to create signatures out of the analysis partitions. The best results in terms of our metrics was obtained with the percentile signature function. These initial signatures are then projected onto a lower-dimensional space using the manifold learning techniques described in Section (3). The final form of our signatures is then measured using the mean square distance

(msd) function which computes the Euclidean distance between one signature and the mean signature for all partitions as defined:

$$msd(s_i) = d_1(s_i, < s >), \tag{4.1}$$

where $< s >= \frac{\sum_{i=0}^{n-1} s_i}{n}$ and d_1 represents the Euclidean distance.

The results of our experiment are shown in Figure (4.3) and Table (4.1). Figure (4.3) shows the amount of data flagged by our framework as anomalous (hence will be saved to the disk) at each of the 20 timesteps, using the different signature functions. We decided to add the identity signature function which treats the initial signatures formed using the percentile function as the final form of the signatures with no projection unto a manifold. The recall value, which simply is the proportion of true anomalous cells (ground truth) contained within each flagged anomalous partition under the various settings is also depicted in Figure (4.3). The recall metric is a crucial way to judge our framework since it provides us with knowledge about the false positives and false negatives discovered using our method. To decide on the most effective combination of functions used in ISML and which parameter values to set for these functions, we would prefer our recall value to be high whilst the amount of data saved to disk is low.



(a) Identity (b) MDS (c) Isomap (d) t-SNE (e) LLE

(f) Identity (g) MDS (h) Isomap (i) t-SNE (j) LLE

Fig. 4.3: % of data flagged as anomalous (top) and recall percentage (bottom) for the various manifold learning techniques

Analyzing the top graphs in Figure (4.3), we can observe that LLE saved the least amount of data to disk at each timestep (between 5% and 70%) whereas MDS and the identity functions saved between 60% and roughly 80% of the data at each timestep.t-SNE

On average, LLE stored 22% of the data to disk which represents roughly a 78% decrease in the storage usage but had the worst performance in terms of recall. In fact, only 18% of our injected ground truth was recovered on average by using LLE to create the signatures, with the most amount of ground truth at a timestep being 68%. The identity function had the highest recall percentage (83.33%) followed by MDS (71.67%) and t-SNE (73.33%). Isomap flagged, on average, a little above half of the number of partitions at each timestep as anomalous and recovered close to 60% of the ground truth. We realised from Table (4.1)

ML methods	% **of flagged partitions**	**Recall**(%)
Identity	73.65	83.33
MDS	70.7	71.67
Isomap	53	58.33
T-SNE	72.75	73.33
LLE	21.8	18.33

Table 4.1: Summary of results obtained using the various manifold learning methods

that there is a trade-off between the recall value and the amount of partitions flagged as anomalous. The more partitions are flagged (hence the more data saved to disk), the more accurate our framework seems to recover the ground truth.

Our experiments revealed that the manifold learning methods based in some sense on Euclidean distances such as t-SNE and MDS performed better in terms of recall than those based on the intrinsic geodesic distance on the manifold like Isomap and LLE. This is not too surprising since our measure function (mean square distance) is based on the Euclidean distance. We conjecture that by using measure functions that incorporate geodesic information, these results will be improved in terms of the recall metric.

5. Conclusion. Our work presents the Machine Learning framework, an approach designed to optimize data storage and improve decision-making while simulation and analysis are being performed concurrently. ISML aims to optimize data capture and event detection in HPC simulations by leveraging machine learning techniques, particularly manifold learning, to intelligently identify and preserve significant events as they occur.

Our approach is based on a well-defined semantic of signatures to measures to decisions. These building blocks of the framework compress the information provided by the data at each timestep, in an intelligent manner, in order to identify anomalous regions in the simulation domain while simulations are still being run.

This paper revealed that storage disk usage can be optimized for some artificially large and high-dimensional datasets which intrinsically lie on a low-dimensional manifold by using manifold learning techniques to create the signatures out of the analysis partitions. These new techniques employed in this setting require new measure functions that can efficiently be employed on the manifold to evaluate the signatures.

6. Future work. We acknowledge some improvements that could be done to the experiments performed with our current framework and propose some ideas worth exploring.

1. Due to the poor performance, in our setup, of the manifold learning techniques that make use of the geodesic distance, we propose a measure function that tries to mimic the mean square distance function on the manifold using the geodesic distance instead of the Euclidean distance. We call this function **geo-msd** and it is defined as:

$$geo - msd(s_i) = d_2(s_i, \mu), \tag{6.1}$$

where d_2 represents the Riemannian distance on $\mathcal{M}$ and

$$\mu = \underset{p \in \mathcal{M}}{\operatorname{argmin}} \left(\frac{1}{n} \sum_{i=0}^{n-1} d_2(s_i, p)^2 \right), \tag{6.2}$$

considered as the mean signature on the manifold $\mathcal{M}$, which is just the sample Fréchet mean.[1]
It measures the geodesic distance between one signature and the mean signature for all partitions on the low-dimensional space. Eq (6.2) may yield a set of solutions on $\mathcal{M}$ as shown in [4][3].

2. The parameters used for the various functions were either set at their default values in Python or hand-tuned to provide a locally optimal result as a function of the recall value. We suggest to thrive to provide an optimal way to select the parameters of the various functions used for the experiment.
3. Taking into account reviewers' comments, we intend to include the relative costs of the different methods used to create the signatures as a metric in deciding the best method by looking at a Pareto plot of the number of correctly identified cells (anomalous or not) vs CPU time.

REFERENCES

[1] P. T. Fletcher, *Statistics on manifolds*, in Riemannian Geometric Statistics in Medical Image Analysis, Academic Press, (2020), pp. 39–74.

[2] G. Hinton and S. Roweis, *Stochastic neighbor embedding*, In Advances in Neural Information Processing Systems, 15 (2022), pp. 833– 840.

[3] N. Miolane, A. L. Brigant, J. Mathe, B. Hou, N. Guigui, Y. Thanwerdas, S. Heyder, O. Peltre, N. Koep, H. Zaatiti, H. Hajri, Y. Cabanes, T. Gerald, P. Chauchat, C. Shewmake, B. Kainz, C. Donnat, S. Holmes, and X. Pennec, *Geomstats: A python package for riemannian geometry in machine learning*, (2020).

[4] X. Pennec, *Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements*, Journal of Mathematical Imaging and Vision, 25 (2006), pp. 127– 154.

[5] L. K. Saul and S. T. Roweis, *An introduction to locally linear embedding*, NYU Computer Science Notes, (2001).

[6] T. M. Shead, I. K. Tezaur, W. L. Davis IV, M. L. Carlson, D. M. Dunlavy, E. J. Parish, P. J. Blonigan, J. Tencer, F. Rizzi, and H. Kolla, *A novel in situ machine learning framework for intelligent data capture and event detection*, in Machine Learning and Its Application to Reacting Flows: ML and Combustion, N. Swaminathan and A. Parente, eds., Springer International Publishing, Cham, (2023), pp. 53–87.

[7] L. van der Maaten and G. Hinton, *Visualizing data using t-sne*, Journal of Machine Learning Research, 9 (2008), pp. 2579–2605.

[8] B. Yousefi, M. Khansari, R. Trask, P. Tallon, C. Carino, A. Afrasiyabi, V. Kundra, L. Ma, L. Ren, K. Farahani, and M. Hershman, *Density-based isometric mapping*, (2024).

LARGE LANGUAGE MODEL ACCURACY ON POST-PROCESSED AI-GENERATED CODE

MARIA CAMILA GAITAN-CARDENAS*, CHRISTOPHER M. SIEFERT†, AND SARAH W. TSAI‡

Abstract. Artificial intelligence (AI) and machine learning are being employed in the creation of code, with large language models (LLMs) proving especially proficient in comprehending and producing natural language. Open weight models, such as Mistral [1], provide advantages such as a small memory size and the ability to operate without human involvement. Nevertheless, there are concerns about the effectiveness of AI code generation. In this work, we evaluated AI-generated code using text-based accuracy metrics, namely BLEU and CodeBLEU. To improve our metrics, we used post-processing techniques to isolate relevant portions of the generated code. Two approaches based on abstract syntax trees (AST) are employed to extract statements from both the reference code and the output code: utilizing semicolons and automating tree-sitter searches. The recursive semi-colon technique retrieves statements by locating the most deeply nested semi-colon in the code that is generated or referenced, whereas the tree-sitter query uses the tree-sitter parser to search for statements. Tree-sitter produces an AST for every code snippet and the reference. Node types are employed to compare the generated code with the reference code in order to assess its appropriateness for evaluation metrics. It is evident that the AST cleaned BLEU outperforms the fundamental BLEU score in the majority of cases.

1. Introduction. As human beings we have the ability to use language, allowing us to express and communicate. This capacity begins to emerge in early childhood and continues to expand throughout a person's lifespan. Machines, without the aid of sophisticated artificial intelligence (AI) algorithms, are unable to comprehend and communicate using human language. For a long time, researchers have been faced with the difficult task of enabling computers to possess the ability to read, write, andcommunicate in a manner similar to humans.

Large language models (LLMs) are a class of models that undergo extensive training on vast quantities of data, enabling them to comprehend and produce natural language and several other forms of content, hence facilitating a diverse array of activities. LLMs have gained widespread recognition because of their pivotal role in popularizing generative AI and serving as a focal point for enterprises seeking to implement artificial intelligence in various business domains and application cases.

AI code generation refers to the application of machine learning (ML) techniques to generate code in response to a user's input. For instance, Google Gemini Code Assist [2] provides developers with the ability to generate and complete code. Code can be produced by following established guidelines, adhering to organizational rules, and even by providing a description using plain language.

As the utilization of generative AI, such as ChatGPT[4], increases, AI code creation becomes increasingly widespread. "Open Weight" models are frequently advantageous for several reasons, including their smaller physical space requirements, ability to run on personal hardware, lack of dependence on human involvement, and provision of information regarding the training data used by certain Language Models (LLMs). This raises the question of whether open weight models are sufficiently effective for generating scientific computer code.

The majority of LLM code generation primarily focuses on Python. However, in this particular case, we examined C++ code utilizing Kokkos[3]. Kokkos is a C++ library that provides a programming model for creating applications that may run efficiently on various high-performance computing systems. It offers abstractions for both parallel code

*Sandia National Laboratories, mcgaita@sandia.gov
†Sandia National Laboratories, csiefer@sandia.gov
‡Sandia National Laboratories, swtsai@sandia.gov

execution and data management. Kokkos is specifically developed to optimize performance on intricate node topologies that have numerous levels of memory hierarchies and diverse execution resources.

We examined text-based accuracy measures from existing research to evaluate the code generated by AI. For this particular case, BLEU[5] and CodeBLEU[6] metrics were used. Both of these are influenced by extraneous or superfluous content, which holds true for the majority of text metrics. Abstract syntax trees were introduced because they offer the capability to eliminate code that does not satisfy certain criteria.

The remainder of this paper is organized as follows: Section 2 discusses the motivation behind the research, Section 3 discusses the methods and terminology used throughout the paper, Section 5 discusses the results of the work and examines and discusses the impact. Finally, we conclude the survey in Section 6 by summarizing the major findings and discuss the remaining issues for future work.

2. BLEU vs. Fluff: A Motivating Example. The BLEU (Bilingual Evaluation Understudy)[5] score measures the similarity between the candidate text and the reference texts. Scores closer to one imply a higher degree of similarity. The BLEU metric offers a comprehensive evaluation of the quality of a model. The BLEU metric has a numerical range of 0 to 1. Only candidate text that is identical to a reference text will achieve a score of 1. Therefore, it is not guaranteed that a human translator will achieve a score of 1. It is crucial to acknowledge that the greater the number of reference texts per sentence, the higher the score will be. Therefore, it is important to exercise caution when making comparisons across assessments that have various numbers of reference texts.

While BLEU has been highly successful in evaluating machine texts and has considerably stimulated research in this field, it is not appropriate for evaluating code synthesis unless the specific characteristics of the programming language are taken into account. A natural language is a language that has organically developed in humans through usage and repetition, whereas code is intentionally created to generate different types of output. The formula for BLEU is:

$$BLEU = BP * exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.1}$$

BP = The brevity penalty is calculated by dividing the length of the machine-generated translation by the length of the smallest source translation.

w_n = The variable w_n represents the weight that is assigned to the accuracy score of the n-gram. Weights are commonly assigned as 1/n, where n represents the number of n-gram sizes employed.

p_n = The variable p_n represents the precision rating for the specific n-gram size.

While BLEU is capable of assessing AI-generated code, its performance is hindered by the inclusion of unnecessary code. In this example, we illustrate how performing post-processing on code created by artificial intelligence might enhance evaluation metrics like BLEU. Consider a human prompt as follows:

Prompt: Create a Kokkos View of doubles of size 10 by 3.

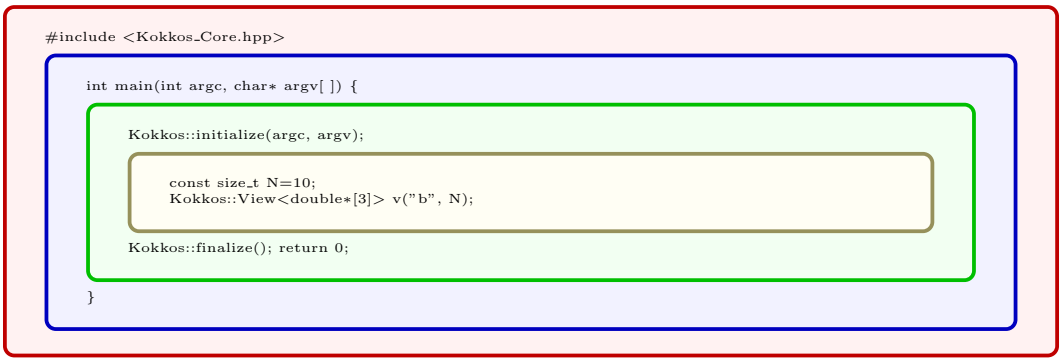


FIGURE 2.1. *A hierarchy of potential LLM predictions with varying BLEU scores base on the amount of included text. BLEU Scores: Red(0.31), Blue(0.36), Green(0.45), Yellow(0.74)*

Below, we can observe the reference (desired output) and a notional LLM-generated prediction. The AI-generated code produces a complete program, which is not necessary to fulfill the human request.

- **Reference:**

```
const size_t N=10;
Kokkos::View<double*[3]> v("b", N);
```

- **Notional Prediction:**

```
#include <Kokkos_Core.hpp>
int main(int argc, char* argv[]) {
  Kokkos::initialize(argc, argv);
  const size_t N=10;
  Kokkos::View<double*[3]> v("b", N);
  Kokkos::finalize();
  return 0;
}
```

Despite producing the correct answer, the BLEU score for the above notional prediction would only be 0.31. We refer to this extraneous code as "fluff." In fact, depending on how much fluff is generated, the BLEU score can be shifted almost arbitrarily. As illustrated in Figure 2.1 all the largest set of generated code (shown by the color red) has BLEU score was 0.31. However, as more and more of that "fluff" is removed, the BLEU score increased by 0.43 points to reach 0.74. The example illustrates that current metrics relying solely on text similarity are insufficient for evaluating generated code due to potential variations in code elements, such as variables.

3. Methodology.

3.1. Tree-Sitter. CodeBLEU takes into account not just the superficial (n-gram) similarity, but also the similarity in syntax and meaning. The n-gram match applies varying weights to distinct n-grams. The syntactic match evaluates the score by comparing sub-trees in the AST. The semantic match measures semantic similarity using the data-flow structure. CodeBLEU is a composite metric that incorporates the original BLEU, the weighted n-gram match, the syntactic AST match, and the semantic data-flow match. Tree-sitter is used by CodeBLEU to do the AST matching.

Tree-sitter[8] is a software application that generates parsers and provides a library for parsing incrementally. The tool can construct a concrete syntax tree for a given source file and effectively modify the syntax tree while the source file is changed. The objective of Tree-sitter is to be:

- Capable of analyzing and interpreting any programming language in a broad manner.
- Efficient enough to analyze each input in a text editor.
- Durable enough to yield valuable outcomes even when syntax errors are present.
- Runtime library, built in pure C, can be integrated in any application without any dependencies.

Tree-sitter involves four primary categories of objects: languages, parsers, syntax trees, and syntax nodes. The terms used in C to refer to these are TSLanguage, TSParser, TSTree, and TSNode.

A TSLanguage is an object that is not easily understood and it specifies how to analyze a specific programming language. Tree-sitter generates the code for each TSLanguage. Several languages are currently accessible in distinct git repositories inside the Tree-sitter GitHub organization. Refer to the following page for instructions on how to generate new languages. A TSParser is an object that maintains its state and can be assigned a TSLanguage. It is used to generate a TSTree from a given source code. A TSTree is a data structure that describes the hierarchical structure of a source code file, capturing the syntax of the entire file. The data structure includes TSNode objects that represent the organization of the source code. Additionally, it has the capability to be modified and utilized for generating a fresh TSTree if there are any alterations made to the source code. A TSNode denotes an individual node within the syntax tree. The program monitors its initial and final places within the source code, as well as its connections to other nodes such as its parent, siblings, and offspring.

1. Parsers
   (a) The parsing capability of Tree-sitter is accessible through C application programming interfaces (APIs). Applications developed in higher-level programming languages can utilize Tree-sitter by employing binding libraries like as node-tree-sitter or the tree-sitter rust crate.
2. Queries
   (a) The syntax highlighting mechanism of Tree-sitter relies on tree queries, which are a versatile framework for matching patterns on the syntax trees of Tree-sitter.
   Query Examples:
      i. **Retrieves the contents of the main function:**
      CPP_LANGUAGE.query (”””(function_definition (compound_statement) @compound)”””)
      ii. **Identifies initialize and finalize statements:**
      CPP_LANGUAGE.query (”””(call_expression) @capture”””)
      iii. **Identifies declaration nodes inside main function:**
      CPP_LANGUAGE. query(”””(compound_statement (declaration) @declarator)”””)

3.2. CodeBLEU. CodeBLEU takes into account the syntactic information by comparing the tree structure, in addition to the sequence-level matching. Programming language, unlike natural language, possesses inherent tree structures, such as the AST. To determine the accuracy of the tree-sitter parsing [7], we first extract all the sub-trees, then compare the extracted sub-trees with the reference sub-trees. CodeBLEU [6] is defined as the

weighted combination of four components: BLEU, $BLEU_{weight}$, $Match_{ast}$, and $Match_{df}$. BLEU is calculated using the standard BLEU method. $BLEU_{weight}$ measures the weighted 1-ngram match with a penalty for mismatching language keyworks that is five times greater than all other tokens. $Match_{ast}$ focuses on the syntactic AST match, utilizing the syntactic information of the code. Each node in the AST represents a specific construct that appears in the source code. $Match_{df}$ considers the semantic dataflow match, taking into account the semantic similarity between the generated code and the reference code. The weighted ngram match and the syntactic AST match are employed for grammatical accuracy, while the semantic data-flow match is utilized for determining logical correctness. The weight (hyper-parameters) are α, β, γ, and δ. The formula for CodeBLEU is:

$$CodeBLEU = \alpha * BLEU + \beta * BLEU_{weight} + \gamma * Match_{ast} + \delta * Match_{df}. \tag{3.1}$$

$\alpha = \beta = \gamma = \delta = 0.25$.

3.3. Drawbacks of CodeBLEU. CodeBLEU is responsive to superfluous code or fluff. The quality of CodeBLEU is negatively impacted by the length of the code. Put simply, there is a negative association between the length of the code and the accuracy of the CodeBLEU metric in reference to the ground truth. The reason for this is likely because lengthier codes typically include a greater number of changes in their syntactic structure. Thus, when the length of codes increases, the accuracy scores of CodeBLEU eventually diminishes and approaches that of BLEU.

4. Proposed Abstract Syntax Tree-based Approach. The issue at hand is the evaluation of AI-generated code by utilizing text-based accuracy measures from previous studies. Our approach applies post-processing to the AI-generated code prior to subjecting it to accuracy measures. By extracting statements from both the reference code and the output code. We employed two distinct techniques to extract the statements: finding semicolons in the code and using tree-sitter queries. The recursive semi-colon technique is used to extract statements by searching for the deepest semi-colon in the generated code or reference code. The tree-sitter query utilizes processes generated by the tree-sitter parser to search for statements (see subsection 3.1). After locating the statement, it is extracted and then compared using abstract syntax trees. After extracting statements, we employ Tree-Sitter to generate the AST for each code snippet and the reference. Next, we employ node types to compare the generated code with the reference code in order to determine its suitability for evaluation metrics. We omit statements that lack a corresponding reference type. Calculate the BLEU metric for the remaining code.

5. Results. This section provides a concise overview of the current discoveries.

The following diagrams illustrate the progress or absence thereof in relation to the approaches above. Figure 5.1 and Figure 5.2 illustrates the visual comparison between the raw code snippets ran through BLEU and the post-processed samples ran through BLEU. CodeBLEU will be employed in subsequent research. As demonstrated, it is evident that the AST cleaned BLEU outperforms the basic BLEU score for the majority of cases. There are a few instances where the raw data samples outperform the post-processed samples, indicating that while the AST cleaned samples may perform better in some scenarios but there is still a need for some adjustments in the proposed approach.

In the subsequent sections, we will examine the most vs the least significant improvements, taking into account the different methodologies employed.

5.1. Semicolon Approach.

Biggest Improvement

Prompt:
A functor to compute the inclusive parallel prefix sum of a Kokkos::View array_type.

Reference:

```
template <typename array_type>
struct InclusiveParallelPrefixSum {
  typedef typename array_type::value_type idx;
  array_type array_sum;
  InclusiveParallelPrefixSum(array_type arr_) : array_sum(arr_) {}

  KOKKOS_INLINE_FUNCTION
  void operator()(const size_t ii, size_t &update, const bool final) const {
    update += array_sum(ii);
    if (final) {
      array_sum(ii) = idx(update);
    }
  }
};
```

Original Output:

```
template <class ViewType>
struct InclusiveScanFunctor {
  ViewType view;

  InclusiveScanFunctor(ViewType view_) : view(view_) {}

  KOKKOS_INLINE_FUNCTION
  void operator()(const int i, int& update, const bool final) const {
    const int val_i = view(i);
    if (final) {
      view(i) = update;
    }
```

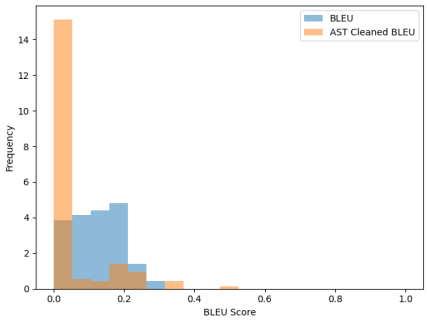


FIGURE 5.1. *Semicolon approach*

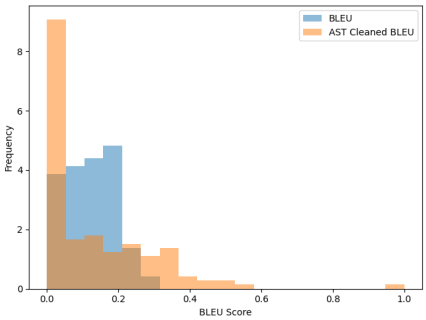


FIGURE 5.2. *Query Approach*

```
    update += val_i;
  }
};

int main(int argc, char** argv) {
  Kokkos::initialize(argc, argv);
  {
    using view_type = Kokkos::View<int*>;
    using host_view_type = view_type::HostMirror;

    view_type view("view");
    host_view_type h_view = Kokkos::create_mirror_view(view);

    const int N = 10;
    view = view_type("view", N);
    h_view = host_view_type("h_view", N);

    for (int i = 0; i < N; i++) {
      h_view(i) = i;
    }

    Kokkos::deep_copy(view, h_view);

    InclusiveScanFunctor<view_type> functor(view);
    Kokkos::parallel_scan("scan", N, functor);

    Kokkos::deep_copy(h_view, view);

    for (int i = 0; i < N; i++) {
      printf("h_view(%i) = %i\n", i, h_view(i));
    }
  }
  Kokkos::finalize();

  return 0;
}
```

Modified Output:

```
 template <class ViewType>
struct InclusiveScanFunctor {
  ViewType view;
  InclusiveScanFunctor(ViewType view_) : view(view_) {}
  KOKKOS_INLINE_FUNCTION
  void operator()(const int i, int& update, const bool final) const {
    const int val_i = view(i);
    if (final) {
      view(i) = update;
    }
```

```
    update += val_i;
  }
};
```

BLEU Score Before Post-Processing: 0.12

BLEU Score After Post-Processing: 0.32

Worst Degradation

Prompt:
Create a complex number of type Kokkos::complex<double> with real part -1 and imaginary part 3.14. Name it "z"

Reference:

```
Kokkos::complex<double> z(-1, 3.14);
```

Original Output:

```
#include <Kokkos_Core.hpp>

int main() {
    Kokkos::initialize();

    Kokkos::complex<double> z(-1, 3.14);

    Kokkos::finalize();
    return 0;
}
```

Modified Output:

The AST string was empty.

BLEU Score Before Post-Processing: 0.28

BLEU Score After Post-Processing: 0.0

5.2. Query Approach.

Biggest Improvement

Prompt:
Define a functor that takes a matrix row-map as a Kokkos view and finds the max degree.

Reference:

```
template <typename Reducer, typename ordinal_t, typename rowmap_t>
struct MaxDegreeFunctor {
  using ReducerVal = typename Reducer::value_type;
  MaxDegreeFunctor(const rowmap_t &rowmap_) : rowmap(rowmap_) {}
  KOKKOS_INLINE_FUNCTION void operator()(ordinal_t i, ReducerVal &lmax) const {
    ordinal_t deg = rowmap(i + 1) - rowmap(i);
    if (deg > lmax) lmax = deg;
  }
  rowmap_t rowmap;
};
```

Original Output:

```
#include <Kokkos_Core.hpp>

template<class ViewType>
struct MaxDegreeFunctor {
  typedef typename ViewType::value_type value_type;
  ViewType row_map;
  value_type max_degree;

  MaxDegreeFunctor(ViewType row_map_) : row_map(row_map_) {}

  KOKKOS_INLINE_FUNCTION
  void operator()(const int i, value_type &update) const {
    update = Kokkos::Experimental::max(update, row_map(i+1) - row_map(i));
  }

  KOKKOS_INLINE_FUNCTION
  void init(value_type &update) const {
    update = 0;
  }

  KOKKOS_INLINE_FUNCTION
  void join(volatile value_type &update, const volatile value_type &input) const {
    update = Kokkos::Experimental::max(update, input);
  }
};

int main() {
  Kokkos::initialize();
  {
```

```
    using view_type = Kokkos::View<int*>;
    view_type row_map("row_map", 5);
    // Fill row_map with some values
    //...

    MaxDegreeFunctor<view_type> functor(row_map);
    Kokkos::parallel_reduce(5, functor);

    std::cout << "Max degree: " << functor.max_degree << std::endl;
  }
  Kokkos::finalize();
  return 0;
}
```

Modified Output:

```
struct MaxDegreeFunctor {
  typedef typename ViewType::value_type value_type;
  ViewType row_map;
  value_type max_degree;
  MaxDegreeFunctor(ViewType row_map_) : row_map(row_map_) {}
  KOKKOS_INLINE_FUNCTION
  void operator()(const int i, value_type &update) const {
    update = Kokkos::Experimental::max(update, row_map(i+1) - row_map(i));
  }
  KOKKOS_INLINE_FUNCTION
  void init(value_type &update) const {
    update = 0;
  }
  KOKKOS_INLINE_FUNCTION
  void join(volatile value_type &update, const volatile value_type &input) const {
    update = Kokkos::Experimental::max(update, input);
  }
};
```

BLEU Score Before Post-Processing: 0.12

BLEU Score After Post-Processing: 0.20

Worst Degradation

Prompt:

Impelement a templated routine for computing the dot product between two Kokkos views x and y: routine code for real Kokkos views, where the type tScalar of the components of x and y is neither Kokkos::complex<float> nor Kokkos::complex<double>. In such situation it is not necessary to take the conjugate of y, and so this candidate routine 2 can just call the candidate routine 1.

Reference:

```
template <class tScalar, class tVecView, class tDevice>
typename std::enable_if< !std::is_same<tScalar, Kokkos::complex<float>>::value && !st
candidateDotProduct2(const tVecView & x, const tVecView & y) {
  return candidateDotProduct1<tScalar, tVecView, tDevice>(x, y);
}
```

Original Output:

```
#include <Kokkos_Core.hpp>

template <typename tScalar, typename tExecutionSpace>
tScalar dot_product(Kokkos::View<tScalar*, tExecutionSpace> x, Kokkos::View<tScalar*
    static_assert(!std::is_same<tScalar, Kokkos::complex<float>>::value &&
                  !std::is_same<tScalar, Kokkos::complex<double>>::value,
                   "tScalar cannot be Kokkos::complex<float> or Kokkos::complex<double

    tScalar result = 0;
    Kokkos::parallel_reduce("dot_product", x.extent(0),
        KOKKOS_LAMBDA(const int& i, tScalar& update) {
            update += x(i) * y(i);
        }, result);
    return result;
}
```

Modified Output:

The AST string was empty.

BLEU Score Before Post-Processing: 0.31

BLEU Score After Post-Processing: 0.00

5.3. Discussion. Based on the results, using the semi-colon technique is more efficient in the post-processing of code snippets, resulting in higher BLEU scores. The query strategy exhibits promise, although it needs father development. Automated post-processing research has demonstrated the superior effectiveness of the semi-colon strategy compared to the query approach. However, the query approach is more effective in scenarios when utilized on a single data point at a time, instead of automating its application over the full dataset. The semicolon technique is valuable for identifying statements in the code that do not involve functions.

6. Conclusion. AI and LLMs play a crucial role in enabling machines to understand and communicate using human language. LLMs, which undergo extensive training on vast data, enable AI to produce natural language and various content forms. AI code generation, a popular application of AI and machine learning techniques, involves generating code in

response to user input. Open weight models, such as Google Gemini Code Assist, are advantageous due to their smaller physical space requirements and lack of human involvement. However, the effectiveness of these models for generating scientific computer code remains a question.

We employed text-based accuracy metrics, such as BLEU and CodeBLEU, to assess the code produced by AI. Our evaluation specifically targeted abstract syntax trees in order to remove any unnecessary or redundant content. Our findings indicate that although there is room for improvement, applying post-processing techniques to ASTs of AI-generated code enhances accuracy measure scores and yields a more efficient response to human requests. As shown, it is clear that the AST cleaned code score performs better than the basic code in most circumstances. There are a few cases where the unprocessed data samples perform better than the post-processed samples, suggesting that although the samples cleaned by AST may perform better in certain situations, some modifications are still necessary in the suggested approach.

In future work, our goal is to enhance the accuracy metrics of C++ AST, particularly by incorporating requests for functions, using statements, typedefs, and other elements.

REFERENCES

[1] M. AI, *Mistral ai - large language models*, 2024.
[2] G. Developers, *Gemini code assist overview — gemini for google cloud*, 2024.
[3] Kokkos, *Kokkos: The programming model*, 2024.
[4] OpenAi, *Chatgpt - instant answers. greater productivity. endless inspiration.*, 2024.
[5] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, *Bleu: a method for automatic evaluation of machine translation*, in Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.
[6] S. Ren, D. Guo, S. Lu, L. Zhou, S. Liu, D. Tang, N. Sundaresan, M. Zhou, A. Blanco, and S. Ma, *Codebleu: a method for automatic evaluation of code synthesis*, 2020.
[7] N. Sobo and R. Winfrey, *Tree-sitter*, 2024.
[8] Tree-Sitter, *Tree-sitter is a parser generator tool and an incremental parsing library*, 2024.

MIXTURE OF NEURAL OPERATOR EXPERTS FOR NONTRIVIAL BOUNDARY CONDITIONS AND MODEL SELECTION

DWYER DEIGHAN*, JONAS ACTOR †, AND RAVI PATEL ‡

Abstract. While Fourier-based neural operators are best suited to learning mappings between functions on periodic domains, several works have introduced techniques for incorporating non trivial boundary conditions. However, all previously introduced methods have restrictions that limit their applicability. In this work, we introduce an alternative approach to imposing boundary conditions inspired by volume penalization from numerical methods and Mixture of Experts (MoE) from machine learning. By introducing competing experts, the approach additionally allows for model selection. To demonstrate the method, we combine a spatially conditioned MoE with the Fourier based, Modal Operator Regression for Physics (MOR-Physics) neural operator and recover a nonlinear operator on a disk. Next, we extract a large eddy simulation (LES) model from direct numerical simulation of channel flow and show the domain decomposition provided by our approach. Finally, we train our LES model with Bayesian variational inference and obtain posterior predictive samples of flow on unseen domains.

1. Introduction. The accuracy of a traditional PDE solver depends on the resolution of its mesh. In practice, a fully resolved simulation is prohibitively expensive for most systems, even on the largest supercomputers. Instead, one must run under-resolved simulations and provide models for subgrid scale dynamics. This is especially true for the Navier-Stokes equations in the turbulent regime. A common solution is to approximate the turbulent dynamics with a statistical approximation such as a large eddy simulation (LES) or Reynolds averaged Navier-Stokes (RANS) model. However, these models are imperfect as they encode various empirical assumptions and hand-tuned approximations.

Neural operators are a new class surrogate models that can learn resolution invariant solution operators for families of PDEs from either high fidelity simulation or experimental data. Resolution invariance is important because it means that if it learns the correct operator, it will work consistently at any mesh resolution. Additionally, neural operators such as Modal Operator Learning for Physics (MOR-Physics) have been shown to be effective for learning important PDEs such as Burger's equation and Kuramoto-Sivashinsky equation and obtaining homogenized PDE models from particle simulations [11, 12].

However, the Fourier transform exhibits Gibbs phenomenon when attempting to model discontinuities and cannot handle non-periodic boundary conditions. The Fourier Neural Operator (FNO) [8] tries to remedy this by adding "bias functions", $Wv(x)$ that operate along the Fourier convolution layers. However, it has been shown [10] that this does not fully alleviate the weakness that FNO has around non-periodic boundary conditions and discontinuities, e.g., shock waves.

Instead, we propose a mixture of experts model where the weighting of experts is determined locally across space. This way the model can choose different experts on two sides of a discontinuity or any non-periodic transition, e.g. zero velocity at the walls for channel flow. In particular, we apply a POU-Net [6] like ensemble method with the MOR-Physics Operator [11] to better learn boundary conditions (POU-MOR-Physics). POU-Net [6] is comparable to Mixture of Experts (MoE) [15] except that it is able to choose different experts locally across a spatial field. This method also has the benefit of facilitating ensemble-based UQ [2] and model selection. To demonstrate the method, we simulatenously learn the boundary conditions and a LES model for channel flow using the Johns Hopkins Turbulence Database (JHTDB) [7, 5, 13].

*University at Buffalo, The State University of New York, dwyerdei@buffalo.edu
†Sandia National Laboratories, jaactor@sandia.gov
‡Sandia National Laboratories, rgpatel@sandia.gov

Because we are limited to a single DNS, we embed a simple forward-Euler solver in our model for training and implement Uncertainty Quantification (UQ) to provide a range of predictions outside the training data. The integration of the PDE solver with the learned correction serves a purpose similar to data augmentation: enabling higher-quality predictions despite the limited data. The UQ is instrumental in identifying the support of the dataset at prediction time, particularly for detecting out-of-distribution (OOD) queries, and for modeling the notoriously high aleatoric uncertainty in turbulence.

The UQ arises from a hybrid approach utilizing the diversity among expert models and Mean-Field Variational Inference (MFVI). It has been demonstrated in [3] that "sub-space method" such as MFVI and ensemble methods can serve complementary roles, each alleviating the other's weaknesses. Additionally it was suggested in [3] that ensemble methods could be potentially made more efficient for UQ by explicitly enforcing ensemble member diversity. We believe our method indirectly tests this suggestion by forcing our experts to specialize.

Our model operates by applying a learned correction after each timestep of the regular forward-Euler solver. This correction is designed to capture the sub-grid dynamics that the low-resolution solver fails to resolve, effectively enhancing its accuracy to better match the DNS data. By integrating the learned correction into the forward-Euler method at every timestep, the model adjusts the solver's output, compensating for the missing fine-scale dynamics. In contrast to papers like [9] and [14], we apply our PDE constraints by construction rather than by penalty of the model.

To improve the stability of the learned sub-grid dynamics operator, we employed several auto-regressive time steps during training, following methodologies similar to those proposed by Bengio et al. [1] (but without the curriculum). Statistical auto-regressive time series models often suffer from exponentially growing errors because the model's flawed outputs feed back into its inputs, compounding errors over multiple time steps. By exposing the model to this process during training—taking several auto-regressive steps before each optimization step—the model learns to correct its own compounding errors, mitigating the problem during prediction time [1].

2. Methods.

2.1. Problem descriptions.

We demonstrate our method with two numerical examples. This section describes the problem formulations. In the first example, we learn a nonlinear operator for functions on a disk. In the second example, we learn an LES model for turbulence for channel flow.

2.1.1. 2D synthetic data.

Our first dataset consists of synthetically generated pairs of functions, (u_i, v_i), on the unit disk that vanish on the boundary, $\Omega = \{(x,y) : x^2 + y^2 \leq 1\}$. We generate u_i by sampling a Gaussian process. We construct a regular grid of points, G, over a box in which Ω is embedded, $\mathcal{B}$, and compute the point evaluations of a function $\hat{u}$ on the grid as,

$$\begin{aligned} &z_i \sim \mathcal{GP}(0, K) \\ &\hat{u}_i(x', y') = z_i(x', y') | \{z_i(x,y) = 0 : (x,y) \notin \Omega\} \quad (x,y), (x',y') \in G \end{aligned} \tag{2.1}$$

where K is a symmetric positive definite kernel, $K : \mathcal{B} \times \mathcal{B} \to \mathbb{R}$. We compute $\hat{v}_i$ by taking the second order finite difference approximation of the Laplacian of $\hat{u}_i^2$.

We obtain a mapping, $u_i \mapsto v_i$, from this synthetic data by solving the least squares problem,

$$\min_{\theta} \sum_i ||\hat{\mathcal{P}}(\hat{u}_i; \theta) - \hat{v}_i||^2_{\ell_2(G)} \tag{2.2}$$

where $\hat{\mathcal{P}}$ is the model operator and θ are the parameters of $\hat{\mathcal{P}}$. We discuss the parameterization of $\hat{\mathcal{P}}$ in Section 2.3

Since $\hat{u}_i$ and $\hat{v}_i$ vanish at the boundaries of $\mathcal{B}$, we glue the opposite sides of the box together and treat them as periodic functions. We arrive at (u_i, v_i) by taking the restriction of $\hat{u}_i$ and $\hat{v}_i$ to the domain, Ω. We construct $\mathcal{P}(u)$ by extending u to the domain $\mathcal{B}$ where it evaluates to zero outside of Ω. $\hat{P}$ maps this extension to another function on $\mathcal{B}$ as discussed above. v is the restriction of this action to Ω. The model operator with the restriction and extension takes the form, $\mathcal{P} : u \overset{\text{extend}}{\mapsto} \hat{u} \overset{\hat{\mathcal{P}}}{\mapsto} \hat{v} \overset{\text{restrict}}{\mapsto} v$

2.1.2. Large Eddy Simulation Modeling. For our next example, we obtain a large eddy simulation (LES) model from direct numerical simulation (DNS) of turbulent channel flow. The Navier-Stokes equations are,

$$\begin{aligned} &\partial_t u + \nabla u \otimes u = -\frac{1}{\rho}\nabla p + \nu \nabla^2 u \qquad && x \in \Omega \\ &\nabla \cdot u = 0 && x \in \Omega \\ &u = 0 && x \in \partial\Omega \end{aligned} \tag{2.3}$$

(2.3) is typically expensive to integrate because real systems often contain a large range of spatio-temporal scales that must be resolved. The LES equations are obtained by applying a low pass spatio-temporal filter to the Navier-Stokes equations,

$$\begin{aligned} &\partial_t \tilde{u} + \nabla \tilde{u} \otimes \tilde{u} = -\frac{1}{\rho}\nabla \tilde{p} + \nu \nabla^2 \tilde{u} - \nabla \tau \qquad && x \in \Omega \\ &\nabla \cdot \tilde{u} = 0 && x \in \Omega \\ &\tilde{u} = 0 && x \in \partial\Omega \end{aligned} \tag{2.4}$$

where $\tau = \widetilde{u \otimes u} - \tilde{u} \otimes \tilde{u}$ is the residual stress tensor. Since the small scale features have been filtered out, (2.4), can be much cheaper to numerically integrate than (2.3). However, τ is an unclosed term that must be modeled. Traditionally, one uses a combination of intuition and analysis to arrive at a simple model with a few parameters that can be fitted to DNS simulation or experiments. Here, we will use POU-MOR-Physics to provide the closure.

To obtain our model, we will first consider LES in a triply periodic domain.

$$\begin{aligned} &\partial_t \tilde{u} + \nabla \tilde{u} \otimes \tilde{u} = -\frac{1}{\rho}\nabla \tilde{p} + \nu \nabla^2 \tilde{u} - \nabla \tau \qquad && x \in \Omega \\ &\nabla \cdot \tilde{u} = 0 && x \in \Omega \end{aligned} \tag{2.5}$$

We use the Chorin projection method to eliminate the pressure term and the explicit Euler time integrator to obtain an evolution operator,

$$\begin{aligned} \hat{u}^{n+1} &= \tilde{u}^n - \mathcal{F}^{-1}\left(i\kappa \cdot \mathcal{F}\tilde{u}^n \otimes \tilde{u}^n + \frac{i\kappa}{||\kappa||_2^2}(\kappa \otimes \kappa) : \mathcal{F}\tilde{u}^n \otimes \tilde{u}^n - ||\kappa||_2^2 \mathcal{F}\tilde{u}^n \right) \\ &= \text{LES}(\tilde{u}^n) \end{aligned} \tag{2.6}$$

where $\mathcal{F}$ is the Fourier transform and κ is the wave vector.

This update operator does not include the boundary conditions or the missing sub-grid scale physics. We propose to model these as a POU-MOR-Physics correction to the action above and obtain,

$$\tilde{u}^{n+1} = \mathcal{P}(\text{LES}(\tilde{u}^n); \theta) = \mathcal{U}_\theta(\tilde{u}^n) \tag{2.7}$$

where $\mathcal{P}$ is the POU-MOR-Physics operator discussed in Section 2.3. Given a time-series for the velocity field evolution from DNS, $\{u_d^n\}^{n=0,\ldots,N}$, on a regular grid, G, we solve the least squares problem,

$$\min_{\theta} \sum_{n=1}^{N} \left\| \mathcal{U}_\theta^n \tilde{u}_d^0 - \tilde{u}_d^n \right\|_{\ell_2(G)}^2 \tag{2.8}$$

to obtain an LES model. Note that we have applied the low-pass filtering operator to the data. In practice, (2.8) is not computationally tractable, so we implement a multiple shooting strategy and obtain the optimization problem,

$$\min_{\theta} \sum_{n=1}^{N-P} \sum_{p=1}^{P} \left\| \mathcal{U}_\theta^p \tilde{u}_d^n - \tilde{u}_d^{n+p} \right\|_{\ell_2(G)}^2 \tag{2.9}$$

where P is a hyperparameter.

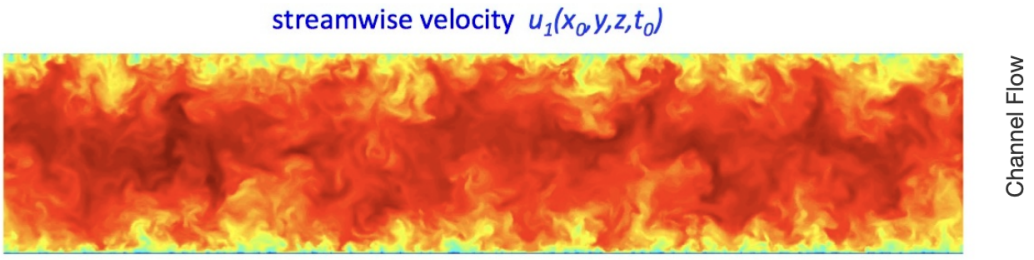


FIG. 2.1. *John Hopkins Turbulence Dataset: Channel Flow DNS*

Our target application is extracting an LES model from DNS provided by the JHTDB. The simulation data is obtained from a channel flow problem with no-slip boundary conditions (BCs) on the top and bottom of the flow, and periodic BCs on the left, right, front and back.

We greatly sub-sample the DNS data spatially (see Table 2.2), we keep the full resolution in the time dimension. This was motivated by the need to have a larger training dataset. The spatial sub-sampling is performed after applying a box filter to the DNS data.

Parameter	**Value**
Viscosity	$v = 5 \times 10^{-3}$
Friction velocity Reynolds number	$Re_\tau \sim 1000$
Domain Length	$8\pi \times 2 \times 3\pi$
Grid Size	$2048 \times 512 \times 1536$

TABLE 2.1

Simulation Data Parameters of 3d DNS channel flow data from the JHTDB. This table has the parameters of the DNS simulation.

Parameter	Value
Spatial Stride	$s_x = s_y = s_z = 20$
Sub-Sampled Dimensions	$103 \times 26 \times 77$
Time Dimension	$t = 4000$
Box Filter Dimension	$b = 20$

TABLE 2.2

Post Processing Parameters. For post-processing steps we applied box filtering and then spatial *stride (without time stride).*

2.2. Scaling. The JHTDB channel flow problem [5], characterized by three spatial dimensions plus a recursive time dimension, presents significant computational challenges that demand careful resource management. To mitigate the substantial memory requirements, we utilized the real Fast Fourier Transform (rFFT) within the MOR-Physics Operator, effectively conserving memory without compromising performance.

To handle the computational load efficiently, we employ 20 A100 GPUs alongside the Fully-Sharded Data Parallel (FSDP) strategy [17]. This approach builds upon Data-Distributed Parallelism by dividing the data batch across multiple GPUs and averaging the independent gradients after each iteration. According to the linear scaling rule from [4], scaling the learning rate proportionally with the batch size can expedite training. We implemented this by adjusting the learning rate in line with the increased batch size distributed over the GPUs.

FSDP further reduces the memory burden on each GPU by sharding the model parameters, optimizer states, and gradients across all available GPUs. It broadcasts the necessary shards just in time for forward and backward passes, overlapping communication with computation to eliminate communication overhead. This strategy not only conserves memory but also enhances computational efficiency.

Following the methodologies outlined in [16] and [4], we introduced a warm-up phase for the learning rate to approach linear scaling effectively. This warm-up period is crucial to avoid suboptimal local minima early in the training process, as the linear scaling rule may not be effective when the model parameters are changing rapidly at the outset.

Adhering to the linear scaling rule from [4], we set the batch size to batch_size $= n$ and the learning rate to learning_rate $= 0.00025n$, where n represents the number of GPUs utilized. Although memory constraints necessitated a smaller per-GPU batch size, we found that combining this approach with gradient clipping yielded effective training results.

2.3. Model Design. We construct our model by combining a spatially conditioned POU network and neural operators,

$$(\mathcal{P} \circ a)(x) = \sum_i G_i(x)(\mathcal{N}_i \circ a)(x) \tag{2.10}$$

This section describes the components of this model, the gating network G and the neural operators, $\mathcal{N}_i$.

2.3.1. MOR-Physics Operator. We use a modified version of the MOR-Physics Operator presented in [11, 12] to construct $\mathcal{N}_i$. Input and output channels are necessary to deal with the 3d input and output velocity components, but hidden channels improve performance, so we used them throughout the model.

We follow the notational conventions *similar* to those from the FNO paper [8]. Also all the equations below ignore the batch dimension.

DEFINITION 2.1. *Given the Banach spaces A, V, U:*

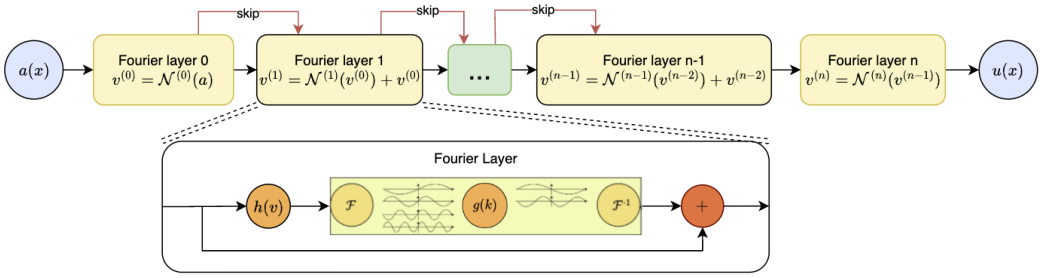


FIG. 2.2. ***MOR Operator Diagram:*** *It is important to recall that $g(k)$ is actually implemented with a tensor so we need to truncate higher modes of the input (or sometimes g(k)). Also the arrows denote composition, and h(v) is a learned (local) activation function.*

- *The neural operator $N : A \to U$ is defined by $N(a) = u$ for all $a \in A$ and $u \in U$.*
- *$\forall v^{(l)} \in V$, the function $v^{(l)} : \mathbb{R}^d \to \mathbb{R}^{d_v^{(l)}}$ represents the intermediate state of $u \in U$ at hidden layer l.*
- *$\forall k \in \mathbb{R}^d$, the learned weights of a MOR Operator layer in the Fourier domain are given by $g^{(l)}(k) \in \mathbb{C}^{d_v^{(l)} \times d_v^{(l-1)}}$.*
  - *$d :=$dimensionality of the problem (e.g. 1d, 2d, etc...)*
  - *$d_v^{(l-1)}, d_v^{(l)} :=$ number of input and output channels for layer l respectively.*
- *The non-linearity represented as $h^{(l)}(x)$ is a learned local activation function (implemented with a neural network).*

Note that $g(k)$ *is implemented as a tensor* due to its use with the FFT. We truncate the higher modes of either the $g^{(l)}(k)$ or $\mathcal{F}(v^{(l)})(k)$ tensors (which ever has more in a given dimension) so that their shapes are made compatible. This effectively results in a low pass filter; more details can be seen in [11].

MOR Operator Layer Equation:

$$\mathcal{N}^{(l+1)}(v^{(l)})(x) = \mathcal{F}^{-1}(g^{(l+1)} \cdot \mathcal{F}(h^{(l+1)} \circ v^{(l)}))(x) \tag{2.11}$$

Note that the multiplication operation in the operator 2.11 is a matrix multiplication; recall 2.1 $g^{(l+1)}(k) \in \mathbb{C}^{d_v^{(l+1)} \times d_v^{(l)}}$ and $\mathcal{F}(h^{(l+1)} \circ v^{(l+1)})(k) \in \mathbb{R}^{d_v^{(l)}}$. This is comparable to how channel dimensions are handled in Convolutional Neural Networks (CNNs).

MOR Operator Layer + Skip Connection Equation:

$$\tilde{\mathcal{N}}^{(l+1)}(v^{(l)}) = \mathcal{N}^{(l+1)}(v^{(l)}) + v^{(l)} \tag{2.12}$$

We used skip connections throughout the hidden layers. But due to changing channel dimensions it cannot them apply to the first or last layer.

MOR Operator (aka Expert) Equation:

$$\begin{aligned} \tilde{\mathcal{N}}^{(l+1)}(v^{(l)}) &= \mathcal{N}^{(l+1)}(v^{(l)}) + v^{(l)} \text{ (skip connection layer)} \\ N(a) &= (\mathcal{N}^{(0)} \circ \tilde{\mathcal{N}}^{(1)} \circ ... \circ \tilde{\mathcal{N}}^{(n-1)} \circ \mathcal{N}^{(n)})(a) = u, \text{ s.t.} \\ d_v^{(l)} &= d_{hidden} : 0 < \forall l < n \text{ (channel dimensions)} \\ d_v^{(0)} &= d_{in}, d_v^{(n)} = d_{out} \text{ (channel dimensions)} \end{aligned} \tag{2.13}$$

Gating Network Equation:

$$G(x) : R^d \to R^{\tilde{n}}, \text{ s.t. } \mathcal{N}_i \in \{\mathcal{N}_j\}_{j=1}^{\tilde{n}}$$
$$\tilde{G}(x) = \frac{e^{G(x)}}{\sum_{i=1}^{\tilde{n}} e^{G_i(x)}} \tag{2.14}$$

The gating network takes the coordinates as inputs and produces a Softmax weight vector that is used to get a weighted combination of expert (aka N_i) outputs at location $x \in R^d$. The Gating network does *not* take a as input, it only uses the location x which is sufficient to partition the space for different experts.

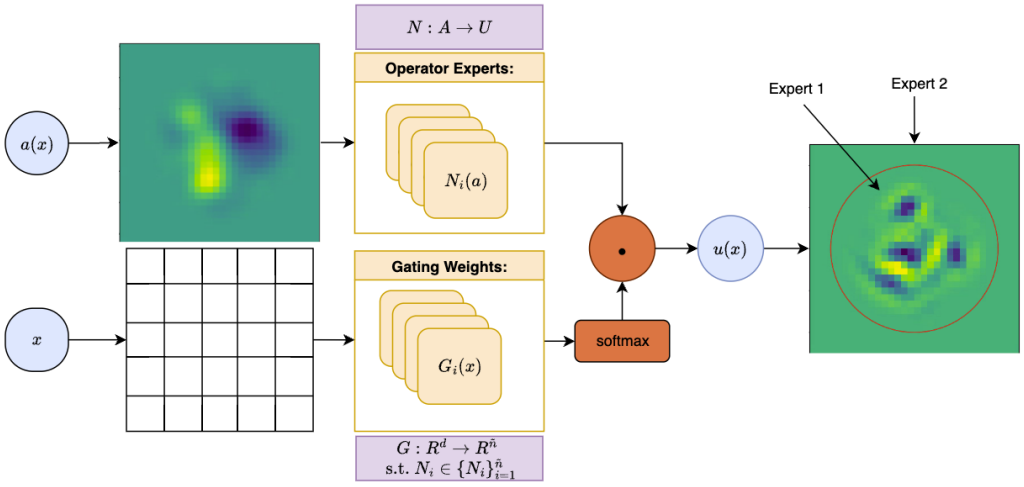


FIG. 2.3. ***Mixture of Experts Diagram:*** *Unlike regular Mixture of Experts [15] the weights are spatially localized and the gating network $G_i(x)$ doesn't see the actual input ($a(x)$).*

3. Results.

3.1. 2D MOR-Physics Operator Problem. We generate synthetic 2d data consisting of pairs of functions on the unit disk that vanish on the boundary, as discussed in Section 2.1.1 and train our mixture of experts model to fit the operator, $\mathcal{P}(a) \approx u$. The data have activity in the center but have zero boundary conditions, $\forall x \in \partial D, a(x) = u(x) = 0$. This process generates pairs $(\mathbf{a}, \mathbf{u})$ that are suitable for training models and conducting further analysis because they are non-periodic and need different experts to fit different regions.

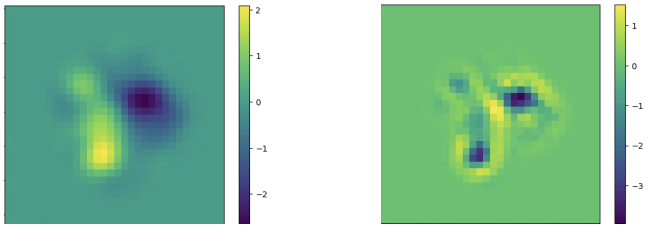


FIG. 3.1. *(Left) input and (Right) output functions from 2d synthetic data.*

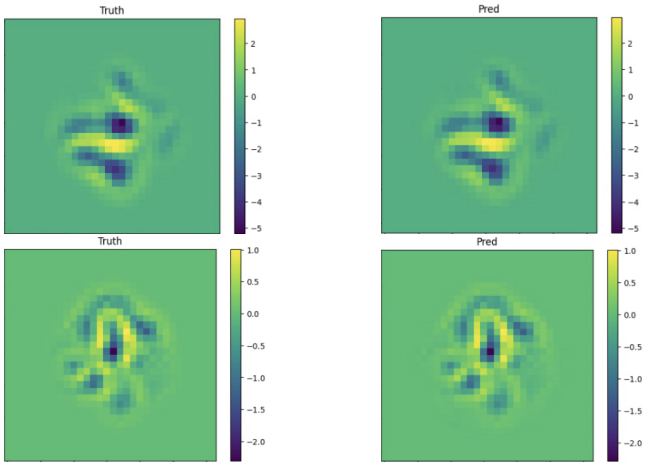


FIG. 3.2. *(Left) Test data and (Right) prediction 2d synthetic exemplar. Top and Bottom are results from different test data samples.*

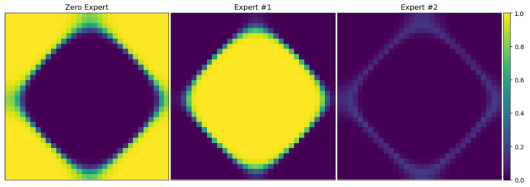


FIG. 3.3. *2d Expert Partitions: We investigated the gating network's partitioning of the problem to see if it choose different experts for the domain and boundary layer. The results confirm our hypothesis that it would learn these partitions and use the special "Zero Expert" near the boundary. It seems the expert #2 is mostly used between the other two, although it primarily relies on the other two overall.*

As can be seen in Figure 3.2 the model was able to flawlessly fit the problem. In fact it achieved a validation $R^2 > 99.999\%$ implying it explained that percent of the variance, which is essentially a perfect fit. And as seen in 3.3 it was clearly able to assign the "Zero Expert" $Z(a) = 0, Z : A \to U$ to the boundaries as was intended.

3.2. Large Eddy Simulation. Figures 3.6,3.9, and 3.12 compare the x, y, and z components of velocity between our LES model and the JHTDB data. We find our model is able to closely match the true DNS evolution. It is evident from all these figures that the learned simulation is able to largely *avoid divergence*, because it is clearly able to reproduce late simulation artifacts found in the DNS. This is at least partially attributable to the extra auto-regressive timesteps taken during training to stabilize prediction (effectively teaching it to correct its own errors). However, it is important not to forget that this model also achieved validation $R^2 = 98.81\%$ (i.e. it explains 98.81% of the variance). The only catch here is that the model was tasked with reproducing the simulation it had been trained on (since there was only one simulation from JHTDB with these settings). Even so, this is impressive as 20% of the simulation was held out from the training data.

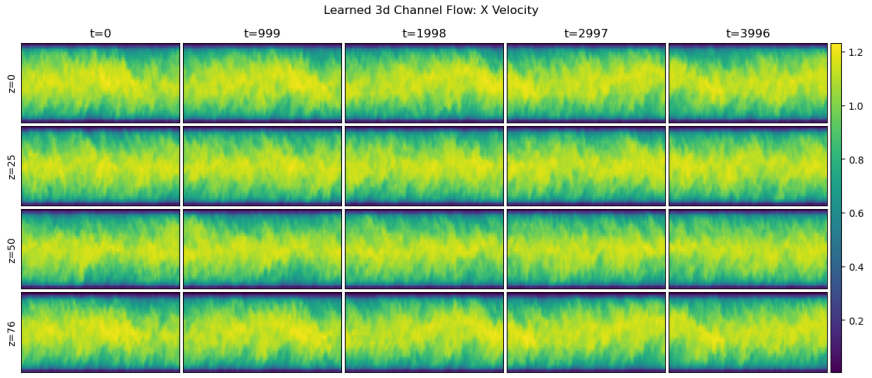


FIG. 3.4. *Learned Simulation: X Velocity*

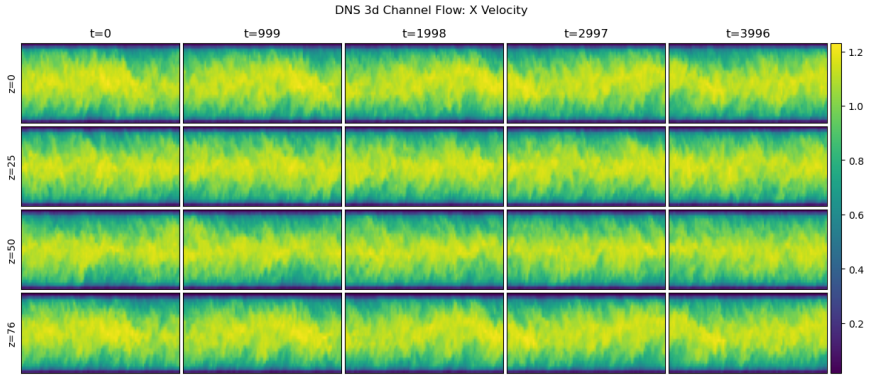


FIG. 3.5. *DNS: X Velocity*

FIG. 3.6. *Comparison of X velocity field from learned simulation vs DNS from JHTDB.*

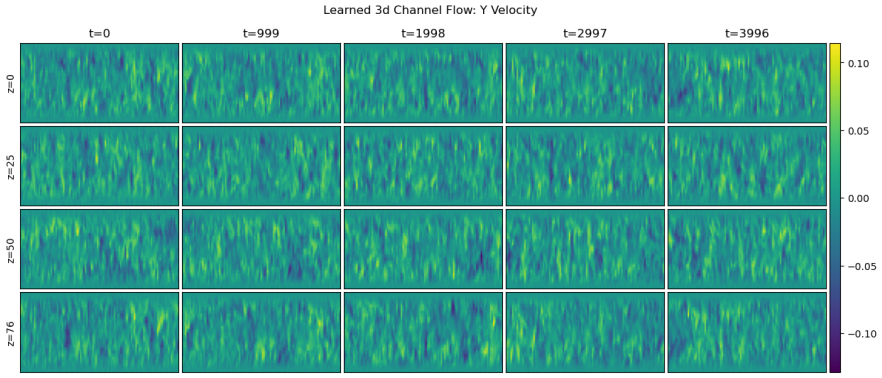


FIG. 3.7. *Learned Simulation: Y Velocity*

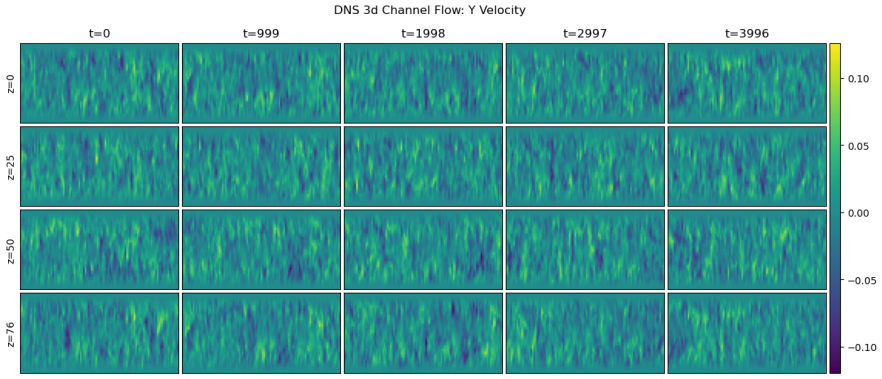


FIG. 3.8. *DNS: Y Velocity*

FIG. 3.9. *Comparison of Y velocity field from learned simulation vs DNS from JHTDB.*

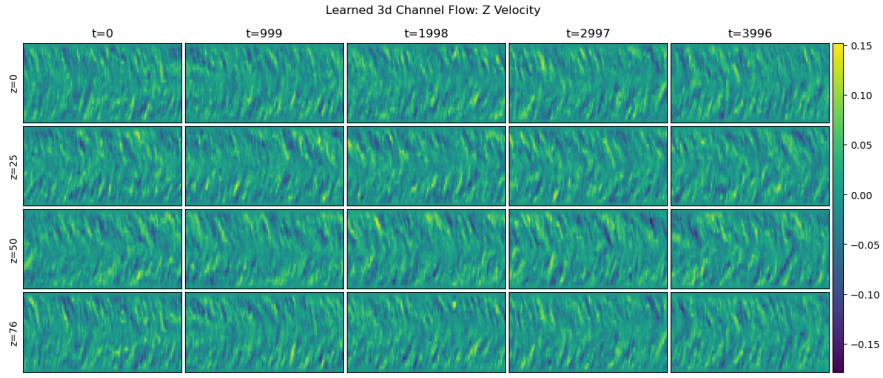


FIG. 3.10. *Learned Simulation: Y Velocity*

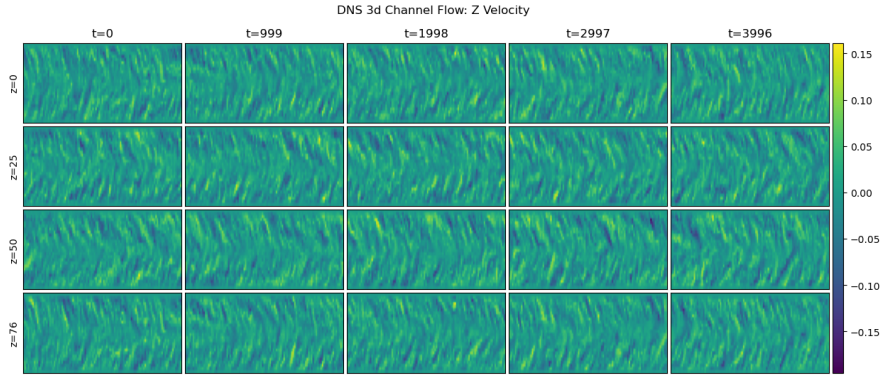


FIG. 3.11. *DNS: Y Velocity*

FIG. 3.12. *Comparison of Y velocity field from learned simulation vs DNS from JHTDB.*

3.3. Model Expert Partitions. We inspect the gating network's partitioning of the domain in order to investigate whether it was partitioning the space differently between the domain and boundary layer. And in particular, if it happened to learn how to use the Zero Expert we included. The partitions are shown in 3.13.

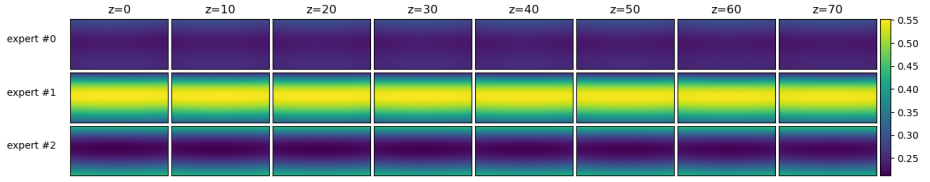


FIG. 3.13. *3d Expert Partitions: Here "expert #0" is the Zero Expert. While this version of the model didn't learn to utilize the Zero Expert as we had hoped, it did definitely learn to partition the space based on the boundary layer. It can be seen that it used $\mathcal{N}_1$ for predictions inside the domain and $\mathcal{N}_2$ for predictions inside the boundary layer. Furthermore the transition from $\mathcal{N}_1$ to $\mathcal{N}_2$ is continuous roughly approximating the strength of the velocity at those points in the simulation.*

3.4. Model Training Metrics. In 3.16 and 3.19 we show the results when we apply our Mixture of Experts MOR-Physics Operator method to the full 3d JHTDB channel flow problem. In particular, we show the training and validation metrics of the best model we trained on the Kahuna supercomputer. Notably there seems to be almost no discrepancy between the training and validation performance which indicates good generalization.

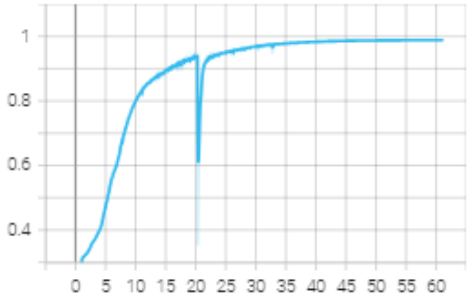


FIG. 3.14. *Train R^2 vs Step*

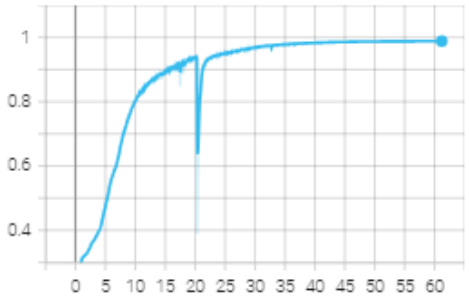


FIG. 3.15. *Validation R^2 vs Step*

FIG. 3.16. *The metric is defined as: $R^2(\tilde{y}, y) = 1 - \sum_i (\tilde{y}_i - y_i)^2 \overline{V}ar[y] \in (-\infty, 1]$ $R^2 = 1$ implies perfect prediction, $R^2 = 0$ indicates your model is no better than $f(x) = E[y]$, and $R^2 < 0$ is even worse. These $R^2 \approx 98.8\%$ values are really good! It means that the regression problem is 98.8% solved. Or strictly speaking that the regression model explains 98.8% of the variance in the y variable.*

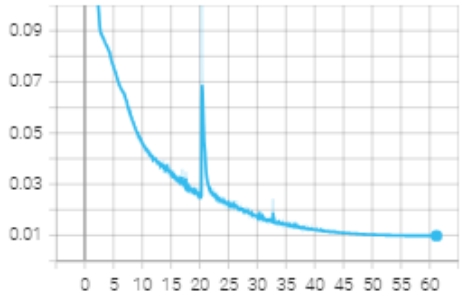


FIG. 3.17. *Train wMAPE vs Step*

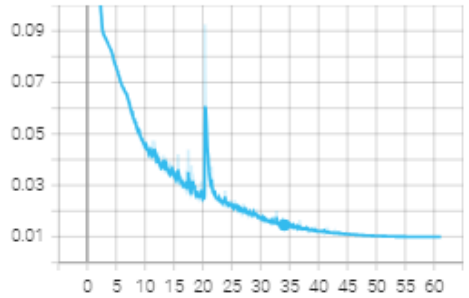


FIG. 3.18. *Validation wMAPE vs Step*

FIG. 3.19. *The metric is defined as: $wMAPE(\tilde{y}, y) = \frac{\sum_{i=1}^{n} |\tilde{y}_i - y_i|}{\sum_{i=1}^{n} |y_i|} \in [0, 1]$ wMAPE (Weighted Mean Absolute Percentage Error) measures the weighted percentage difference between predicted and actual values. A wMAPE of 0% indicates perfect prediction, while higher values reflect larger deviations from the actual values. In these plots wMAPE steadily decreases, eventually converging around 0.01 after 92,000 steps. This final value indicates that the model's predictions are, on average, within 1% of the true values, representing a very good result!*

4. Conclusion. In this work, we addressed the challenge of learning non-periodic boundary conditions and discontinuities in PDE simulations using a Mixture of Experts (MoE) approach, specifically applied to the MOR-Physics neural operator. The motivation for this work stems from the limitations of traditional Fourier-based PDE solvers, which struggle with non-periodic boundary conditions, such as those found in Navier-Stokes simulations. By integrating a forward Euler PDE solver and applying a correction operator at each timestep, we were able to enhance solver accuracy. Our methods were tested on both 2D synthetic data and LES for 3D channel flow, demonstrating promising results. The

model was highly successful in the 2D case, where it flawlessly captured the zero boundary conditions. Furthermore, the 3D channel flow problem yielded auspicious results with training and validation metrics showing that the model explained 98.8% of the variance in the target variable and had a final wMAPE of around 1%, indicating a very strong fit to the data. Similarly in the 3d case the model's learned simulation reproduced the original simulation so well it was almost indistinguishable from the original.

5. Future Work. Future efforts will focus on expanding this research in a few areas:

- **Uncertainty Quantification (UQ) Using a Mixture of Experts Ensemble + VI**: We aim to explore the uncertainty quantification capabilities of our Mixture of Experts method by validating the UQ against known results and determining the model's confidence in its predictions.
- Energy Spectrum figures to determine if the flow statistics match between the learned simulation and the actual DNS data.
- **Publication**: The results and findings from this ongoing work are being prepared for publication, which will provide a more comprehensive evaluation of our approach and its implications for PDE modeling and neural operators.

REFERENCES

[1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, *Scheduled sampling for sequence prediction with recurrent neural networks*, Advances in neural information processing systems, 28 (2015).

[2] T. Fan, N. Trask, M. D'Elia, and E. Darve, *Probabilistic partition of unity networks for high-dimensional regression problems*, International Journal for Numerical Methods in Engineering, 124 (2023), pp. 2215–2236.

[3] S. Fort, H. Hu, and B. Lakshminarayanan, *Deep ensembles: A loss landscape perspective*, arXiv preprint arXiv:1912.02757, (2019).

[4] P. Goyal, *Accurate, large minibatch sg d: training imagenet in 1 hour*, arXiv preprint arXiv:1706.02677, (2017).

[5] J. Graham, K. Kanov, X. Yang, M. Lee, N. Malaya, C. Lalescu, R. Burns, G. Eyink, A. Szalay, R. Moser, et al., *A web services accessible database of turbulent channel flow and its use for testing a new integral wall model for les*, Journal of Turbulence, 17 (2016), pp. 181–215.

[6] K. Lee, N. A. Trask, R. G. Patel, M. A. Gulian, and E. C. Cyr, *Partition of unity networks: deep hp-approximation*, arXiv preprint arXiv:2101.11256, (2021).

[7] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, *A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence*, Journal of Turbulence, (2008), p. N31.

[8] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, *Fourier neural operator for parametric partial differential equations*, arXiv preprint arXiv:2010.08895, (2020).

[9] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, *Physics-informed neural operator for learning partial differential equations*, ACM/JMS Journal of Data Science, 1 (2024), pp. 1–27.

[10] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis, *A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data*, Computer Methods in Applied Mechanics and Engineering, 393 (2022), p. 114778.

[11] R. G. Patel and O. Desjardins, *Nonlinear integro-differential operator regression with neural networks*, arXiv preprint arXiv:1810.08552, (2018).

[12] R. G. Patel, N. A. Trask, M. A. Wood, and E. C. Cyr, *A physics-informed operator regression framework for extracting data-driven continuum models*, Computer Methods in Applied Mechanics and Engineering, 373 (2021), p. 113500.

[13] E. Perlman, R. Burns, Y. Li, and C. Meneveau, *Data exploration of turbulence simulations using a database cluster*, in Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007, pp. 1–11.

[14] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational physics, 378 (2019), pp. 686–707.

[15] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, *Outrageously large neural networks: The sparsely-gated mixture-of-experts layer*, arXiv preprint arXiv:1701.06538, (2017).

[16] L. N. Smith and N. Topin, *Super-convergence: Very fast training of neural networks using large learning rates*, in Artificial intelligence and machine learning for multi-domain operations applications, vol. 11006, SPIE, 2019, pp. 369–386.

[17] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, et al., *Pytorch fsdp: experiences on scaling fully sharded data parallel*, arXiv preprint arXiv:2304.11277, (2023).

EXPLORING MACHINE LEARNING SURROGATES FOR MOLECULAR DYNAMICS SIMULATIONS

ARTHUR FEENEY* AND SIVA RAJAMANICKAM†

Abstract.
Molecular dynamics (MD) simulations are often restricted to integration timesteps that are smaller than the vibrational frequency of atoms—on the order of femtoseconds (10^{-15} seconds). However, scientists and engineers are interested in phenomena that occur over much longer timescales. This difference in time scales means that MD simulations often require many millions of integration timesteps. This work explores accelerating MD simulations via data-driven techniques to approximate the time-evolution and enable taking very large timesteps. We show that it is possible to reproduce the long-time dynamics of single-chain polymers. We also present an E(3)-equivarient model, that is equivariant with respect to a history of atom positions. Finally, we discuss possible future directions to overcome current limitations.

1. Introduction. Molecular dynamics (MD) simulations are often restricted to integration timesteps that are smaller than the vibrational frequency of atoms—on the order of femtoseconds (10^{-15} seconds). However, scientists and engineers are interested in phenomena that occur over much larger timescales: as an extreme example, protein-ligand association may occur over $10^1 - 10^3$ seconds. This difference in time scales means that MD simulations often require many millions of timesteps. This problem has been studied for decades, so there are a number of existing approaches to alleviate this "timescale barrier." One approach is to make each integration step faster, which may involve using better hardware or developing software optimizations [5]. A second approach is using strategies like coarse graining [2], where the coarsened system may have smoother vibrational frequencies and thus can be simulated using larger timesteps. A third approach is accelerated MD [4, 7], which accelerates how quickly an MD simulation can explore the configuration space by reducing energy barriers between states.

This work attempts to accelerate MD simulations by taking much larger timesteps, at the expense of accuracy. We explore approximating the time-evolution operator, $T_{\Delta t}$, using machine learning. For a fixed timestep Δt, which is ideally much larger than the maximum timestep that can be used with a standard MD simulator, we step forward in time via repeated application of this operator $x(\Delta t N) = T^N_{\Delta t}(x(0))$, where $x(t)$ is the positions of all particles at time t and N is the number of times the operator is applied.

We note that reproducing a particular trajectory is effectively impossible due to the chaotic nature of MD simulations. Thus, our goal is only to produce representative trajectories, from which we can compute approximations of desired properties. A potential advantage of this, compared to learning to predict properties directly from a configuration, is that we can avoid categorizing properties before training. After we have trained our model, the produced trajectories can be used for more general analysis.

This work is an extension of recent work by Fu [3], which explores a similar idea to accelerate MD simulations. Their major contributions that we use were the use of coarse-graining and having separate "encoder module" (which does the coarse-graining) and "dynamics module" (which is used for a coarse-grained forward simulation). An additional goal of this project is to explore what may be necessary to build a potential "foundation model" for surrogates of MD simulations. By this, we essentially mean a model that can handle a variety of materials and conditions without having to make problem-specific architecture and modeling choices. This is of course a more distant goal.

*University of California Irvine, afeeney@uci.edu
†Sandia National Laboratories, srajama@sandia.gov

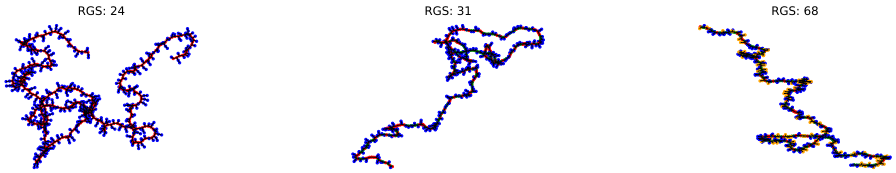


Fig. 2.1: Several class II polymers [10] with different radii of gyration. From left to right, these polymers are those with RGS values that had the minimum, median, and maximum variance over the course of a full simulation. So, the left-most polymer tends to be clumped together while the right-most fluctuates. The atom colors correspond to different particle types.

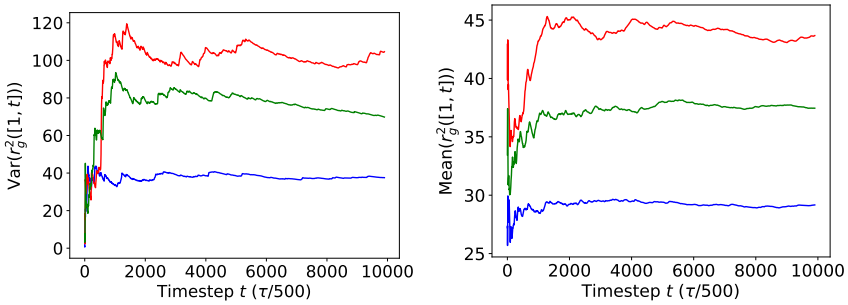


Fig. 2.2: Illustrating the time to convergence of r_g^2 for three polymers from our dataset [3], with the minimum (blue), maximum (red), and median (green) r_g^2 variance. In order to get good approximations of r_g^2 the simulations must be run for sufficiently long so that the mean and variance nearly converge. The y-axes correspond to the variance (left) or mean (right) through to the first t timesteps.

2. Background.

2.1. Polymers. A polymer is a large molecule composed of many identical parts called monomers. For instance, polyethylene is a polymer chain with monomers of C_2H_4. In a simulation, a polymer chain may consist of atoms linked by rigid bonds. The angle between bonds θ and the torsional angle ϕ defined by three adjacent bonds are both allowed to vary. In addition to the potential for interatomic forces U, there is a bond angle potential U_θ and a potential for the torsional angles U_ϕ [1].

The metric we are interested in is the *distribution* of the radius of gyration squared (RGS), which we denote r_g^2. The RGS can be informative for understanding rheological behavior of polymers, because it can be related to the onset of chain entanglements and gelation [10]. The RGS requires sufficiently long simulations to reliably reach good approximations. This time to convergence is illustrated in Figure 2.2, which shows that the time-averaged mean and variance of the RGS take several million τ to converge. We show several single-chain polymers with varying radii of gyration in Figure 2.1.

Our dataset consists of single-chain polymers in a solvent liquid. Instead of explicitly modelling all of the solvent particles, they are handled implicitly via a Langevin thermostat.

$$\ddot{x}(t) = f(x,t) - \alpha\dot{x}(t) + \beta(t) \tag{2.1}$$

This equation essentially says that the atoms' acceleration depends on the various interatomic forces, a damping term $\alpha \geq 0$, and a noise term β that has zero mean and is uncorrelated between timesteps. The "random" interactions between the solvent and polymer are handled implcitly by the noise term. In practice, we found that handling this non-determinism makes it challenging for an ML model to reproduce the dynamics of a system.

2.2. Graph Neural Networks. Graph neural networks can be applied to graph structured data $G = (V, E)$. Graph neural networks process graphs via "message passing," which propagates information along edges. The update of a vertex $i \in V$, based on it's neighbors $N(i)$, may look like

$$\begin{aligned} m_{ij} &= f(v_i, v_j, e_{ij}) \\ m_i &= \sum_{j \in N(i)} m_{ij} \end{aligned} \tag{2.2}$$

The result m_i is the new representation of node i and m_{ij} is the message from node j to node i. The v_i and v_j correspond to node attributes, which may be encodings of positions, node types, velocities, etc. The e_{ij} are edge attributes, which could be the length of the edge. When applying graph neural networks to MD problems, the vertices V often correspond to atoms. Edges may be defined somewhat arbitrarily depending on the problem; they could correspond to bonds, interatomic forces, or auxiliary edges that are only intended to propagate information.

2.3. Equivariance. For many physics applications, it can be beneficial to construct machine learning models that are explicitly equivariant to classical groups [6, 9]. Ideally, our models should be robust to changes in orientation of the simulation box. So, we are primarily concerned with the three-dimensional Euclidean group E(3), which includes rotations and translations.

DEFINITION 2.1 (Equivariance). *A function $f : X \to X$ is equivariant with respect to a group G if for all $g \in G$ and $x \in X$, we have $f(\rho(g)x) = \rho(g)f(x)$ where $\rho(g)$ is the representation of g in X.*

Similarly, f is *invariant* with respect to G if $f(\rho(g)x) = f(x)$. A simple way to construct an equivariant GNN is based on "scalarization" [9, 6], which relies on the fact the an equivariant vector times an invariant scalar results in an equivariant vector.

3. Methods. This work is exploratory and we are mostly interested in looking at what works and what does not. Thus, we make some simplifying assumptions. First, the model does not take force information as input. This makes it possible to avoid force calculations altogether, since the model should be able to infer it from position information. Second, because of the prior assumption, we are limited to datasets that consist of "similar" problems. I.e., we cannot the potential function. Third, it is acceptable to not satisfy conservation laws as long as we can match the expected trends. Fourth, we assume rare events will not be an issue. In some problems, phenomena like diffusion may occur relatively infrequently, making it difficult for a model to learn it well. For the polymer dataset, such rare phenomena are mostly absent.

3.1. Auto-regressive Model. We have a dataset of simulations, from which we sample a range of timesteps to use as input for training. The model processes this history and approximates the atom positions in the next timestep. After a model is trained, we evaluate it via a timestepping method. The machine-learning community calls this an "auto-regressive" model. This is similar to numerical integrators. We step forward in time, using the model's most recent output as a new input. This is a very common strategy for handling forward dynamics with machine learning.

3.2. Modeling Langevin Dynamics. As discussed in section 2.1, the polymer simulations use a Langevin thermostat to model an implicit solvent [8]. This introduces non-determinism into the simulations, which over large timesteps can be moderately large. In practice, this non-determinism seems difficult for a standard ML model to handle, as shown in Figure 4.1.

In order to handle this non-determinism, we have the model output a mean and variance for a Gaussian distribution for the particle accelerations: $m_{t+1}, \sigma_{t+1} = T(x_t)$ and sample $\ddot{x}_{t+1} \sim \mathcal{N}(m_{t+1}, \sigma_{t+1})$. This acceleration is plugged into the semi-implicit Euler integrator to get the new particle positions [3]. LAMMPS' implementation of the langevin thermostat uses a uniform distribution, but to the best of our knowledge it is not possible train a model to output a uniform distribution, and either a uniform or normal distribution can be used with Langevin dynamics.

3.3. Coarse-Graining. Coarse-graining is a common strategy used in polymer simulations to reduce the degrees of freedon, while preserving the fundamental physics [2]. This takes the form of clustering "fine-grained" atoms into larger "coarse-grained" atoms. We borrow this idea from Fu [3], but it is relevant to discuss here because of the potential impact on equivariance. The encoder network is only applied to invariant node features. The coarse graining process only uses the the adjacency pattern, so it is certainly invariant to translations of particle positions and will have no effect on the equivariance of the model.

3.4. History. Since we use coarse-graining, the problem becomes non-Markovian. So, passing in history information becomes necessary (or at least beneficial). Unfortunately, equivariant GNNs are typically only constructed to handle a single input coordinate associated with each node. We make an extension so that each node can be E(3)-equivariant with respect to a short history of positions.

3.5. Equivariant GNN. Our equivariant model is similar to EGNN [6], but it should be able to process a a history of velocities for each node. We restate two relevant lemmas from [9], essentially highlighting that $O(d)$-equivariant vector-valued functions can be expressed as a function of scalar products.

LEMMA 3.1. *An $O(d)$-invariant scalar function of $V \in \mathbb{R}^{n \times d}$, denoted $f(V)$, can be written as a function g of the scalar products of the v_i: $f(V) = g((\langle v_i, v_j \rangle)_{i,j \in [n]}) = g(VV^\top)$.*

LEMMA 3.2. *An $O(d)$-equivariant vector-valued function, h, can be written as $h(V) = \sum_{t \in [n]} f_t(V) v_t = \sum_{t \in [n]} g(VV^\top) v_t$*

Using these Lemmas, we can construct an O(d)-equivariant layer. Since we process a history of positions for each node, we can compute "velocities" by computing a difference between timesteps to make the model translation equivariant. Thus, the layer we will describe can be made E(3)-equivariant.

The input to layer l of our model is a graph $G = (V, E)$. Each node $i \in V$ has an associated history $\mathcal{H}_i^l \in \mathbb{R}^{T \times d}$ capturing the node's (possibly noisy) d-dimensional position over the prior T timesteps. For edges $(i, j) \in E$ we can compute the O(d)-equivariant features $V_{ij} = \mathcal{H}_i^l - \mathcal{H}_j^l$. The product $V_{ij} V_{ij}^\top$ is O(d)-invariant. Based on Lemmas 3.1 and 3.2, our equivariant layer then has a form similar to EGNN.

$$
\begin{aligned}
m_{ij} &= \phi_1(V_{ij}V_{ij}^\top) \\
\mathcal{H}_i^{l+1} &= \mathcal{H}_i^l + C \sum_{j \in N(i)} \operatorname{diag}(\phi_2(m_{ij}))V_{ij}
\end{aligned} \tag{3.1}
$$

The functions ϕ_i are standard MLPs and $N(i)$ denotes the set of neighbors of node i. C is just a chosen constant and can be set to $|N(i)|^{-1}$ [6]. We scale V_{ij} by a diagonal matrix, rather than a scalar, in an effort to improve expressivity.

4. Experiments. The experiments are primarily intended to demonstrate feasibility of MD surrogates.

4.1. Dataset. We look at a polymer dataset from [10]. We show several polymers from the test dataset in Figure 2.1. There are 100 simulations used for training and 40 used for testing. The training simulations are short and only run for $50,000\tau$. The simulations used for testing are run for $5,000,000\tau$. The LAMMPS simulations use a timestep of 0.01τ and our model uses a timestep of 5τ, or $500\times$ larger. (Since one inference of an ML model is slower than a LAMMPS timestep, the hypothetical speedup may be much less than $500x$.) The LAMMPS simulations use a Langevin thermostat to model interactions with an implicit solvent [8]. This introduces some non-determinism to the simulations that we found adds an additional challenge to faithfully approximating the dynamics and estimating the distribution of the RGS.

4.2. Non-determinism. An unexpected challenge with this problem is handling the noise present in Langevin dynamics. We found that using a standard deterministic model typically fails to capture the distribution of the RGS. To illustrate this, we compare two rollouts of the same trained model in Figure 4.1. Recall that the model produces a mean and standard deviation for a Gaussian distribution. For the deterministic output, we directly use the predicted mean as the acceleration. For the Gaussian output, we sample an acceleration from the Gaussian output by the model. The model using a deterministic output clearly fails to capture the distribution of the RGS, since the variance is extremely small. When sampling the output from a Gaussian distribution, the model can mimic the dynamics quite well. This result highlights two key things. First, it is feasible to have an ML model reproduce the long-time dynamics of MD simulations. Second, it seems that standard models will be unable to reproduce stochastic dynamics.

4.3. Equivariant Model. We have negative results for our current approach to rotation equivariance. While the model trains well and achieves a similar one-step validation loss to the non-equivariant model, it is significantly less stable when used in a full test rollout and the RGS values diverge. When the model performs timestepping, it is using its past output as new inputs. So, the input distribution will slightly change. It is difficult to exactly characterize why the model is struggling in the auto-regressive setting. It could be due to overfitting or greater sensitivity to changes in the input distribution.

5. Related Work. This project is inspired by the massive amount of recent work looking at creating machine learning-based surrogates for simulations. For continuum PDEs, there are many variations of Neural Operators and DeepONets. These approaches rely on having a dataset of existing simulations to train a model. The primary interest in these surrogates is due to claimed performance improvements compared to numerical solvers on some problems. There is also work exploring circumventing the timescale barrier by approximating transfer operators [7], which is similar to our own approach. This work is

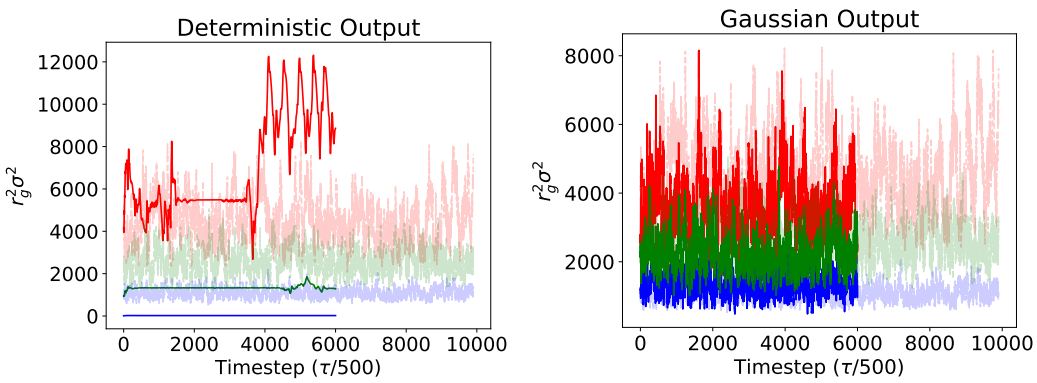


Fig. 4.1: Example rollouts corresponding the the polymers with the minimum (blue), maximum (red), and median (green) variance of r_g^2 over the full simulation. The y-axis shows the instantaneous r_g^2 times its variance. The opaque lines correspond to the LAMMPS simulations and the dark lines are the ML model. The deterministic model obviously struggles to mimic the distribution of the RGS, while the model that samples from a Gaussian can capture the distribution quite well.

mostly based on the approach taken in [3], which, in addition to the polymer dataset, also looks at a Li-ion dataset and uses a denoising score model to improve their results.

6. Limitations. As discussed, we made a number of simplifying assumptions including that the potential function is fixed for all simulations in a dataset. This could be overcome just by computing forces in each step and passing them into the model. Furthermore, a trained model is limited to one thermodynamic ensemble. We also assume a fixed timestep. For some systems, it may be useful to take even larger timesteps or make the timestep a more flexible choice.

There are also general challenges with gathering data and training. The dataset we used is on the order of gigabytes, but single MD simulations may be terabytes. Building and storing a sufficiently large and diverse dataset seems to be a major challenge. Furthermore, faithfully reproducing dynamics over longer time scales is fairly difficult. As with most applications of deep learning, more data is better. As an optimistic estimate, training a surrogate would likely require on the order of several dozen simulations. Our dataset has around 100 simulations for training. Training a model, even on this fairly simple polymer dataset, can take over 12 hours. This makes it challenging to perform hyperparameter tuning or experiment with different modelling approaches. A possible way simplify creating and storing datasets of more diverse systems could be to consider only extremely coarse timesteps (like 10^6 fs).

7. Conclusion and Future Work. The main results highlight that creating a surrogate for MD simulations is feasible. The surrogate model can reproduce the variation and expectation of the radius of gyration for unseen polymer systems. We also identify some notable challenges for "foundation models." A major hurdle will be handling both stochastic and deterministic dynamics. The results indicate that standard supervised training "input-output" pairs does not reproduce the dynamics of a stochastic system. In future work, it could be beneficial to prioritize stochastic systems, and treat deterministic systems as a special case with zero noise.

REFERENCES

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Oxford University Press, 06 2017.

[2] S. Dhamankar and M. A. Webb, *Chemically specific coarse-graining of polymers: Methods and prospects*, Journal of Polymer Science, 59 (2021), pp. 2613–2643.

[3] X. Fu, T. Xie, N. J. Rebello, B. Olsen, and T. S. Jaakkola, *Simulate time-integrated coarse-grained molecular dynamics with multi-scale graph networks*, Transactions on Machine Learning Research, (2023).

[4] D. Perez, B. P. Uberuaga, Y. Shim, J. G. Amar, and A. F. Voter, in Chapter 4 Accelerated Molecular Dynamics Methods: Introduction and Recent Developments, R. A. Wheeler, ed., vol. 5 of Annual Reports in Computational Chemistry, Elsevier, 2009, pp. 79–98.

[5] K. Santos, S. Moore, T. Oppelstrup, A. Sharifian, I. Sharapov, A. Thompson, D. Z. Kalchev, D. Perez, R. Schreiber, S. Pakin, E. A. Leon, J. H. L. I. au2, M. James, and S. Rajamanickam, *Breaking the molecular dynamics timescale barrier using a wafer-scale system*, 2024.

[6] V. G. Satorras, E. Hoogeboom, and M. Welling, *E(n) equivariant graph neural networks*, CoRR, abs/2102.09844 (2021).

[7] C. Schütte, S. Klus, and C. Hartmann, *Overcoming the timescale barrier in molecular dynamics: Transfer operators, variational principles and machine learning*, Acta Numerica, 32 (2023), p. 517–673.

[8] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, *LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales*, Comp. Phys. Comm., 271 (2022), p. 108171.

[9] S. Villar, D. W. Hogg, K. Storey-Fisher, W. Yao, and B. Blum-Smith, *Scalars are universal: equivariant machine learning, structured like classical physics*, Neural Information Processing Systems, (2021).

[10] M. A. Webb, N. E. Jackson, P. S. Gil, and J. J. de Pablo, *Targeted sequence design within the coarse-grained polymer genome*, Science Advances, 6 (2020), p. eabc6216.

DESIGNING A MACHINE-LEARNED INTERATOMIC POTENTIAL FOR GOLD-PROMOTED NICKEL CATALYSTS UTILIZING MAGNETIC TRAINING DATA

ISABELLA FURRICK*, MITCHELL WOOD†, AND ALYSSA HENSLEY‡

Abstract. The hydrogen oxidation reaction (HOR) plays a critical role in various fossil-fuel replacement technologies, including hydrogen fuel cells and biofuel upgrading processes. To enhance the accessibility of these technologies, nickel (Ni) catalysts doped with gold (Au) offer a cost-effective alternative to the conventional, yet expensive, platinum catalysts. However, during HOR, such NiAu catalysts undergo adsorbate-driven surface reconstruction, where atoms in the subsurface migrate to the surface or vice versa. This phenomenon is difficult to capture via computational models due to the large length- and time-scales needed. Here, we overcome the limitations of length- and time-scales in computational models of NiAu catalysts by developing a machine learned interatomic potential (ML-IAP). A key question associated with developing ML-IAPs for Ni-based systems is whether or not inclusion of Ni's magnetism within the ML-IAP training data is necessary to accurately capture the structure and energies of NiAu nanoparticles. Here, we address this issue by developing a ML-IAP from spin-polarized first-principles calculations. To evaluate the impact of including magnetism in the training data on ML-IAP model accuracy, we plan to benchmark this spin-inclusive model against a comparable ML-IAP that does not account for spin in future work. We developed a comprehensive training dataset of over 60,000 data points and used the ML-IAP fitting software, FitSNAP, to create a linear Atomic Cluster Expansion (ACE) potential from this data. The potential was then applied to various structures using the molecular dynamics code, LAMMPS, with fourteen system-specific objective functions optimized through the optimization software, Dakota, to refine the model parameters. Our final model successfully predicted 11 out of 12 energy-related objective functions within 0.03 eV/atom. Additionally, it accurately captured the behavior of NiAu systems across a range of sizes, compositions, structures, and temperatures. This work paves the way to address two critical questions in the large-scale modeling of Ni-based catalyst systems: is incorporating magnetism within the training data essential for achieving an accurate ML-IAP model, and can we develop a computational model capable of capturing NiAu catalyst surface reconstruction during HOR? These advancements promise to enhance catalyst performance, thus driving sustainable energy technology and leading us towards a carbon-free future.

1. Introduction. Energy derived from fossil fuels contributes to global warming and energy security concerns [1]. Alternative energy sources are being explored by researchers across the world with hydrogen fuel cells and biofuels being two promising options. A key reaction for both technologies is the hydrogen oxidation reaction (HOR). Hydrogen fuel cells produce energy by splitting hydrogen (H_2) into protons and electrons at the anode and then recombining them with oxygen (O_2) at the cathode via HOR. This technology offers an efficient and environmentally friendly method for energy conversion with water (H_2O) as the only by-product [2, 3]. Biofuels are derived from renewable biological sources such as plant matter. To improve their suitability for use in internal combustion engines, H_2 is used to reduce the oxygen content of bio-oils via hydrodeoxygenation (HDO) [4, 5]. In both applications, a heterogeneous catalyst is required to facilitate HOR, with platinum (Pt) being the most efficient and widely used material [6]. However, the scarcity and high cost of Pt increases the price of sustainable energy technologies, limiting their widespread use [7, 8]. Thus, it is critical to find a cheaper alternative to Pt.

Nickel (Ni) is a promising alternative catalyst material for HOR because it is abundant, cost-effective, and highly stable in alkaline environments and near the HOR equilibrium potential [9, 10]. However, it does not achieve the same level of selectivity and yield as Pt. One effective method for enhancing the performance of a pure Ni catalyst is doping it with a secondary promoter metal [11]. A study by Furrick et al. demonstrated that adding just 6% gold (Au) to a Ni catalyst led to significant improvements in performance,

*Stevens Institute of Technology, ifurrick@stevens.edu
†Sandia National Lab, mitwood@sandia.gov
‡Stevens Institute of Technology, ahensley@stevens.edu

specifically increasing turnover frequency [12]. Introducing a secondary metal into a catalytic system adds a layer of complexity to the design. During reaction, when molecules are adsorbing to and desorbing from the catalyst surface, bimetallic catalysts may undergo a phenomenon known as surface reconstruction, where metal atoms from the subsurface move to the surface and vice versa [13, 14, 15]. The composition and structure of the catalytic surface directly affect reaction kinetics, and bimetallic systems achieve optimal performance when their constituent metals are arranged in specific configurations. Specifically, for Ni catalysts, dopants have a more pronounced effect on HOR when located in the surface layer of the catalyst [12]. Though extremely difficult to probe experimentally, understanding and accounting for in situ surface reconstruction is crucial for the design of bimetallic catalysts. Therefore, we aim to develop a model to study surface reconstruction in NiAu catalysts.

Quantum mechanical calculations, such as density functional theory (DFT) and *Ab Initio* Molecular Dynamics (AIMD), provide the most accurate energetic characterization of systems but are computationally expensive. These methods have limitations regarding the size and timescales of systems that can be modeled. Surface reconstruction occurs across entire catalytic nanoparticles which are too large to be modeled with first-principles calculations. Molecular Dynamics (MD) simulations can model larger systems and longer time scales but with less accuracy. In MD simulations, the interatomic potential (IAP) calculates the forces and energies on atoms and defines the physical properties of the system. Here, we take a multiscale modeling approach by employing machine learning (ML) to train an IAP based on a database of DFT and AIMD calculations, bridging the gap between quantum mechanical and molecular dynamics methods.

One of the key challenges in computationally modeling Ni-based systems is accurately capturing the magnetic properties of Ni. Ni is ferromagnetic, meaning it retains its magnetism even without an external magnetic field, a phenomenon known as spontaneous magnetization [16, 17]. The magnetic moment of a Ni atom originates from the unpaired electrons in its 3d orbital. In ferromagnetic materials like Ni, when the orbitals from the unpaired valence electrons of adjacent atoms overlap, the total energy with respect to the distribution of charge is minimized when electrons have parallel spins. Computationally searching for these minimum energy states can be accomplished within DFT and AIMD simulations but is often challenging due to the need to find self-consistent charge densities that strongly interact with itself, subject to these constraints on spin. Critically, it remains unknown how the inclusion or exclusion of magnetism and spin in Ni atoms within the DFT and AIMD training data impacts the accuracy of ML-IAPs when modeling catalytic behaviors such as surface reconstruction. The long-term objective of our study aims to shed light on this uncertainty by comparing predictions for NiAu catalyst reconstruction during HOR from separate ML-IAPs developed from two distinct training data: magnetic and nonmagnetic. This specific project accomplished a major component of our long-term objective by developing an accurate NiAu ML-IAP trained on solely magnetic DFT and AIMD data.

2. Methods. The long-term objective of this ongoing project is to develop a three-element (Ni, Au, O) potential to study surface reconstruction during HOR on an O-covered NiAu nanoparticle. A logical starting point for developing a new multi-element potential is to address the constituent elements with similar bonding characteristics before trying to generalize it. As such, we will construct training data and optimize potentials specific to the NiAu binary alloy. Two NiAu potentials were constructed to assess whether incorporating magnetism in the training data is necessary in the final potential: a *magnetic potential*, where Ni atoms are modeled with spin, and a *non-magnetic potential*, where spin is not included for Ni atoms. In both potentials, Au is modeled without spin, as it is non-magnetic. These differences come down to how the DFT training data is generated, which will di-

rectly translate to differences in the pair of potentials. For simplicity, neither potential will explicitly resolve the spin dynamics [18], but observations of the differing predictions may motivate this more complex modeling method in MD. This report will detail the process of developing the magnetic two-element NiAu potential.

2.1. Atomic Cluster Expansion (ACE). An interatomic potential is needed by every classical molecular dynamics simulation. The model form of these potentials has evolved over time from analytic forms of simple bonding types such as dispersion and electrostatics (Lenard-Jones[19], Coulomb[20]) to complex machine learned [21] versions. In the past decade, such data-driven models have shown remarkable accuracy while offering a favorable balance between precision and computational cost [22]. A ML-IAP has three key components 1) a training set that contains ground truth values of total energy and forces per atom, 2) A set of descriptors (features) that serve as input to 3) a model form that outputs per-atom energy and force. Constructing the training set that captures the desired material properties and end use cases is non-trivial, and will be explained in detail in subsequent sections.

For simplicity we will adopt a linear model form versus a more complex neural network, though the same procedures outlined here apply to the training of either model form. Which leaves the choice of descriptor set, where special attention to the accuracy versus computational cost of constructing this basis set is needed. Many suitable options exist in the literature, and our search is narrowed by their availability in LAMMPS[23, 24]. A zoo of acronyms exist to define these descriptor sets; SNAP[25], GAP[26], POD[27]. The 'parent' set to all of these atom-centered bases is the Atomic Cluster Expansion (ACE) method[28, 29, 30]. Much like our quantum-chemical understanding of localized electron orbitals, the ACE descriptor set combines the radial and angular components into rotation and permutation invariant functions. Rather than capturing the electron density around an atom, the ACE basis is used to capture the density of neighboring atoms that surround a central atom. Equation 2.1 captures the basic representation of the tensor products that comprise each ACE basis function.

$$\Phi_{\boldsymbol{nlm}}(\boldsymbol{r}^N) = \prod_{j=1}^{N} R_n(r_{ij}) Y_l^m(\hat{\boldsymbol{r}}_{\boldsymbol{ij}}) \tag{2.1}$$

The product of basis functions for Φ is evaluated for each neighboring atom j at distance r_{ij} from the central atom i, where the product extends over all neighbors, N. Full detail of the mathematical formulation can be found in Goff *et. al.*[29]. The ACE basis needs truncation at some upper limit of basis functions–n, l, m–as there are diminishing accuracy returns for these higher order functions. As such, there are a small number of hyperparameters that control the number of radial and angular functions to be included in the basis expansion. The first of which is the distance away from the central atom to expand the neighboring atom density, known as the radial cutoff, or $rcutfac$ in our fitting code. Importantly, ACE decomposes the contributions from differing $ranks$ of atomic interactions, which defines the n, l, m limits in Equation 2.1. For example, the force experienced by an oxygen in a O_2 molecule constitutes of a 2-body interaction, and would be mapped onto a $rank = 1$ ACE descriptor. Separably, the interactions between triplets (and so forth) of atoms would be mapped onto higher rank descriptors. Radial basis functions,$R_n(r_{ij})$, are evaluated up to $nmax$, defined for each rank of descriptor. Angular functions, $Y_l^m(\hat{\boldsymbol{r}}_{\boldsymbol{ij}})$, are terminated at $lmax$ which are independently defined for each of the $ranks$. Lastly, $mumax$ is singly defined for all $ranks$. An extension to a chemically informed basis is available in Ref. [29].

2.2. Training Data Set. ML-IAPs predict forces and energies in a system by numerically interpolating between quantum-mechanical reference data, typically derived from high-fidelity density functional theory (DFT) calculations[31]. However, when predicting properties for new atomic configurations, extrapolation-estimating unknown values based on observed trends-is often necessary, which can lead to inaccurate or physically implausible results[32]. To reduce the need for extrapolation, we aimed to effectively span descriptor space by providing the model with a large, diverse training data set. Descriptor space encompasses the full range of possible structures that can be formed from a given composition of elements at any given set of conditions (i.e. temperature, pressure). By covering a larger portion of the descriptor space, the model's need for extrapolation is minimized, leading to more reliable predictions.

All first principles calculations were carried out using the Vienna Ab Initio Simulation Package (VASP). Initial DFT calculations were performed on the Dorothy high-performance computing (HPC) cluster at Stevens Institute of Technology. The electron-electron interactions were modeled using the Revised Perdew-Burke-Ernzerhof (RPBE) exchange-correlation functional [33]. All calculations were conducted with a minimum plane wave cutoff energy of 400 eV. Convergence of the wavefunction and charge density within each self-consistent field step was reached at a total energy tolerance of 10^{-4} eV. The Methfessel-Paxton (N = 1) smearing method was applied with a smearing width of 0.1 eV. In this work, all VASP calculations were spin-polarized to account for the magnetism of Ni, with the initial magnetic moment of Ni set to 2.0 and Au set to 0.0. The calculations were spin-polarized with an ISPIN value of 2. We specified the same positive relative orientation for all Ni atoms which is consistent with ferromagnetic materials. The first Brillouin zone of all DFT calculations was sampled using a k-point resolution of 0.02 in units of 2πÅ^{-1}.

Training data generally falls into two categories. The first category consists of domain knowledge structures, which are based on an understanding of the material system and include known or anticipated structures such as bulk metals, alloys, surfaces, and expected defects (e.g., vacancies and substitutions). The second category includes structures generated by methods like active learning and genetic algorithms, which automate the creation of training data. These structures are often referred to as beyond domain knowledge structures[34]. Here, only single-point DFT calculations were performed on all domain and beyond domain knowledge structures.

For this potential, the domain knowledge structures included four subsets. First, bulk Au and Ni in three different crystal structures–face-centered cubic (fcc), body-centered cubic (bcc), and hexagonal close-packed (hcp)–were examined with various lattice constants. Second, substitution defects–where one metal atom within either the fcc Ni or fcc Au crystal structure was replaced with a metal atom of the opposite type–were included for three supercell sizes: p(2x2x2), p(3x3x3), and p(4x4x4). Third, vacancy defects–where one metal atom was removed from either the fcc Ni or fcc Au crystal structure–were considered in each of three supercell sizes: p(2x2x2), p(3x3x3), and p(4x4x4). Fourth, possible bulk alloy crystal structures of Ni_3Au and $NiAu_3$ were obtained from Materials Project [35] and included. The descriptor space covered by the domain expertise DFT data is shown in Figure 2.1B.

To generate beyond domain knowledge structures, a genetic algorithm called Universal Structure Predictor: Evolutionary Xtallography (UPSEX) was used to predict stable and metastable structures based solely on chemical composition. For three compositions (Ni, Au, and NiAu), USPEX was configured to generate 100 structures with between four and twelve atoms, with a minimum ion distance of 2 angstroms. The descriptor space covered by the USPEX DFT data is shown in Figure 2.1C.

AIMD simulations provide insight into atomic movements under applied temperature over short timescales using the forces calculated at the DFT-level. These simulations were

performed with VASP on the Sandia HPC cluster, Amber. The canonical ensemble was used, keeping the number of atoms, structure volume, and system temperature constant throughout the simulation. Each simulation was run with 1.0 femtosecond timesteps at two different temperatures (300K and 1000K), using a minimum cutoff energy of 400 eV. AIMD simulations were conducted on both domain and beyond domain knowledge structures. For domain knowledge structures, AIMD data included bulk fcc, bcc, and hcp Ni and Au, slab calculations for the three fcc crystal facets with the lowest surface energies—(111), (100), and (110)—for both Ni and Au [36, 37, 38, 39], Ni_3Au and $NiAu_3$ alloys, and all substitution defects. The descriptor space covered by the domain expertise AIMD data is shown in Figure 2.1D. Additionally, AIMD was run on the beyond domain knowledge structures. The five USPEX structures from each composition (e.g., Ni, Au, NiAu) with the lowest formation energy (in eV/atom) were selected for AIMD simulations. The descriptor space covered by the USPEX AIMD data is shown in Figure 2.1E.

With training data from DFT and AIMD, we compiled a comprehensive training data set consisting of 60,231 data points (see Table 2.1 for a detailed breakdown of the quantity of each type). The descriptor space covered by the entire training data set is displayed in Figure 2.1A. This t-distributed stochastic neighbor embedding (t-SNE) projection of the data highlights similarities and differences between each category of training data. It is important to note the volume of descriptor space mapped by AIMD versus other training data types, which results in less extrapolative predictions and, thus, more stable models.

Table 2.1
Training Data Overview.

Training Data Type	Number of Training Data Points
Au Bulk AIMD	2,000
Au Surface AIMD	6,000
Au USPEX AIMD	10,000
Au Ground States	41
Au USPEX	100
NiAu Ground State AIMD	6,006
NiAu USPEX AIMD	10,000
NiAu Ground States	4
NiAu USPEX	100
Ni Bulk AIMD	2,326
Ni Surface AIMD	6,000
Ni USPEX AIMD	9,354
Ni Ground States	41
Ni USPEX	100
Substitution AIMD	8,159

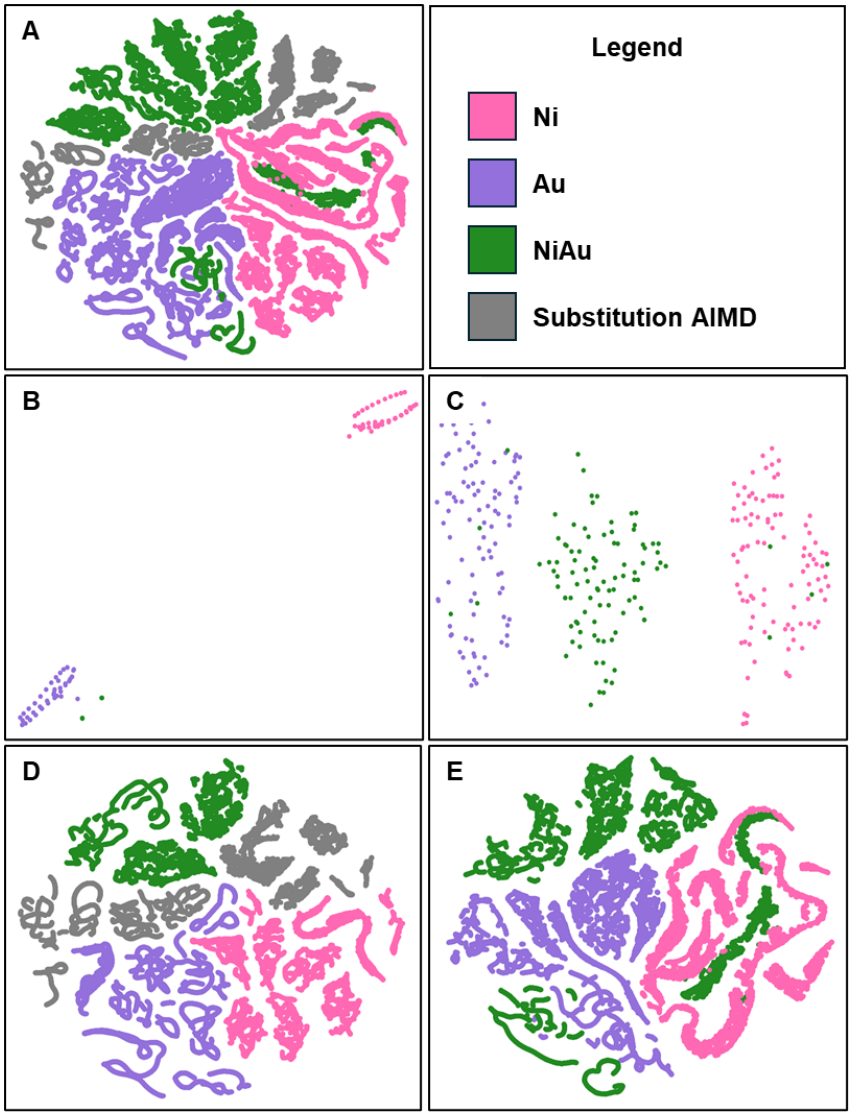


FIG. 2.1. *Visualization of the NiAu training data set coverage on the ACE descriptor space in 2D using distance preserving t-SNE analysis. (A) All data. (B) Domain Knowledge DFT (bulk, alloy, vacancy, and substitution structures). (C) USPEX DFT. (D) Domain Knowledge AIMD (bulk, alloy, surface, and substitution structures). (E) USPEX AIMD.*

2.3. Optimization Methodology. To produce a viable ML-IAP from our NiAu training data, a number of free variables in an ACE ML-IAP must be optimized. These generally fall into three categories: hyper-parameters that affect the calculation of the descriptors, group weights that shift the contribution of different components of the DFT training data within the ML-IAP optimization, and basis set parameters that affect the number of descriptors generated for a given system. Group weights directly influence the linear regression (RIDGE regression, L_2 regularization penalty of 10^{-6}) by biasing the loss function toward (or away from) subsets of the training set. The number of these group weights is unbounded, but we limit the number by collecting similar training data into said

groups.

To enhance model accuracy, we used Dakota- a software toolkit developed by Sandia- to identify the set of hyper-parameters and group weights that minimize force and energy errors, along with fourteen additional objective functions errors detailed in Section 3.1. Specifically, we employed Dakota's Single Objective Genetic Algorithm (SOGA) function for such optimization. The hyper-parameters optimized in this study were radial cutoff and lambda. The radial cutoff determines the number of atoms in a local neighbor list. Atoms exert energies, forces, and stresses on each other, and the radial cutoff determines how far an atom can be from another atom before its affect is negligible. The radial cutoff was scanned over the range 3.0 to 7.0 Å. Lambda determines how much emphasis is placed on the energies, forces, and stresses from nearby atoms versus those from atoms farther away. Lambda was scanned over the range 0.0 to 4.0. In an ACE potential, there is a hyper-parameter set for each element-element interaction type. For this system, there existed one set of radial cutoffs and lambdas for each of the following interaction types: Ni-Ni, Ni-Au, Au-Ni, and Au-Au.

Basis set parameters determine the number of ACE descriptors that are generated by FitSNAP to describe a given system. There is a trade-off between computational cost and the accuracy that additional descriptors bring to a system. Optimizing basis set parameters with a genetic algorithm like Dakota can be computationally expensive, thus basis set testing is done by hand. Three potentials, each using different basis set parameters as described in Table 2.2, were trained on the complete training dataset. Their hyper-parameters and group weights were optimized through Dakota to achieve maximum accuracy. To evaluate the computational cost associated with each basis set, a short, energy-minimization MD run was performed on a Ni fcc (111) facet containing 36 atoms. The MD simulation was run on a single core on Sandia's Amber cluster, and the time (in seconds) it took to run the simulation was recorded in Table 2.2. To evaluate the accuracy of each basis set, the energy and force root mean squared errors (RMSEs) were extracted from FitSNAP's automatically generated metrics file. The objective function RMSE was calculated based on the fourteen system-specific objective functions described below in Section 3.1. These three error metrics were also recorded in Table 2.2

TABLE 2.2
Basis set optimization parameters and results.

	Basis Set 1	Basis Set 2	Basis Set 3
Ranks	1 2 3 4 5 6	1 2 3 4	1 2 3
lmax	1 2 2 2 1 1	1 2 2 2	0 6 2
nmax	22 2 2 2 1 1	22 2 2 2	18 4 2
nmaxbase	22	22	22
Number of Descriptors	876	1082	698
Time to Run MD (seconds)	FAILED	24.9119	19.6926
Energy RMSE	0.0437034	0.108847	0.137224
Force RMSE	0.313889	0.232557	0.126694
Objective Function RMSE	3.4590939	2.174479262	1.634084964

Basis Set 1 was found to be inadequate for this potential. Despite resulting in the lowest energy RMSE, it had the highest RMSE values for both force and objective function among the three basis sets. More importantly, a brief MD simulation on a small Ni fcc facet failed, suggesting that the potential is unstable. Since this potential cannot be used effectively for small structures, it is not reliable for the simulations we are interested in, which involve larger structures, extended time scales, and varying temperature and pressure conditions. Basis Set 2 successfully completed the MD simulation and achieved lower force and objective function RMSEs than Basis Set 1. However, it was not selected as the optimal basis set. Basis Set 3, however, achieved a 25% lower objective function error RMSE compared to Basis Set 2 and completed the MD simulation approximately 21% faster. Due to its superior efficiency and accuracy, Basis Set 3 was chosen as the optimal set of parameters for this potential.

3. Results.

3.1. Objective Functions. The overall accuracy of a ML-IAP is determined by the potential's ability to predict known material properties, system stability, and energies/forces. To evaluate model accuracy in regards to known material properties, we set fourteen objective functions and tracked their errors over the course of potential development. To test the model's capacity for predicting structural properties, we set objective functions for the lattice parameters of fcc Ni and fcc Au. Predicted lattice constants were obtained through short LAMMPS simulations on bulk fcc Ni and fcc Au, and we determined the error by subtracting these predictions from the energy minimizing lattice constants from DFT. For evaluating the model's accuracy in predicting the energetic properties of the individual elements of the system, we created objective functions for the formation energies of bulk Ni and Au in fcc, bcc, and hcp phases. The predicted energies were derived from short LAMMPS simulations and compared to DFT-calculated energies. Using the same method, we set objective functions for the formation energies of Ni_3Au and $NiAu_3$ to assess the potential's performance in modeling the energetic properties of element combinations in the system. Additionally, we tested the model's ability to capture system imperfections by setting objective functions for the vacancy and substitution energies of fcc Ni and fcc Au, following a similar methodology to the other systems described above.

To evaluate the progression of the potential over the course of its development, the energy/force and objective function errors of three successive potentials are compared in the bar chart shown in Figure 3.1. The potential represented by the teal bars, Potential 1, was fit on June 18th. It was trained using only domain knowledge and USPEX DFT calculations (totaling 386 training data points), covering the descriptor space shown in Figure 2.1B and C. Potential 1 was generated with Basis Set 1 and did not undergo any hyper-parameter or group weight tuning. The potential represented by the pink bars, Potential 2, was created on August 8th and was trained using domain knowledge DFT, USPEX DFT, and USPEX AIMD data, totaling 29,740 training data points and covering the descriptor space shown in Figure 2.1B, C, and E. This potential utilized Basis Set 3 and underwent tuning of hyper-parameters and group weights via Dakota. Finally, the potential represented by the orange bars, Potential 3, was created on August 12th and trained on the entire training dataset, which includes domain knowledge DFT, USPEX DFT, domain knowledge AIMD, and USPEX AIMD. This dataset covers the descriptor space shown in Figure 2.1A, comprising a total of 60,231 training data points. Like the previous potential, it used Basis Set 3, with its hyper-parameters and group weights optimized by Dakota.

Among the three potentials, Potential 3 demonstrated the highest accuracy, reporting the lowest errors in 8 key metrics: force MAE, Ni lattice parameter error, Au lattice parameter error, Ni hcp formation energy error, Ni substitution energy error, Au fcc formation

energy error, Ni_3Au formation energy error, and $NiAu_3$ formation energy error. It fully outperformed the other models in objective functions aimed at evaluating the model's accuracy in capturing both the structural properties of the system and the energetic properties of elemental combinations. Furthermore, Potential 3 achieved the second lowest energy errors for 5 metrics: Ni fcc formation energy error, Ni bcc formation energy error, Ni vacancy energy error, Au hcp formation energy error, Au vacancy energy error, and Au substitution energy error. Note, the bar chart in Figure 3.1 is presented on a log scale. In non-log space, the units are as follows: Energy MAE in eV/atom, Force MAE in eV/Å, Ni/Au lattice parameter errors in Å, and the remaining metrics in eV/atom. For Potential 3, all formation, vacancy, and substitution energy objective function errors were below 0.03 eV/atom, with the exception of the Au substitution energy error, which proved to be particularly challenging to model accurately. Despite this, the overall level of objective function error achieved by Potential 3 is remarkably precise, on the order of irreducible error between MD and DFT predictions. As one moves between computational methods, differences in approximations made by each method. A small baseline error is expected in all energy errors due to the underlying theory differences of a true many-body (DFT) interaction and those that are localized by using a radial cutoff (ML-IAP). Detailed comparisons between DFT calculated adsorption energies and experimental microcalorimetry measurements show that careful selection of the exchange-correlation functional results in an error of at most 0.2 eV [40]. Thus, the errors predicted by Potential 3 are approximately an order of magnitude smaller than the acceptable error range typically seen between DFT calculations and experimental data.

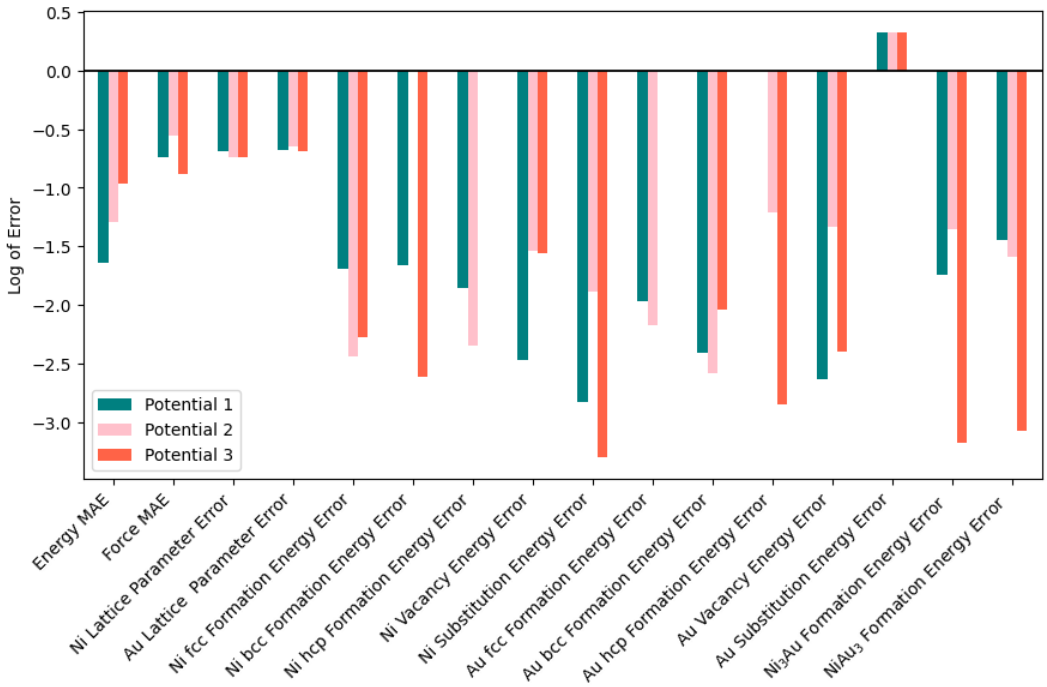


FIG. 3.1. *Error metric comparison over the course of potential development.*

3.2. Stability Tests. While objective function errors are an excellent way to gauge the accuracy of a potential, they often do not provide the complete picture. The true purpose of

a potential is to apply it to structures outside the training data set and use it in molecular dynamics simulations under various conditions, such as different temperatures, pressures, or time scales. A well-trained and optimized potential will be stable and exhibit physically and chemically realistic behavior under these circumstances.

Potentials can fail in several ways. One common issue is "black-holing," where repulsive forces are poorly represented, causing too many atoms to occupy the same space. In this case, LAMMPS outputs the following error message: "Encountered very small distance. Stopping." Another failure mode is "exploding," which occurs when attractive forces are inadequately captured, causing atoms to drift apart in space, leading to the error "ERROR: Lost atoms." Lastly, the simulation may complete, but the structure's behavior may deviate significantly from realistic physical behavior. For example, this could manifest as a pure nickel surface melting at an unrealistically low temperature, such as 300 K.

Potential 3, which achieved the best objective function error results, was subjected to a series of stability tests to evaluate its ability to accurately capture the physics and chemistry of Ni and Au across various length/time scales and under different reaction conditions. Figure 3.3 presents the results of two such stability tests. The potential was first applied to a pure Ni(110) facet in LAMMPS. This simulation ran for 1 picosecond at 300K. The facet maintained its structure throughout the simulation, indicating stability. Although this is a promising result, the Ni(110) facet was part of the training dataset, so further tests were needed to assess the model's ability to extrapolate and predict the behavior of structures not explicitly included in the training data. Additionally, we aimed to test the model's performance over longer simulation times and at temperatures higher than room temperature. To this end, Potential 3 was applied to a 2-nanometer NiAu nanoparticle with a composition of 20% gold, where the gold atoms were randomly distributed within the nanoparticle. This simulation was run for 10 picoseconds at 1000 K. Excitingly, the nanoparticle maintained its structure, with atoms vibrating and moving in a physically realistic manner. To evaluate this behavior further and in a more quantitative way, the Ni-Ni, Ni-Au, and Au-Au radial distribution functions (RDFs) of the simulation were evaluated before and after 10 picoseconds, as shown in Figure **??**. The RDF describes how the density of surrounding atoms varies as a function of distance from a reference atom, providing insight into the frequency of specific atomic separations. RDFs of crystalline structures exhibit pronounced peaks, indicating preferred atomic distances within the structure [41]. For this simulation, first peaks of the RDFs at both 0 ps and 10 ps show that the Ni-Ni peak appears at the shortest distance (2.5Å and 2.5Å), followed by the Ni-Au peak at a larger distance (2.6Å and 2.7Å), and finally the Au-Au peak at the farthest distance (2.7Å and 3.0Å), consistent with the larger atomic radius and lattice constant of Au compared to Ni. Comparing the two RDFs, the peaks at 10 ps have broadened and are less smooth than those at 0 ps, which is expected as the temperature increased from 0 to 1,000 K (near the metals' melting temperatures). The rounded peaks at 0 ps indicate that the system retains a mostly crystalline structure, while the more jagged peaks at 10 ps suggest increasing atomic disorder as thermal energy is introduced, disrupting the crystal lattice. These observations confirm that while the nanoparticle remains structurally intact, increasing thermal energy leads to noticeable atomic rearrangements and a gradual loss of long-range order. Overall, the stability tests outlined here demonstrate that Potential 3 can accurately capture the structural integrity and atomic interactions of Ni and Au systems across different conditions, while also revealing its ability to handle extrapolation, longer simulation times, and elevated temperatures with realistic atomic behavior.

4. Conclusions. In this study, we developed a two-element NiAu ML-IAP trained on spin polarized DFT data. During the development of the potential, we expanded our

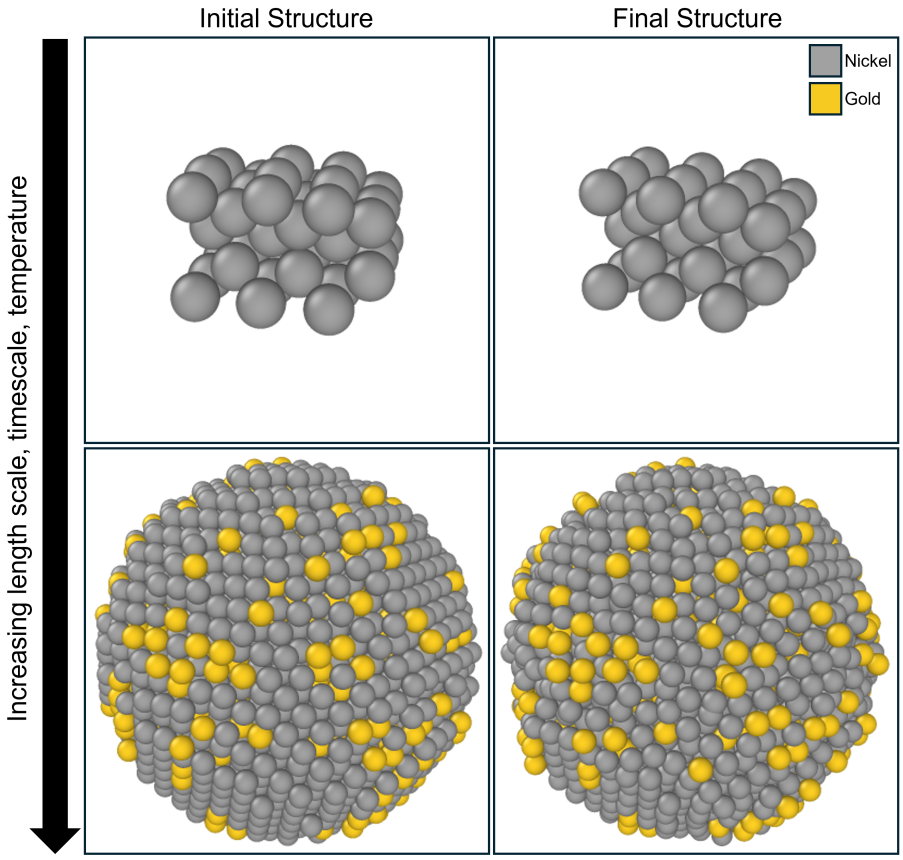


FIG. 3.2. *Stability test results for the Ni fcc (110) facet at 300 K (top) and NiAu nanoparticle at 1000 K(bottom). Structures shown are from before and after MD simulations.*

training dataset from fewer than 400 to approximately 60,000 data points by including AIMD data for both domain knowledge and beyond domain knowledge structures. This addition significantly broadened our span of the descriptor space and improved model accuracy. We evaluated three basis sets and determined that Basis Set 3, which contained the fewest number of descriptors, was the most effective for this potential. It resulted in the lowest computational cost for simulations and minimized errors in energy, force, and objective functions. Additionally, optimizing hyper-parameters and group weights was crucial for the development of the ML-IAP. The use of SOGA via Dakota for tuning these parameters significantly enhanced the accuracy of our model in a short time.

The most advanced potential, Potential 3, was trained on the full 60,231-point dataset, employed the most effective basis set (Basis Set 3), and had its hyper-parameters and group weights optimized using Dakota. Among the 12 energy-related objective function error metrics assessed, the model achieved accuracy within 0.03 eV/atom for 11 of them. Moreover, the model demonstrated the ability to accurately predict the behavior of NiAu systems across various sizes, compositions, and structures, and under different reaction conditions. These included larger structures that may be similar to training data, but allow for emergent properties such as surface reconstruction to occur naturally through the

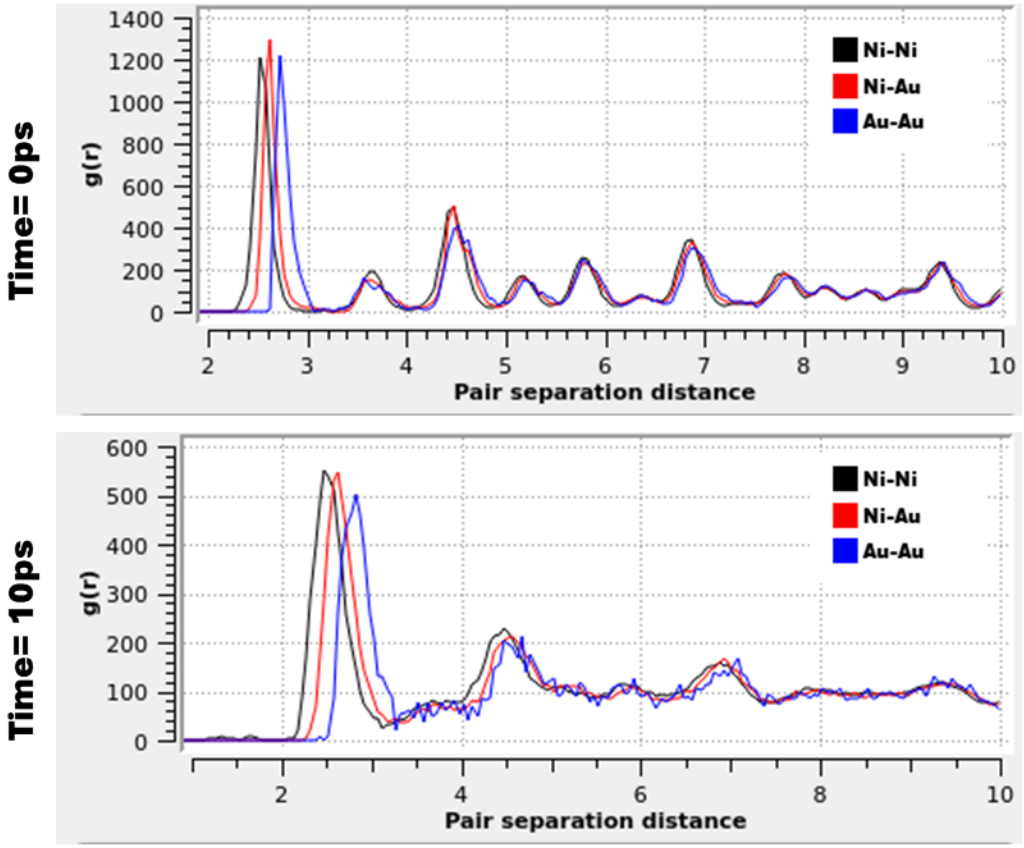


FIG. 3.3. *Radial distribution function results for NiAu nanoparticle before (top) and after (bottom) MD simulation at 1000K.*

simulation. As MD predicts new structures each timestep, extrapolative predictions are inevitable. The models' low errors and consistent performance with both familiar and new structures attests to the model's robustness.

The next step in this project is to compare the stability and objective function errors of the NiAu potential trained on spin polarized DFT data with those of a potential trained on non-spin polarized DFT data for the same elements. This comparison aims to determine if incorporating Ni spin into the training dataset significantly impacts modeling accuracy for the structure of NiAu catalytic nanoparticles. Gaining insight into how magnetism and spin influences the accuracy of Ni-containing ML-IAPs will be valuable for advancing future modeling techniques, particularly given the challenges of incorporating magnetism. After comparing the magnetic and non-magnetic models, we will select one for oxygen integration based on a balance of accuracy and ease of further development. The chosen model will then be used to study adsorbate-driven surface reconstruction on O-covered NiAu catalyst nanoparticles.

Computational modeling, specially ML-IAPs, help capture physical and chemical behaviors of systems that cannot be seen with the naked eye or probed experimentally. The NiAu model developed in this study serves two key purposes: (1) it helps to address a long-standing question in modeling about the role of magnetism in accurately simulating

the behavior of Ni in an ML-IAP, and (2) it serves as a foundational step toward creating a three-element (Ni, Au, O) potential to study adsorbate-driven surface reconstruction on O-covered NiAu catalyst nanoparticles during HOR. This work will advance our understanding of these issues, ultimately paving the way for designing more efficient catalysts for hydrogen fuel cells, aiding the transition away from fossil fuels, and contributing to a carbon-free future.

5. Acknowledgments. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. A.J.R.H. acknowledges support from institutional funds from the Department of Chemical Engineering and Materials Science at Stevens Institute of Technology. I.F. acknowledges funding support from the Science Undergraduate Laboratory Internship (SULI) program and Stevens Institute of Technology Provost's Office Research Funds. Preliminary computations were performed on the Dorothy HPC at Stevens Institute of Technology.

REFERENCES

[1] S. E. Hosseini. Fossil fuel crisis and global warming. *In Elsevier eBooks*, pages 1–11, 2022.

[2] Uchimura M. Wang C. et al. Strmcnik, D. Improving the hydrogen oxidation reaction rate by promotion of hydroxyl adsorption. *Nature Chem*, 5:300–306, 2013.

[3] MyatNoeZin Myint Zhongbin Zhuang Robert V. Forest Qianrong Fang Jingguang G. Chen Wenchao Sheng, Adam P. Bivens and Yushan Yan. Non-precious metal electrocatalysts with high activity for hydrogen oxidation reaction in alkaline electrolytes. *Energy Environmental Science*, 7:1719–1724, 2014.

[4] et al Huynh, Thuan Minh. Hydrodeoxygenation of bio-oil on bimetallic catalysts: From model compound to real feed. *Journal of Sustainable Bioenergy Systems*, 5:151–160, 2015.

[5] et al Lugo-José, Yuliana K. Gas-phase, catalytic hydrodeoxygenation of propanoic acid, over supported group viii noble metals: Metal and support effects. *Applied Catalysis a General*, 469:410–418, 2013.

[6] O. Holton and J. Stevenson. The role of platinum in proton exchange membrane fuel cells. *Platinum Metals Review*, 57:259–271, 2013.

[7] et al. Sun, Yongling. The impact of widespread deployment of fuel cell vehicles on platinum demand and price. *International Journal of Hydrogen Energy*, 36:11116–11127, 2011.

[8] et al. Topalov, Angel A. Dissolution of platinum: Limits for the deployment of electrochemical energy conversion? *Angewandte Chemie International Edition*, 51:12613–12615, 2012.

[9] et al. Alexandr G. Oshchepkov, Alexander. Recent advances in the understanding of nickel-based catalysts for the oxidation of hydrogen-containing fuels in alkaline media. *ACS Catalysis*, 10:7043–7068, 2020.

[10] et al. Davydova, Elena. Stability limits of ni-based hydrogen oxidation electrocatalysts for anion exchange membrane fuel cells. *ACS Catalysis*, 9:6837–6845, 2019.

[11] et al. Davydova, Elena. Hydrogen oxidation on ni-based electrocatalysts: The effect of metal doping. *Catalysts*, 8:454, 2018.

[12] Shuqiao Wang Thomas Robinson Alyssa Hensley Isabella Furrick, Ayodeji Omoniyi. Integration of facet-dependent, adsorbate-driven surface reconstruction into multiscale models for the design of ni-based bimetallic catalysts for hydrogen oxidation. *Chem Cat Chem*, 2024.

[13] Liu C. Su D. Xin H. L. Fang H. Eren B. Zhang S. Murray C. B. Salmeron M. B. Wu, C. H. Bimetallic synergy in cobalt–palladium nanocatalysts for co oxidation. *Nature Catalysis*, 2:78–85, 2018.

[14] Tamura H. Tanaka K. Sasahara, A. Catalytic activity of pt-deposited rh(110) bimetallic surface for no + h2 reaction. *The Journal of Physical Chemistry B*, 101:1186–1189, 1997.

[15] Vandermause J. Van Spronsen M. A. Musaelian A. Xie Y. Sun L. O'Connor C. R. Egle T. Molinari N. Florian J. Duanmu K. Madix R. J. Sautet P. Friend C. M. Kozinsky B. Lim, J. S. Evolution of metastable structures at bimetallic surfaces from microscopy and machine-learning molecular dynamics. *Journal of the American Chemical Society*, 142:15907–15916, 2020.

[16] Marcus P. M. Schwarz K. Mohn P. Moruzzi, V. L. Ferromagnetic phases of bcc and fcc fe, co, and ni. *Physical Review. B*, 34:1784–1791, 1986.

[17] Eschrig H. Perlov A. Y. Oppeneer P. M. Halilov, S. V. Adiabatic spin dynamics from spin-density-functional theory: Application to fe, co, and ni. *Physical Review. B, Condensed Matter*, 58:293–302, 1998.

[18] Svetoslav Nikolov, Mitchell A Wood, Attila Cangi, Jean-Bernard Maillet, Mihai-Cosmin Marinica, Aidan P Thompson, Michael P Desjarlais, and Julien Tranchida. Data-driven magneto-elastic predictions with scalable classical spin-lattice dynamics. *npj Computational Materials*, 7(1):153, 2021.

[19] John Edward Jones. On the determination of molecular fields.—ii. from the equation of state of a gas. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 106(738):463–477, 1924.

[20] Anthony K Rappe and William A Goddard III. Charge equilibration for molecular dynamics simulations. *The Journal of Physical Chemistry*, 95(8):3358–3363, 1991.

[21] Yunxing Zuo, Chi Chen, Xiangguo Li, Zhi Deng, Yiming Chen, Jorg Behler, Gábor Csányi, Alexander V Shapeev, Aidan P Thompson, Mitchell A Wood, et al. Performance and cost assessment of machine learning interatomic potentials. *The Journal of Physical Chemistry A*, 124(4):731–745, 2020.

[22] Chen C. Li X. Deng Z. Chen Y. Behler J. Csányi G. Shapeev A. V. Thompson A. P. Wood M. A. Ong S. P. Zuo, Y. Performance and cost assessment of machine learning interatomic potentials. *The Journal of Physical Chemistry A*, 124:731–745, 2020.

[23] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, 117(1):1–19, 1995.

[24] Aidan P Thompson, H Metin Aktulga, Richard Berger, Dan S Bolintineanu, W Michael Brown, Paul S Crozier, Pieter J In't Veld, Axel Kohlmeyer, Stan G Moore, Trung Dac Nguyen, et al. Lammps-a

flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, 271:108171, 2022.

[25] et al. Thompson, A. P. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285:316–330, 2015.

[26] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403, 2010.

[27] Ngoc Cuong Nguyen and Andrew Rohskopf. Proper orthogonal descriptors for efficient and accurate interatomic potentials. *Journal of Computational Physics*, 480:112030, 2023.

[28] Anton Bochkarev, Yury Lysogorskiy, Sarath Menon, Minaam Qamar, Matous Mrovec, and Ralf Drautz. Efficient parametrization of the atomic cluster expansion. *Physical Review Materials*, 6(1):013804, 2022.

[29] James M Goff, Charles Sievers, Mitchell A Wood, and Aidan P Thompson. Permutation-adapted complete and independent basis for atomic cluster expansion descriptors. *Journal of Computational Physics*, 510:113073, 2024.

[30] R. Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review. B.*, 99, 2019.

[31] Ann E Mattsson, Peter A Schultz, Michael P Desjarlais, Thomas R Mattsson, and Kevin Leung. Designing meaningful density functional theory calculations in materials science—a primer. *Modelling and Simulation in Materials Science and Engineering*, 13(1):R1, 2004.

[32] Y. Mishin. Machine-learning interatomic potentials for materials science. *Acta Materialia*, 214, 2021.

[33] L.B.; Nørskov J.K. Hammer, B.; Hansen. Improved adsorption energetics within density-functional theory using revised perdew-burke-ernzerhof functionals. *Phys. Rev. B*, 59:7413–7421, 1999.

[34] et. al Sikorski, E. Machine learned interatomic potential for dispersion strengthened plasma facing components. *The Journal of Chemical Physics*, 158, 2023.

[35] et al. Jain, Anubhav. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1, 2013.

[36] Wang S. Wang, J. Surface energy and work function of fcc and bcc crystals: Density functional study. *Surface Science*, 630:216–224, 2014.

[37] Vitos L. Kwon S. K. Kollár J. Zólyomi, V. Surface relaxation and stress for 5d transition metals. *Journal of Physics Condensed Matter*, 21, 2009.

[38] Marzari N. Singh-Miller, N. E. Surface energies, work functions, and surface relaxations of low-index metallic surfaces from first principles. *Physical Review B*, 80, 2009.

[39] Ruban A. Skriver H. Kollár J. Vitos, L. The surface energy of metals. *Surface Science*, 411:186–202, 1998.

[40] Ghale K. Rieg C. Dang T. Anderst E. Studt F. Campbell C. T. McEwen J. Xu Y. Hensley, A. J. R. Dft-based method for more accurate adsorption energies: An adaptive sum of energies from rpbe and vdw density functionals. *The Journal of Physical Chemistry C*, 121:4937–4945, 2017.

[41] Wu X. Keong K. Sha, W. Molecular dynamics (md) simulation of the diamond pyramid structure in electroless copper deposits. *Woodhead Publishing Series in Metals and Surface Engineering*, pages 82–103, 2011.

EVENT DETECTION USING NEURAL NETWORKS ROBUST TO STATISTICALLY SIMILAR DISTRACTORS

MARTHA GAHL*, G. WILLIAM CHAPMAN†, SAPAN AGARWAL‡, AND FRANCES S. CHANCE§

Abstract. Remote sensing applications increasingly rely on artificial neural networks for online detection of events of interest. However, industry-driven, off the shelf, models require general purpose hardware which can be power and space inefficient, and often ignore temporal aspects of signals. Driven by these constraints, we design an abstract remote-sensing task and test network architectures that can be deployed on analog neural network accelerators. We find that spatiotemporal layers enable accurate and fast on-line event detection, while being robust to spatially similar but temporally distinct distractors. While each such layer is complex, we find that relatively few layers are required to reach high performance. We then investigate biologically inspired methods which incorporate temporal sparsity and low-precision communication to enable even further power-efficiency, and show similar accuracy. Taken together, these results suggest a hardware platform that incorporates temporal sparsity and enables local recurrence, rather than purely feed-forward components.

1. Introduction. In recent years, machine learning approaches have shown great promise for video-processing applications, such as the detection of specific individuals or monitoring of lightning storms. However, these networks are typically run on energy-intensive general-purpose hardware such as GPUs or FPGAs, making them prohibitively size-weight-and-power (SWaP) constrained for deployment to edge-processing applications. An alternative to such hardware is the use of analog neural network accelerators [15], which utilize passive electronic elements to implement portions of machine learning algorithms in an energy-efficient manner [2], but require simple neural network architectures. Here, we investigate the minimal complexity networks which can solve such remote-sensing tasks, while adhering to the general constraints of analog devices.

Task Domain. We utilize an abstract remote-sensing application in which a network must detect spatiotemporal signals of interest in a timely manner, while ignoring spatially similar distractor signals and other noisy-signals in the background of videos. These additional characteristics prevent an ideal algorithm from simply responding to features such as movement or changes in the input signal. While primarily static tasks, such as object recognition, can rely on access to an entire observation to classify a stimulus, spatiotemporal tasks require online detection in the presence of streaming temporal data. When using a neural network and relying on predictions to make decisions with real world implications, it is vital to know what it is that the network has learned to do. This could mean the difference between a network that can be reliably used in novel situations, and one with unpredictable behavior with out-of-distribution examples [6, 7]. In this work, we train networks to differentiate signals of interest from distractor signals to ensure the network is learning about the signal properties themselves and not using artifacts of the dataset to achieve high performance. This will allow the network, and the learned features, to be more readily transferable to other domains.

Analog ANNs. Analog neural networks have shown minimal degradation to their digital counterparts [16] when trained to be aware of the inherent noise in such devices. One particular analog device of interest is crossbar arrays, which utilize memristors as programmable resistors to accelerate in-memory matrix-vector-multiplication (MVM) required by all neural networks, but which constitute the majority of operations in feedforward networks. How-

*Sandia National Laboratories, mtgahl@sandia.gov
†Sandia National Laboratories, gwchapm@sandia.gov
‡Sandia National Laboratories, sagarwa@sandia.gov
§Sandia National Laboratories, fschanc@sandia.gov

ever, video-like signals, such as may be required for the domains described below, contain temporal as well as spatial information. Processing of spatiotemporal signals typically includes early spatial processing by convolutional networks, followed by late post-processing by recurrent neural networks or post-hoc processing by alternative methods such as Kalman filtering, particle filtering, or template matching [1, 10, 11]. While recurrent architectures have been implemented on memristive devices [8], this requires the use of auxillary circuitry which drastically increases power consumption of the overall network [14]. We therefore seek a network which solves the tasks at hand while staying as close as possible to a pure MVM-based feedforward network.

Architecture constraints. To determine the network components that may enable temporal processing, regardless of hardware feasibility, we first review methods for processing of temporal data. Many common architectures used in object detection tasks are not feasible candidates because of the intensive data movement requirements. Two stage detection models, like Faster R-CNN, require multiple large networks or stored weights beyond what current arrays would allow [13]. Alternative single stage detection models, such as YOLO [12] or single-shot detectors [9], do not detect with temporally changing data. Temporal tracking methods require simultaneous access to all frames of a video, precluding use in on-line detection tasks, and also rely on static spatiotemporal filters rather than allowing dynamic integration of previous and current inputs. In the online detection tasks that we address here, a network may only access external inputs from the current timestep and any information explicitly maintained via internal states or recurrent connectivity. Transformer models, which are increasingly popular for sequence data, can incorporate spatial components as well [4]. However such systems require nonlinear attention mechanisms, continual buffering of inputs via a shifting context trace, and other mechanisms all of which require additional operations beyond the MVM offerings of curent analog hardware.

2. Methods.

2.1. Task. We design a temporal task, in which each trial consists of 100 sequentially presented images (frames). On each timestep the target output is based on whether the event trace of interest (see below) has a signal-to-noise ratio greater than one. By restricting our network architectures below to be purely feedforward or causally recurrent (eg: no bidirectional layers or acausal temporal convolutions), the network only has access to the current input and any history which has been encoded in temporal architecture components. The task of the network is then to minimize total cross-entropy loss on detection of events throughout the trial.

We create a dataset pipeline which allows customization of signals, both distractors and events, with generic python functions, including parameters of the signals and the probability distributions of signal waveforms occuring on a given frame. Signal parameters may include properties such as amplitude, time of onset, temporal duration, and spatial patterns. Table 2.1 shows all tunable parameters, the type of distribtion the values are drawn from, and default bounds for the parameters, and the signal type or types that use the parameter during signal generation.

Our initial experiments demonstrated here utilized sigmoidal and Gaussian temporal waveforms as the events and distractors. These waveforms are placed at a random location in a background image, and then convolved by a point-spread function to give a spatiotemporal waveform. This results in adjacent pixels changing proportionally to their proximity to the true source. The generated waveforms have similar amplitudes and first-derivative values for the event and distractor signals and the signal types are similar enough to prevent simple filtering effects from accurately detecting events.

We consider four circumstances when generating data, which are illustrated in Figure

2.1: 1. Trials with no signals present, 2. Trials with only event signals present, 3. Trials with event signals and distractor signals present, and 4. Trials with only distractor signals present. Numbers 1 and 4 should result in negative answers; the network should be able to identify that neither of these situations includes a signal of interest (event). Numbers 2 and 3 should result in positive answers because there is an event signal present. Even if there is also a distractor signal present, we want the network to identify that there is an event signal.

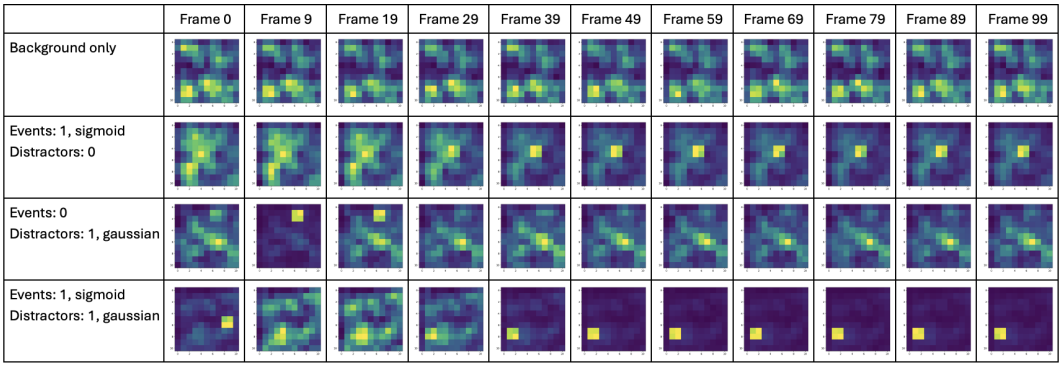


FIG. 2.1. *Exemplar trials along the vertical arrangement, which evolve temporally along horizontal. Each frame is normalized to show more detail.* ***Top*** *Background images are randomly cluttered, as to obfuscate small SNR events.* ***Second*** *Sigmoidal events of interest rise at a randomly chosen frame, and remain on for the remainder of a 100-frame trial.* ***Third*** *Gaussian distractors occur on a random frame and rise for a short time before reaching inflection (frame 20 here) and decaying.* ***Fourth*** *Events of interest and distractors are randomly combined to create a dataset with wide distributions in relative onset and amplitude of events and distractors.*

Parameter	Distribution	Default Minimum	Default Maximum	Signal Type
Location (x and y)	Bivariate Uniform	0	10	Gaussian and Sigmoidal
Amplitude	Uniform Exponent	$log_{10}(1000)$	5	Gaussian and Sigmoidal
Onset	Uniform	4	30	Gaussian and Sigmoidal
Rise	Uniform	0.5	10	Sigmoidal
Width	Uniform	0.5	10	Gaussian
Offset	Uniform	4	30	Gaussian

TABLE 2.1

The tunable parameters used for signal generation. Each parameter type has a distribution that values are drawn from and minimum and maximum bounds for the distribution. Signals that use the parameter for generation are included in the right most column.

2.2. Architecture and Hyperparameter Search. To find an optimal network for this task, we performed an architecture search and a hyperparameter search. In order to most directly map to analog accelerators, we utilize only simplistic multilayer networks, and avoid systems which utilize attention heads, generate anchor boxes, or otherwise rely on non-MVM operations.

Layer types. Our search started with six basic layer-types: dense connections (MLP), convolutional (CNN), recurrent layers (RNN), gated recurrent units (GRU), long short term memory (LSTM) layers, and convolutional-recurrent layers (CRNN). Figure 2.2 shows these architecture building blocks [5, 18]. These layer types are combined into deeper networks, either by utilizing the same type of layers in sequence, or mixing various layer types.

Dense layers preserve no spatial information and do not allow for temporal effects, but offer a greater number of trainable parameters. Convolutional layers preserve spatial information by utilizing a convolutional operator and shared weights among all pixels. Such convolutional layers may extract the point-spread structure of our waveforms, but can not extract temporal information that differentiates our event and distractor signals. Recurrent layers (RNN, GRU, and LSTM) utilize recurrent connections to preserve information from previous timesteps, allowing detection of temporal patterns, rather than simply spatial structure.

Preliminary experiments showed that networks without any recurrent connections (MLP, CNN) can perform well, particularly if they preserve spatial information, but were less stable in their predictions, and demonstrated a greater delay in detecting signals 3.1. Without temporal information, a network likely relies on thresholding intensity for predictions, which is not reliable when distractors are present. Networks that did not preserve any structural information (RNN, LSTM) generally performed poorly, as they lacked the ability to utilize the point-spread function to filter out background changes.

This suggests an architecture best suited for this task would perform both spatial and temporal processing. We therefore focused our search on network structures that contain both spatial and temporal processing, either by combinations of spatial and temporal layers, or by the introduction of a single spatiotemporal layer. The convolutional-recurrent layer follow the governing equations:

$$Y(X, Y(t-1)) = F(W_{ih} * X + W_{hh} * (Y(t-1)))$$

where $*$ indicates the convolutional operator, W_{ih} is the set of feedforward convolutional filters, W_{hh} is the set of recurrent convolutional filters, and F is the nonlinear activation function of choice. Such networks preserve spatial invariance, but detect spatially structured *changes through time*.

Hyperparameters. Constrained by the layer-types above, our hyperparameter search focused on three values: 1. number of layers in the architecture, 2. confidence threshold for positive identification, and 3. learning rate. Increasing the number of layers can allow an architecture to approximate a more complex function, but at the risk of overfitting to training data, meaning a network will perform poorly on testing data. The confidence threshold determines how confident a network must be before identifying an event. For example, with a confidence threshold of 50%, a network could determine the likelihood of a frame containing an event to be 51%, and the likelihood of the frame not containing an event to be 49%, and the frame would be classified as containing an event. Increasing this value means the network is more confident that a frame will contain an event. Conversely, a very large confidence threshold might mean the network not making positive predictions soon after signal onset and waiting for a stronger signal before positively identifying a frame. This can increase the delay in response. Confidence threshold is only used at inference time when network outputs are translated into actual predictions. Finally, learning rate determines the magnitude of the network weight updates in response to a penalty. Too small of a learning rate can cause the network to get stuck in a local minimum in the loss landscape. Too large of a learning rate can cause the network to oscillate in performance and fail to converge, particularly in heavily recurrent networks.

Activation functions. In our initial set of experiments, all networks use the ReLU activation function. Due to the desire to deploy on low-energy edge-based hardware, we also investigate binary activation functions. Previous work [3] has shown that recurrent layers cannot be trained for binary activation, and we therefore implement binary activation recurrent layers as leaky-integrate and fire (LIF) units. LIF units are a biologically inspired architecture in which each unit contains a local real-valued state that persists over time. The units communicate with each other only by binary spikes when reaching a threshold. They follow the equation:

$$\begin{aligned} \tau_v \frac{dv_L(t)}{dt} &= -v_L(t) + \sum_{n \epsilon A} W_{nL} S_n(t) \\ S_L(t) &= v_L(t) \geq 1 \\ v_L(t+1) &= \begin{cases} 0, & \text{if } v_L(t) + \frac{dv_L(t)}{dt} > 1 \\ v_L(t) + \frac{dv_L(t)}{dt}, & \text{otherwise} \end{cases} \end{aligned} \tag{2.1}$$

Where $v_L(t)$ indicates a local real-valued internal state that is not communicated to other units or layers, τ_v is a corresponding time constant with the value 4. S_L indicates transient spikes which are transmitted to other units, and tend to be temporally sparse.

2.3. Performance Metrics. In an event detection network, there are a variety of metrics that are useful in creating a complete picture of network performance. We report both frame-based and trial-based accuracy. Frame-based accuracy is simply the portion of frames across all validation trials which are correctly classified as containing an event or not. Trial accuracy reports the portion of trials on which the network correctly detects an event, on at least one frame. The degree to which these two metrics differ is caused by the detection delay, which measures how many frames after signal onset a network takes to respond with a positive output. This measurement provides insight to the sensitivity of the network to signals, as frames just after signal onset have a low-SNR, while later frames at the peak of signal waveforms may be orders of magnitude greater amplitude than background values. This is important, particularly for online, real time event detection because it dictates the onset of other processes involved in positive event identification, like localization of the event or response to the event.

Models are trained using binary cross entropy loss on a frame-by-frame basis, which implicitly optimizes for delay, recall, and precision. Recall measures the ability to select all true positive frames, as such missing false negatives decrease this metric. Precision measures the ability to only select true positive frames, and is penalized by false positives. Poor recall and precision are equally and explicitly penalized in the network. Anytime a frame is incorrectly identified, whether it be a false positive or a false negative, the network receives a direct penalty. Long delay is implicitly penalized in the network. The longer it takes to detect an event, the more frames are incorrectly identified, and the more penalty the network receives. This forces the network to try to decrease the delay between signal onset and first prediction.

2.4. Experiments. For all experiments described here, create trials which are 11x11 pixels and 100 frames long. The training set consisted of 10,000 trials equally (2,500 per case) chosen to have no signals, distractor only, event only, or event and distractor present. The events were sigmoidal and the distractors were Gaussian, and had a shared amplitude range from which individual amplitudes were sampled. An independent validation dataset of 10,000 trials was generated following the same splits and parameters.

While trials were equally split to contain or not contain an event, the loss function is based on frames. Due to the temporal nature of the events, a variable number of frames in each trial which contained an event was random, creating a imbalance in the number of frames which contained events. To account for this, we weighted our loss function such that positive frames were penalized more strongly, than negative frames. This weight was calculated as the ratio of positive to negative frames, resulting in a potentially equal loss across negative and positive frames. We used binary cross entropy loss and an Adam optimizer, and backpropagation through time to incorporate the temporal relationships among frames.

3. Results.

	Frame Accuracy (%)	Trial Accuracy (%)	Average Delay (frames)
1 layer MLP	84.71	71.50	11.45
2 layer MLP	51.99	44.81	8.38
1 layer CNN	86.28	79.5	13.97
2 layer CNN	90.02	79.88	11.44
1 layer RNN	46.84	28.83	-0.88
2 layer RNN	46.78	28.81	-0.89
1 layer LSTM	75.05	59.95	11.06
2 layer LSTM	79.64	64.45	10.96
1 layer CRNN	88.27	62.65	8.73
2 layer CRNN	90.99	83.28	10.44
3 layer CRNN	91.34	83.68	10.11

TABLE 3.1

Frame accuracy, trial accuracy, and average delay for simple architectures, each composed of a single type of network.

3.1. Real Valued. Our initial set of experiments compared basic network architecture types, utilizing one- and two-layer versions of MLP, CNN, RNN, LSTM, and CRNN, with learning rates of 0.001. The results for these initial tests are shown in Table 3.1, reported with confidence thresholds of 50%. The frame accuracy is always higher than the trial accuracy, which is due to a stringent definition of trial accuracy. In a trial without an event, if 99 frames are predicted to not contain an event and one frame is predicted to contain an event, the trial will count as a miss. Despite the frame accuracy in this case being 99%, the trial accuracy for that single trial is 0%. Feedforward networks (MLP, CNN) generally have longer delays than recurrent layers (LSTM, CRNN), indicating that they require a higher change in pixel intensity before determining that changes are due to an event and not noise or a distractor. RNN networks have a mean negative delay, indicating the presence of false positive *frames*, or the network making a positive prediction before the signal onset. The MLP, RNN, and LSTM models, which do not preserve spatial information, have lower frame accuracies and lower trial accuracies than models that do preserve spatial information (CNN and CRNN), indicating that maintaining spatial structure is important to differentiate

high frequency noise from structure signals. This data suggest lower average delays require recurrent architectures and high accuracies require spatially aware architectures.

	Frame Accuracy (%)	Trial Accuracy (%)	Average Delay (frames)
1 layer CNN and 1 layer CRNN in series	90.23	69.88	10.06
1 layer CNN and 2 layer CRNN in series	90.57	78.99	10.30
2 layer CNN and 1 layer CRNN in series	89.85	81.06	12.43
2 layer CNN and 2 layer CRNN in series	89.26	85.47	15.94
1 layer CRNN and 1 layer CNN in series	88.91	61.50	7.73
1 layer CRNN and 2 layer CNN in series	88.94	66.87	9.45
2 layer CRNN and 1 layer CNN in series	89.94	78.78	10.76
2 layer CRNN and 2 layer CNN in series	89.19	74.57	9.37
1 layer CNN and 1 layer GRU in series	88.77	77.98	11.70
1 layer CNN and 1 layer CRNN in parallel	88.58	74.39	10.88
1 layer CNN and 2 layer CRNN in parallel	90.11	75.98	10.24
1 layer CNN and 3 layer CRNN in parallel	90.94	82.51	10.91
2 layer CNN and 1 layer CRNN in parallel	90.23	78.51	10.56
2 layer CNN and 2 layer CRNN in parallel	90.04	80.90	11.69

TABLE 3.2

Frame accuracy, trial accuracy, and average delay for more complex architectures, each composed of two network types, either in series or in parallel.

Based on these results, we constrain our search to architectures with spatial and temporal processing. Some of these architectures are single modules that have both spatial and temporal processing, like the CRNN, and others are combinations of architectures whose outputs are combined to create representations that include both spatial and temporal information. Table 3.2 lists the architectures we considered in this phase of the search and their resulting performances. All of these experiments had confidence scores of 50% and learning rates of 0.001.

The models in Table 3.2 all include some spatial processing and some temporal processing. These models generally produce higher accuracies than the models in Table 3.1 that did not have both spatial and temporal processing. We then looked at the loss plots for these models to understand how they were learning throughout training. We want the loss to decrease consistently and oscillate as little as possible. This is indicative of a stable model that is learning useful features that are consistently helpful for the task. In Figure 3.1 are loss plots that correspond to networks in Table 3.2. All of the models show much oscillation in the loss values, suggesting that the learning rate for all models should be lowered. The loss values for the networks with CNN and CRNN in series show little difference between the loss values at the beginning and end of training. These models are able to achieve reduced loss at different points in training, but the overall flat trend and large oscillations suggest the epochs with low loss are good guesses by the models and do not correspond to learning.

Models that don't show consistent learning are not considered as they are likely to have difficulty adapting to data outside of the training data distribution. Despite having low loss at a few points in training, the model is not learning useful featurers that can be transferred to other data.

Based on these two broad searches, we found three architectures that performed well, had relatively low delays, and demonstrated consistent learning during training: 3-layer CRNN, 1-layer CNN in parallel with 1-layer CRNN, and a 1-layer CNN in parallel with a 3-layer CRNN. Along with a 1-layer CNN followed by a 1-layer GRU as a baseline [17] for comparison, these four architectures make up the list of candidate architectures that we explore more fully. Notably, the baseline architecture is the only candidate architecture that does not include a CRNN. The baseline network we call Architecture 1, the 3-layer CRNN we call Architecture 2, the 1-layer CNN in parallel with 1-layer CRNN we call Architecture 3, and the 1-layer CNN in parallel with a 3-layer CRNN we call Architecture 4. These four architectures are shown in Figure 2.3. We continued our hyperparameter search, optimizing for performance in Architectures 2-4, and found an learning rate of 0.0001, weighted loss, and a confidence level of 60% result in the best performing networks. We trained each of these network architectures five times with random weight initializations and report averaged performance in Table 3.3.

Architecture 1 has the longest delay of all architectures, though the difference between Architecture 1 and the architecture with the shortest delay, Architecture 2, is just over two frames. The average delays are clustered around 10 frames. Interestingly, the average length of a Gaussian signal (a distractor) in the dataset is approximately 18 frames. All models appear to learn to wait to make a decision about whether or not a location of rapidly increasing intensity is an event until after the expected inflection point of a distractor signal. This suggests that all models are able to learn about the shape of the signals and use this information to accurately distinguish events from distractors. This invites further questions about the impact of the signal types on model performance, which we discuss briefly in an outline of our future steps in the conclusion.

All four architectures, even the baseline, are able to consistently achieve a a high recall value, indicating a lack of false negatives. Precision values are lower across all architectures. Taken together, the recall and precision values indicate that the models very rarely miss an event, but are more likely to classify false positives. The frame accuracy, like with the simple architectures, is always higher than the trial accuracy.

Architecture 2 and Architecture 4 differ only in the addition of a 1-layer CNN. Despite this architectural difference between the networks, the difference in their performance across metrics is extremely minimal. This suggests that the addition of the 1-layer CNN does not provide the model with any additional useful spatial processing for this task. In theory, the CRNN should be able to use a subset of its channels as a feedforward CNN. This means that with a sufficiently large CRNN, the addition of a 1-layer CNN is redundant.

3.2. Spiking. Based on the results shown in Table 3.3 that suggest little if any performance benefits from adding parallel spatial processing to Architecture 2 to create Architecture 4, we test only Architectures 1-3 with binary activation functions. The results from these experiments are shown in Table 3.4. All three networks show significantly decreased performance across frame accuracy, trial accuracy, precision, and recall as compared with the same architectures using the ReLU activation function. However, the average delay values for the spiking networks are slightly reduced. Similarly to the ReLU networks, each binary activation network achieves a higher recall than precision, with a significant difference between Architecture 1, the baseline, and Architectures 2 and 3. Architecture 1 has a negative average delay, meaning before the event has occured it is classifying frames as hav-

	Frame Accuracy (%)	Trial Accuracy (%)	Average Delay (frames)	Trial Precision	Trial Recall
Architecture 1: 1 layer CNN and 1 layer GRU in series	87.77	75.91	11.42	0.690	0.942
Architecture 2: 3 layer CRNN	90.95	75.49	9.25	0.689	0.934
Architecture 3: 1 layer CNN and 1 layer CRNN in parallel	88.37	74.43	10.95	0.678	0.931
Architecture 4: 1 layer CNN and 3 layer CRNN in parallel	90.88	78.01	9.36	0.713	0.938

TABLE 3.3

Further investigation of chosen architectures. CRNN-based networks show the highest accuracy and precision, along with low recall. Adding parallel spatial processing adds minimal or no additional performance.

	Frame Accuracy (%)	Trial Accuracy (%)	Average Delay (frames)	Precision	Recall
Architecture 1: 1 layer CNN and 1 layer GRU in series	46.78	28.81	-0.89	0.366	0.576
Architecture 2: 3 layer CRNN	51.99	44.81	8.38	0.473	0.896
Architecture 3: 1 layer CNN and 1 layer CRNN in parallel	52.19	45.67	9.11	0.473	0.913

TABLE 3.4

SNN performance for chosen network architectures. All architectures are able to achieve 100% accuracy, though the negative delay for Architecture 1 suggests errors in predicting events too early.

ing events. We interpret this as trial error because the network is not classifying accurately based on the information available.

4. Conclusion. In this work we investigate hardware constrained network architectures for robust spatiotemporal event detection. We find that convolutional-recurrent networks consistently outperform sequential spatial-then-temporal networks, both in overall accuracy and by responding more quickly to changes in inputs. Incorporating biologically inspired, and temporally sparse and low power, activation functions into these networks shows the same general trend. Currently the performance of spiking models is reduced compared to real-valued activations, but this may be mitigated in the future by optimizing the time constants of the LIF units.

4.1. Future Work. The results of this work prompt additional questions about the relationships between network hyperparameters, signal properties, and alternative activation functions.

Confidence. We performed inference with two confidence levels: 50% and 60%. We found models made fewer mistakes with a confidence of 60%, but we have not done an indepth analysis of how confidence level changes performance. As this is preliminary work, we intend to look at the AUC for different metrics, including accuracy, precision, recall, and delay, for a broader array of confidence values. This can inform optimal confidence level based on the metric that is most important to a client or task.

Signal types. We generated one dataset and ran all of our experiments on those data. Our dataset generation pipeline was made to be modular, and there are more types of signals, and changes to signal properties, that we could readily use. Some of the results suggested that the models were learning properties about the signals in the dataset. For example, we found average delay in our ReLU models to be about half of the average width of distractor signals. By changing signal parameters and signal types, we can quanitfy the extent to which models are able to learn about a given signal and how useful those learned features are in out-of-distribution stimuli.

Activation functions. Our results showed significant differences in model performance just from changing activation functions. We plan to explore more bio-inspired activation functions in an attempt to further decrease the delay between event onset and event detection.

REFERENCES

[1] S. Afshar, A. P. Nicholson, A. Van Schaik, and G. Cohen, *Event-Based Object Detection and Tracking for Space Situational Awareness*, IEEE Sensors Journal, 20 (2020), pp. 15117–15132.

[2] S. Agarwal, T.-T. Quach, O. Parekh, A. H. Hsia, E. P. DeBenedictis, C. D. James, M. J. Marinella, and J. B. Aimone, *Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding*, Frontiers in neuroscience, 9 (2016), p. 484.

[3] G. W. Chapman, C. Teeter, S. Agarwal, T. P. Xiao, P. Hays, and S. S. Musuvathy, *Biological dynamics enabling training of binary recurrent networks*, in 2024 Neuro Inspired Computational Elements Conference (NICE), 2024, pp. 1–7.

[4] A. H. de Oliveira Fonseca, E. Zappala, J. O. Caro, and D. van Dijk, *Continuous spatiotemporal transformer*, in Proceedings of the IEEE International Conference on Machine Learning, IEEE, 2024, pp. 123–130.

[5] I. Developer, *Machine learning and deep learning architectures.* `https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/`. Accessed: 2024-08-30.

[6] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, *Shortcut learning in deep neural networks*, Nature Machine Intelligence, 2 (2020), p. 665–673.

[7] A. Kurakin, I. Goodfellow, and S. Bengio, *Adversarial examples in the physical world*, 2017.

[8] C. Li, Z. Wang, M. Rao, D. Belkin, W. Song, H. Jiang, P. Yan, Y. Li, P. Lin, M. Hu, et al., *Long short-term memory networks in memristor crossbar arrays*, Nature Machine Intelligence, 1 (2019), pp. 49–57.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *Ssd: Single shot multibox detector*, in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part I 14, Springer, 2016, pp. 21–37.

[10] T. J. Ma and R. J. Anderson, *Remote sensing low signal-to-noise-ratio target detection enhancement*, Sensors, 23 (2023), p. 3314.

[11] V. Patraucean, A. Handa, and R. Cipolla, *Spatio-temporal video autoencoder with differentiable memory*, Sept. 2016.

[12] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, arXiv, (2018).

[13] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2016.

[14] M. Spear, J. E. Kim, C. H. Bennett, S. Agarwal, M. J. Marinella, and T. P. Xiao, *The impact of analog-to-digital converter architecture and variability on analog neural network accuracy*, in Proceedings of the IEEE International Conference on Neural Networks, IEEE, 2024, pp. 123–130.

[15] Q. Xia and J. J. Yang, *Memristive crossbar arrays for brain-inspired computing*, Nature Materials, 18 (2019), pp. 309–323.

[16] T. P. Xiao, B. Feinberg, C. H. Bennett, V. Agrawal, P. Saxena, V. Prabhakar, K. Ramkumar, H. Medu, V. Raghavan, R. Chettuvetty, et al., *An accurate, error-tolerant, and energy-efficient neural network inference engine based on SONOS analog memory*, IEEE Transactions on Circuits and Systems I: Regular Papers, 69 (2022), pp. 1480–1493.

[17] T. P. Xiao, W. S. Wahby, C. H. Bennett, P. Hays, V. Agrawal, M. J. Marinella, and S. Agarwal, *Enabling high-speed, high-resolution space-based focal plane arrays with analog in-memory computing*, in 2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits), 2023, pp. 1–2.

[18] K. Zainal Mokhtar and J. Mohamad-Saleh, *An oil fraction neural sensor developed using electrical capacitance tomography sensor data*, Sensors (Basel, Switzerland), 13 (2013), pp. 11385–406.

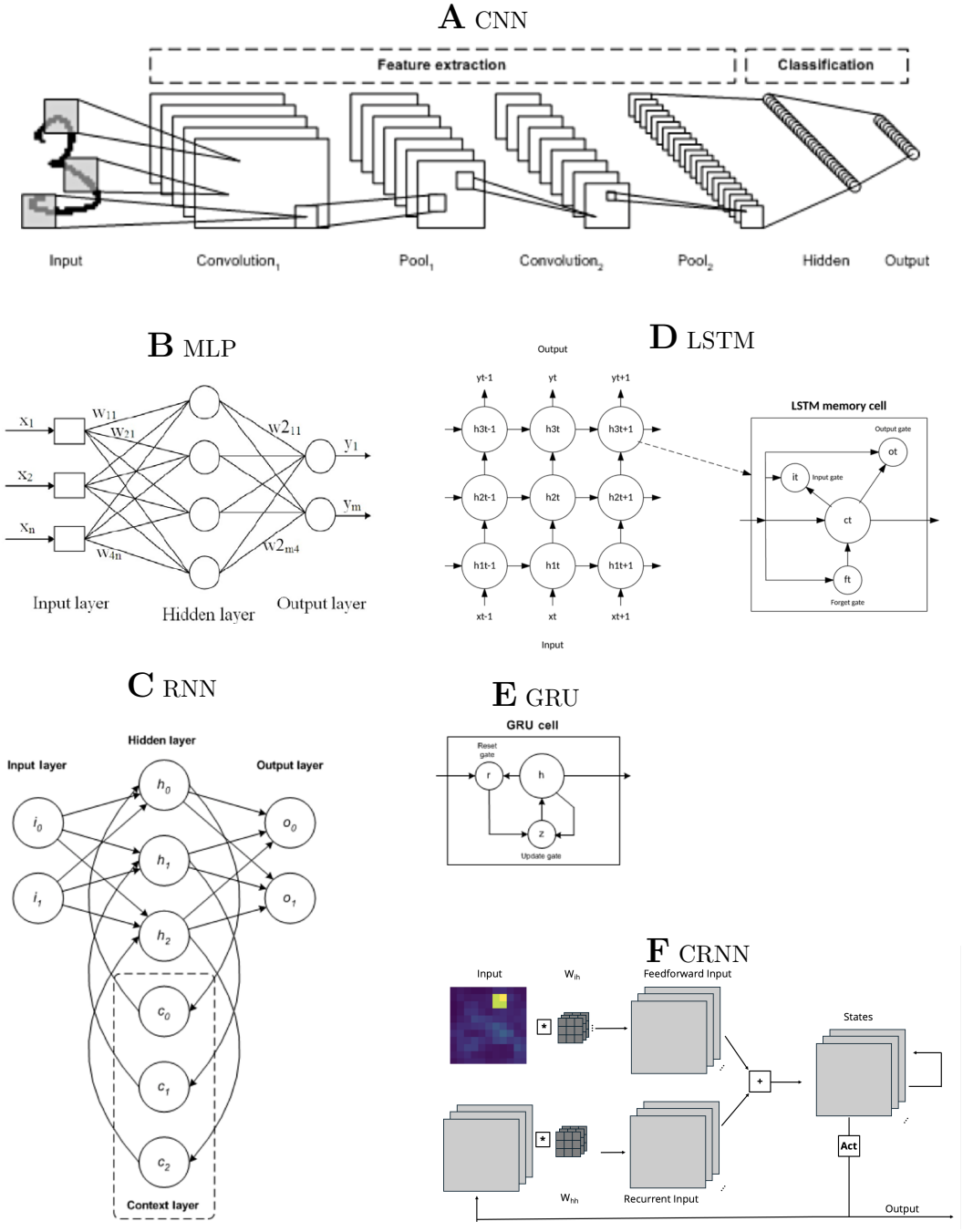


FIG. 2.2. *Architecture building blocks:* ***A*** *Convolutional layers incorporate spatial invariance by utilizing shared weights between neighbors across all pixels [5].* ***B*** *Perceptron layers allow nonlinear mixing of input components, but do not include spatial or temporal invariance [18].* ***C*** *RNN layers incorporate feedforward activity from their input, but also a "context" which represents previous time-steps outputs [5].* ***D*** *LSTM layers: LSTM layers are a type of recurrent network. They have three gates (input, output, and forget) and are meant to minimize occurance of vanishing gradients using a cell state [5].* ***E*** *GRU layers: GRU layers are also reccurent layers, but with only two gates (update and reset) meaning they have fewer parameters than an LSTM [5].* ***F*** *CRNN layers incorporate both spatial and temporal invariance, by utilizing independent feedforward (top pathway) and recurrent (bottom pathway) convolutional weights.*

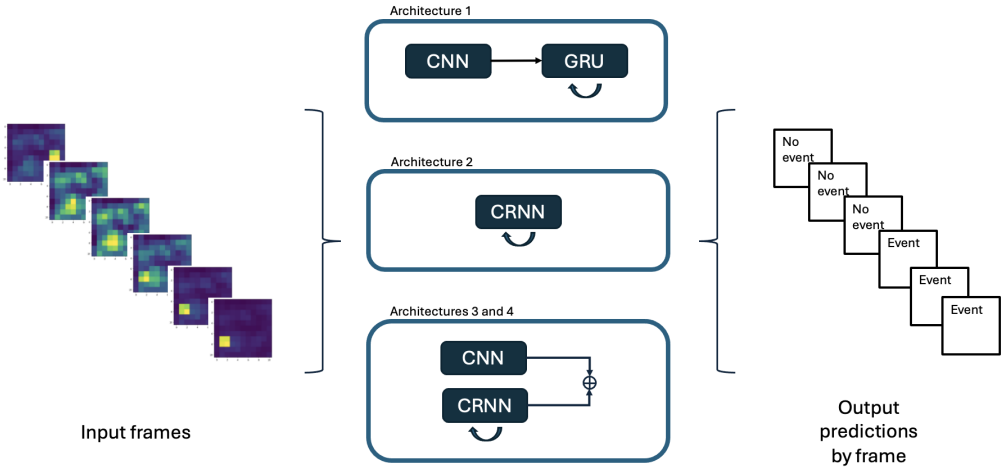


FIG. 2.3. *Illustration of data flow through time.* ***Left*** *Input frames, which are spatially structured and contain sensor magnitude readings, arrive throughout time (frames).* ***Right*** *All architectures classify each frame as containing an event of interest or not, purely based on the preceding frames, or only the current frame in the case of CNN or MLP layers.* ***Top*** *Sequential spatial-then-temporal network process frames in a purely spatial manner before integrating outputs through time.* ***Middle*** *CRNN layers integrate local changes on each timestep.* ***Bottom*** *Parallel architectures combine multiple methods and summate the resulting activations on each timestep.*

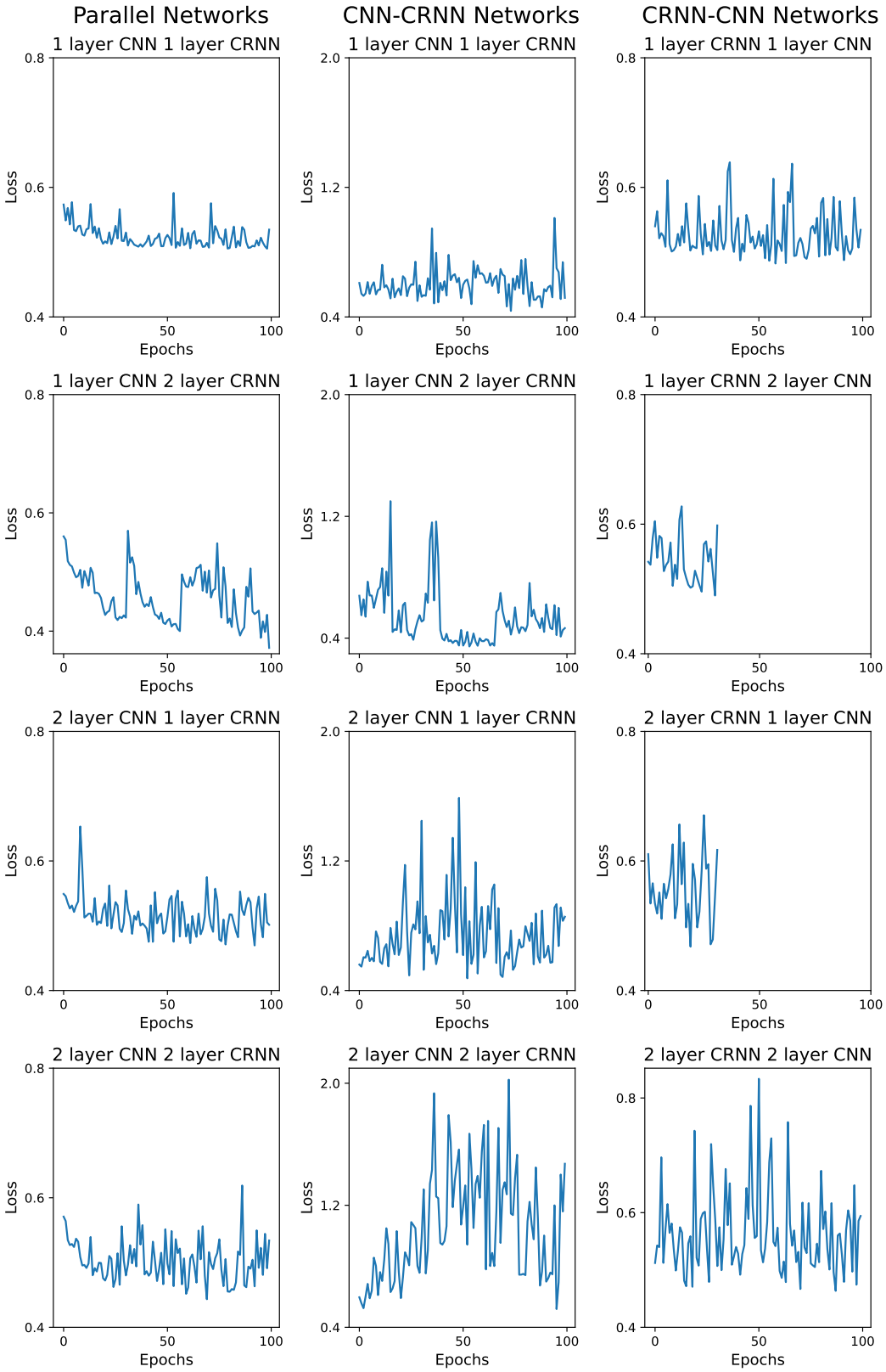


FIG. 3.1. *Loss throughout training for models shown in Table 3.2. X axis is training epochs and y axis is the loss.* **Left** *Models with a CNN and a CRNN in parallel. Both models recieve the stumli and their representations are added at the end.* **Center** *Models with a CNN and CRNN in series. Only the CNN recevies the input stimuli. The output of the CNN is the input to the CRNN.* **Right** *Reverse of the center column.*

MACHINE LEARNED INTERATOMIC POTENTIAL DEVELOPMENT ACCELERATED VIA LARGE-LANGUAGE MODELS FOR NICKEL-GOLD

JENNIFER D. GONZALES-PASION*, MITCHELL WOOD†, AND ALYSSA J. HENSLEY‡

Abstract.
The widespread use of hydrogen alkaline fuel cells, a clean renewable energy source, is hindered by its reliance on scarce and expensive noble metal catalysts. Existing literature indicates that sustainable and cheaper gold-promoted nickel-based catalysts show promising catalytic activity for the hydrogen fuel cell. However, two challenges exist: (1) current computational modeling techniques are unable to reach the necessary length- and time-scales and (2) due to nickel displaying ferromagnetic properties, there is an open question on how magnetism affects model accuracy. To fully understand the chemistry and physics of the nickel-gold system, this study aims to develop a nonmagnetic interatomic potential for gold-promoted nickel catalysts. The methodology for building an accurate machine learned interatomic potential involved the development of training data, conversion of training data points to descriptors, tuning and optimizing sets of parameters to decrease accuracy errors, and modeling the potential against various structures. This nonmagnetic potential is intended to be compared against a magnetic informed potential to assess the impact magnetism has when developing an accurate gold-promoted nickel catalyst model. The current nonmagnetic potential demonstrated low errors while successfully being able to model stable nickel and gold surfaces. This signifies a promising step towards building accurate potentials for the desired system and advancing towards cost effective hydrogen fuel cells.

1. Introduction. The need for a clean renewable energy source stems from the current, primary practice of burning fossil fuels for energy production. When burning fossil fuels, carbon dioxide–a greenhouse gas which is the leading cause of climate change–is released in addition to energy. An alternative energy production route growing in interest are hydrogen fuel cells. These hydrogen fuel cells require heterogeneous catalysts for the hydrogen oxidation reaction (HOR) to occur, a chemical reaction that causes molecular hydrogen to break into its atoms and release electrons before combining with molecular oxygen to form water[4]. The flow of electrons drives the fuel cell to generate power without releasing harmful chemicals to the atmosphere. The choice of catalyst here is key, as without it the HOR would occur at much slower rate deeming the hydrogen fuel cell inefficient. Platinum and iridium are the current standard catalysts for this device, but they prevent the mass production of hydrogen fuel cells due to their high cost and scarcity[4]. Research has shown that nickel (Ni)-based catalysts have great stability in the alkaline media of a hydrogen fuel cell and emphasized promising electrochemical behavior[14]. When doped with a secondary promoter metal, like gold (Au), we can further enhance the properties of the catalyst, letting our desired material become a Ni-Au alloy[8].

As catalytic reactions occur at the nano-scale, experiments cannot fully resolve key information about the working catalyst, such as atomic-level surface reconstruction and active site formation. To tackle this problem requires the use of atomistic molecular dynamics to uncover the mechanisms of this evolving catalytic activity. A further complication that exists in modeling Ni catalysts is the ferromagnetic nature of metallic Ni. Ferromagnetism is a class of magnetism, a phenomenon described as the motions of electronic charges that produces attractive and repulsive forces between objects[17]. The presence of this ferromagnetism poses a critical question: can the dynamic surface structures of Ni-based catalysts be accurately predicted without accounting for the metal's magnetic properties? Our approach will be to utilize molecular dynamics simulations to directly observe surface reconstruction during the HOR as these methods are well suited for the length and timescale of this

*Stevens Institute of Technology, jgonzal7@stevens.edu
†Sandia National Laboratories, mitwood@sandia.gov
‡Stevens Institute of Technology, ahensley@stevens.edu

phenomena.

To address this question for Au-promoted Ni catalysts, we need to first address the inputs required for any molecular dynamics (MD) simulation, an interatomic potential. The main difference between *ab-initio* MD (AIMD) and its classical MD counterpart is that forces on atoms are taken either from gradients of a self-consistent wavefunction (AIMD), or from a reduced order model solely based on atomic positions (classical MD). AIMD is very accurate but very computationally expensive, scaling as the cube of the atom count. Meanwhile, the cost of classical MD is linear with atom count, largely due to the use of a radial cutoff of atom interactions. There has been a large emphasis on machine learning methods used in MD[9, 21], and we aim to capitalize off this for our work on Ni-Au catalyst design. Therefore, we will build a nonmagnetic Ni-Au machine learned interatomic potential (ML-IAP), a complex function that describes the forces and energies based on local atomic positions. The nonmagnetic Ni-Au potential developed here will be compared to a similarly created magnetic potential in the near future to gain further insight on the effects of magnetism on large-scale MD simulation results. Here, we built a nonmagnetic potential using a diverse training data set containing only nonmagnetic structures. A computational tool was then employed to fit initial potentials to the given training data set and output force and energy error metrics for accuracy assessment. Subsequently, the potential was further optimized using a genetic algorithm that applied different data point sampling methods to find the best settings for the ML-IAP to minimize errors. Once the potential was deemed optimized, based on the errors to known properties, it was then tested against a variety of structures under different reaction conditions using MD simulations through the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [15, 18]. During these simulations, our goal was to achieve stable structures, signifying an accurate nonmagnetic potential for Ni-Au catalysts.

2. Methods. A ML-IAP is constructed in three key components: 1) a training set that contains ground truth values of total energy and forces per atom, 2) a set of descriptors (features) that serve as input to 3) a model form that outputs per-atom energy and force. Constructing the training set that captures the desired material properties and end use cases is non-trivial, and will be explained in detail in subsequent sections. For simplicity we will adopt a linear model form versus a more complex neural network, though the same procedures outlined here apply to the training of either model form. Which leaves the choice of a descriptor set, where special attention to the accuracy versus computational cost of constructing this basis set is needed. Many suitable options exist in the literature, and our search is narrowed by their availability in LAMMPS[18]. A zoo of acronyms exist to define these descriptor sets; Spectral Neighborhood Analysis Potential (SNAP)[19], Gaussian Approximation Potential (GAP)[2], Proper Orthogonal Descriptors (POD)[11]. The 'parent' set to all of these atom-centered bases is the Atomic Cluster Expansion (ACE) method[3, 6]. Subsequent sections will detail the choices made to construct the training set and optimize both descriptor and model form hyper parameters.

2.1. Generation of Training Data. The initial set of training data was generated on the Dorothy high-performance computing cluster (HPC) at Stevens Institute of Technology (SIT). Two types of training data were generated: domain and beyond domain knowledge structures. For domain knowledge structures, we used the open access database Materials Project[7] to include bulk Ni (fcc, bcc, hcp) and bulk Au (fcc, bcc, hcp). Additionally, ground state structures for Ni, Au, Ni_3Au alloy, and $NiAu_3$ alloy, were also accessed through Materials Project and are the most stable configurations of both element types. We also created defect structures of pure Ni and Au with single-atom metal vacancies and substitutions of one element type replacing the other. Beyond domain knowledge structures were

generated using the Universal Structure Predictor: Evolutionary Xtallography (USPEX). USPEX can generate randomized stable and metastable structures by using input parameters such as chemical composition, the range of the number of atoms that can exist in a structure, and the ion distance between the atoms[12, 13, 10]. We instructed USPEX to create three sets of structures, 100 pure Au, 100 pure Ni, and 100 Ni-Au alloys. Each set contained a range of 4-12 atoms and a minimum ion distance of 2.0 Å, with the chemical composition changing depending on the set that is being calculated.

Once structures were generated, the energies and forces needed for ML-IAP training were obtained via the Vienna Ab Initio Simulation package (VASP), a quantum mechanics simulation code that solves for the electronic structure with Density Functional Theory (DFT). All DFT calculations were single point and non-spin polarized. Electron-electron interactions were captured using the Revised Perdew-Burke-Ernzerhof (RPBE) functional. A plane-wave basis set was used with an energy cutoff of 400 eV. Electron smearing was modeled with the Methfessel-Paxton (N=1) approach, with a smearing width of 0.1 eV. The first Brillouin zone of all DFT calculations was sampled using a k-point resolution of 0.02 in units of $2\pi\text{Å}^{-1}$. The electronic convergence criteria was set to 10^{-4} eV.

To rapidly generate data that spans a large descriptor space, AIMD simulations were employed using VASP on the Sandia HPC cluster, Amber. AIMD simulations move atoms through space based on the forces calculated and the quantum mechanical level, a set temperature, and Newton's laws of motion. Here,the canonical ensemble was used (e.g., constant number of atoms, volume, temperature), and each simulation was run with 1.0 femtosecond timesteps at two different temperatures (e.g., 300 K and 1000 K). AIMD simulations were performed on all domain knowledge structures at various temperatures ranging from 300 K-2000 K. These values were selected based on the stability and melting temperatures of Ni and Au.

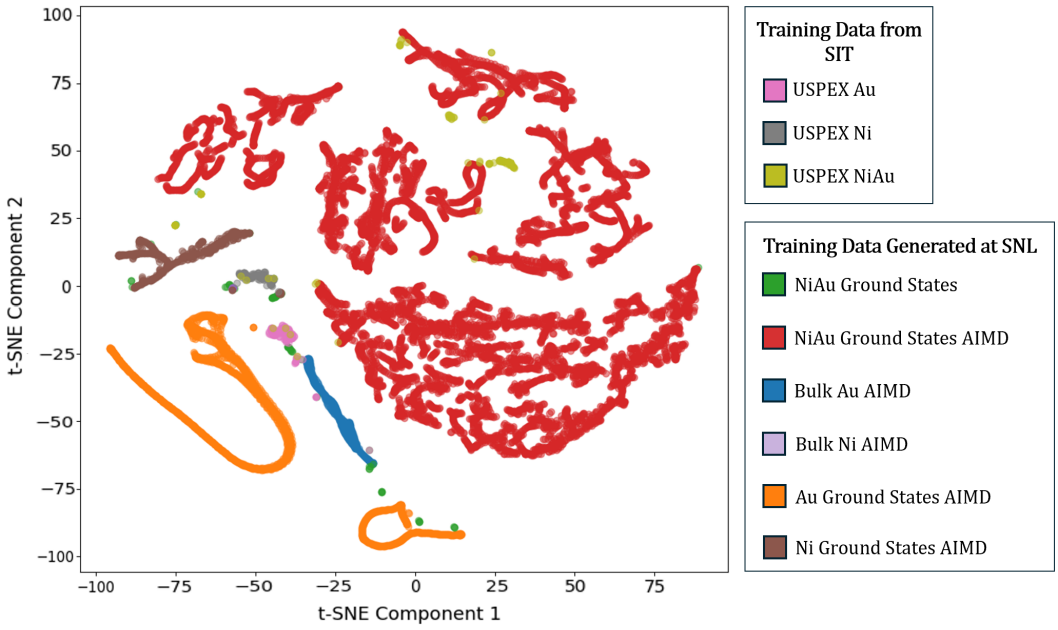


FIG. 2.1. *t-SNE projection of complete training data set plotted against the descriptor space. Note the large volume of descriptor space mapped by AIMD versus other structure generation methods.*

To visualize the diversity and amount of training data, we produced a plot called the

t-distributed Stochastic Neighbor Embedding (t-SNE). This technique takes the given high-dimensional data set and graphs it against a two-dimensional plot while preserving as much of the original information [20]. A t-SNE projection does this by calculating the pairwise similarities between data points and finds the probability of one point being the neighbor of another[20]. Using these probabilities, t-SNE generated a two-dimensional plot based on the similarities between different data points[20]. The points that are similar will tend to attract and vice versa, creating groups of points on the graph.

Using this visualization technique we observed where our structures laid in the descriptor space (as seen by the ML-IAP) and understood how varying training data covers different parts of the space (see Figure 2.1). Previously at SIT, the training data set only included USPEX structures and we saw how little space these structures cover. By using the Sandia resources, we increased coverage with the inclusion of AIMD, further diversifying our data. With AIMD simulations we generated hundreds of data points that are unique, given that the space they cover are at different locations in comparison to the USPEX structures. By providing diverse training data to the ML-IAP, we are more likely to be interpolating between points, which is a more reliable mode of prediction. Without good coverage in descriptor space, extrapolations will be common and, thus, we do not have a reliable way to quantify the accuracy with respect to DFT.

2.2. ML-IAP Construction and Hand-Tuned Optimization. Current empirical potentials, like Lennard-Jones, can be used to simulate systems but are constrained to their respective applications, unable to predict material specific properties. We will instead use ML-IAP, which serves to link DFT and MD, capable of performing simulations with the accuracy of quantum mechanical methods[16]. To do this we will use FitSNAP, a software that automates the conversion of training structures into descriptors that are needed for the model[16]. From FitSNAP, we can construct an ACE potential [3, 6], which differs from other ML-IAP due to the descriptor set used. This descriptor set treats interactions between atoms (within a radial cutoff) as separable terms based on the body order. For example, an O_2 molecule can only exhibit two-body interactions, where H_2O will be a sum of two- and three-body interactions. This chemically intuitive mathematical description is not present in other ML-IAP methods, namely SNAP. Along with the construction of potentials, FitSNAP is also able to provide an error analysis for the user to understand the accuracy in terms of energy and force errors. This is done by producing the loss function that compares the predicted potential against DFT calculations and outputs the difference between the two[16]. Having access to these metrics allowed us to make changes to the adjustable parameters to further improve the accuracy of the potential.

Among these parameters, both group weights and hyper-parameters (i.e., lambda and radial cut off) were adjusted. There are three types of group weights that can be changed, energy, force, and stress. By increasing the value for any of the three we are telling FitSNAP to place more emphasis on that type of training and to decrease its error. These three values exist for each group where our groups are defined as seen in figure 2.1. Since the system we observed for this study is Ni-Au, we did not make any changes to the stress weight due to neither element type having significant properties that require stress to be changed for improvement of the model. This leaves us with changes made to only energy and force in terms of group weights. The accuracy of the potential is far more sensitive to change in radial cutoff and lambda than group weights, and we will later see how this creates difficulties when hand-tuning these parameters. All descriptors are atom-centered, meaning the neighboring atom positions are used to calculate the energy and force on a central atom. This process repeats itself for every atom in a given structure. Radial cutoff sets the maximum distance a neighboring atom can be from the central atom and still exert any forces and energies on

the central atom. Lambda specifies whether emphasis should be placed on further or closer neighboring atoms relative to the central atom.

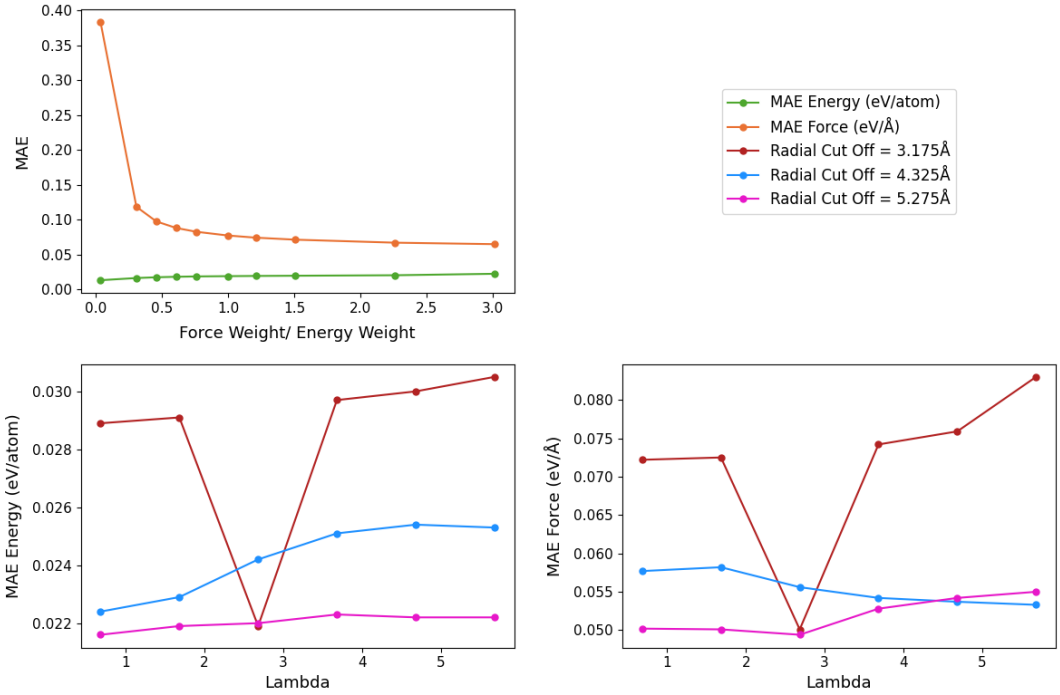


FIG. 2.2. *Graphs of changes in errors as group weights or hyper-parameters are adjusted.*

Using the generated training data, an initial potential was first created by adapting an example input script provided in the FitSNAP code repository. This script contained preset group weight values that were chosen for their specified system but can be modified for different training data sets. As for the hyper-parameter values, a separate script generates starting values based only on the element types. With these values we can begin to run FitSNAP and produce our first potentials.

To build intuition on how changes in the parameters affect the force and energy errors, an assessment of the outputted metrics was done with each FitSNAP run. This hand-tuning process was first executed against the group weights given that they are less sensitive. In figure 2.2, the graph of the ratio of the group weights versus mean absolute error (MAE) shows a total of 10 data points, each representing a FitSNAP run. We saw overtime how a significant decrease in the force MAE was made while preserving the energy MAE as we increased the force weight. Eventually, the group weights lost influence over the MAE and trade offs between the force and energy errors were noticed. Since we reached a saturation point where further hand-tuning of the group weights did not make notable improvements, we moved to the hyper-parameters.

Since the potential accuracy is far more sensitive to lambda and radial cutoff change, we set base values for radial cutoff by performing a nearest neighbor analysis on Ni-Au USPEX generated structures. From this preliminary analysis, we found that the radial cut offs of the first, second, and third nearest neighboring atoms from the central atom were located at values 3.175, 4.325, and 5.275$\mathring{A}$ respectively. For lambda, we intuitively chose to increase by integer increments. As seen in figure 2.2, there is no noticeable optimum set

of values achieved with respect to energy and force MAE, demonstrating the difficulty of hand-tuning these parameters as the errors rise and fall at different rates. Another dilemma that arose is the time needed to run more fits and analyze the metrics. This slow process only allowed 5-10 fits to be run per day and raised the need for an automation system. Thus, we turned to the genetic algorithm called the design analysis kit for optimization and terascale applications (Dakota) for further optimizations.

2.3. ML-IAP Construction and Genetic Algorithm Optimization. Dakota is a tool used to perform a variety of analyses such as uncertainty quantification, calibration, and design of experiments[1]. For the purpose of this study, we used optimization algorithms to automate the selection of hyper-parameters and group weights with the goal to evaluate more candidate potentials efficiently. The flexibility for Dakota comes from the treatment of connected codes as a black box. This refers to the ability for Dakota to propose parameters to an outside code, with the expectation that said code returns an error metric for Dakota to determine new parameter sets. Presently, the outside code is FitSNAP and it is up to our discretion on how candidate potentials are scored. Our objective functions include the lattice parameter, formation energy, vacancy energy, and substitution energy errors. We selected these because they represent basic properties of the Ni-Au alloy that should be easily reproduced by a suitable parametrization of the ML-IAP. The objective function errors were obtained by running short LAMMPS simulations that deploy the candidate ML-IAP to predict these properties of interest. The remaining steps for setting up Dakota involved selecting sampling methods of the varied parameters and adding additional objective functions for candidates evaluation.

Dakota has many sampling methods on the free variables for the optimization. Of those methods we applied two, Efficient Global Optimization (EGO) and Single Objective Algorithm (SOGA), to our runs[1]. EGO finds the best solution by iteratively using a surrogate Gaussian process model of the objective functions each time more sampling data is added[1]. It then evaluates the improvements in errors to select more sample points. For this study, sample points refer to parameter values of hyper-parameter and group weights[1]. SOGA finds the best solution by initializing a population of random sample points to be evaluated by calculating the objective values. Based on these values, SOGA then chooses the best points to be bred/mutated into a new population. For our template input file we made modifications to lambda and radial cut off, leaving them open for Dakota to know that these were the values to be optimized.

Separate to hyper-parameter tuning, there are a set of values that define the ACE basis set that are not easily optimized. Much like our quantum-chemical understanding of localized electron orbitals, the ACE descriptor set combines the radial and angular components into rotation and permutation invariant functions. Radial basis functions,$R_n(r_{ij})$, are evaluated up to $nmax$, defined for each rank of descriptor. Angular functions, $Y_l^m(\hat{\boldsymbol{r}}_{\boldsymbol{ij}})$, are terminated at $lmax$ which are independently defined for each of the $ranks$. Lastly, $mumax$ is singly defined for all $ranks$. Ongoing work in the group has selected a few basis sets for use that show broad applicability. Here, two basis sets were tested. Lastly, there is a choice in FitSNAP to additionally use an analytical interatomic potential for close atom distances that aren't well represented by DFT. This is called the Ziegler-Biersack-Littmark (ZBL) term and can be subtracted off the DFT energies and forces such that it can be consistent with fitted ML-IAP. With these confirmed inputs we began to run Dakota while varying the sampling method, basis set, and ZBL and retrieve some candidate potentials.

2.4. Potential Deployment. From a large batch of candidate potentials from Dakota, a few were isolated by using a ranking system discussed in greater detail later. With these few candidates we further tested for accuracy by utilizing MD simulations within LAMMPS.

LAMMPS is a versatile tool used by many members within the scientific community for applications in physics, chemistry, and materials science [18]. It offers great flexibility and is able to simulate systems of sizes ranging from small atomic structures to large-scale systems containing billions of particles[18]. Because of its wide range applications it is difficult to explore all its possibilities and understand what is needed to create a script that describes the simulation we wish to occur. Instead of writing these scripts from scratch, we instead used Sandia Artificial Intelligence (SAI) to accelerate the rate it takes to code LAMMPS input scripts without prior knowledge. SAI is a large language model, and is a snapshot of ChatGPT[5] that is locally hosted for uses at the lab.

Before we could implement SAI into this workflow, we first tested if SAI could create a working LAMMPS input script without providing it a pre-made script. Since our primary system for this study is Ni-Au, we instructed SAI with the given prompt: 'Give me a LAMMPS input script for Ni-Au'. SAI then outputted a series of code with commentary describing what each line does. The script provided was then run through LAMMPS and resulted in no output file to check if it gave the desired simulation. After carefully reading through the output section of the code, the issue was identified and corrected by notifying SAI of an incorrect ordering of commands. SAI acknowledged this and gave a revised script that produced an output file containing the simulation. Another problem arose when the only element type found in the structure was Ni. SAI was then asked to explain why this happened and responded with an explanation of the problematic line of code and an updated script that correctly included Au in the structure. This portion of the discussion with SAI ended with a successful simulation of a 50% Ni and 50% Au filled box using the Lennard-Jones potential.

To further challenge SAI, it was given the request: 'Can you give me a LAMMPS input script for Ni-Au alloy using a lennard-jones potential and have it be a nanoparticle'. Additional specifications were given to enhance the chances of achieving the desired simulation. This moved beyond the initial simulation by defining the atomic geometry that should be simulated, something that is usually left to domain expert users. The first script SAI produced was a success, displaying a LAMMPS simulation of a Ni-Au alloy nano-particle and exceeding expectations by including changes in temperature. The temperature was set to increase from 300 K to 1000 K and this inclusion resulted in noticeable motions of the atoms, creating visually pleasing effects for the user to watch. This proved that SAI can create working LAMMPS input scripts with the help from the user. SAI learned from it's previous mistakes, as seen by this new prompt, and required no troubleshooting. SAI could then be used to create specified simulations to qualitatively test the accuracy of the best candidate potential against different structures.

3. Results.

3.1. Best Candidate Potential. A total of eight Dakota runs were conducted, each varying either the sampling method, basis set, or if ZBL was turned on or off. Each of these runs provided a large set of candidate potentials that were difficult to sift through to find the best potential because of the number of outputted error values. Instead of manually going through the metrics of each potential, a ranking system was applied to select the best performing candidate. How this ranking system works is it will rank all the potentials for each objective functions and apply points depending on where they fall (e.g., a potential that ranks as number 54 in force error has 54 points applied to them). The higher the number of points, the worse they are in terms of quantitative errors. Once the potentials were ranked for each objective function, the points were then summed to give a total score. By using this system, we selected the best potential from each Dakota run.

TABLE 3.1
Best candidate potentials from the eight Dakota runs and their errors.

	Basis Set 1				**Basis Set 2**			
Candidates	1	2	3	4	5	6	7	8
Energy Error	0.2001	0.198	0.332	0.208	0.182	0.183	0.175	0.182
Force Error	0.131	0.189	0.295	0.200	0.230	0.229	0.228	0.256
Lattice Parameter Error	0.447	0.442	0.567	0.440	0.458	0.463	0.413	0.434
Formation Energy Error	0.338	0.418	0.299	0.339	0.106	0.162	0.169	0.200
Vacancy Energy Error	0.909	2.036	2.139	2.027	0.046	0.042	0.090	0.088
Substitution Energy Error	0.855	1.996	2.106	1.987	2.033	2.027	2.037	2.015
Overall Ranking	1	5	8	6	2	4	3	7

From Table 3.1, we inspected the objective errors between the top selected potentials and saw how they differ from each other. Some potentials outperformed others for certain objective functions. For example, potential 1 had a substitution energy error below 1 eV/atom while the remaining had an error nearing or equal to a value of 2 eV/atom. Potential 6 did well in predicting the vacancy energy, having the lowest value out of the 48 seen in the table. The same ranking system that was applied to each Dakota run is then applied to the eight potentials to enable comparison. This system found that potential 1, using basis set 1, the SOGA sampling, and ZBL turned off, was the best overall potential. After deeming potential 1 as the most accurate quantitatively, we proceeded with running the potential against different structures in LAMMPS.

3.2. Simulations of Stability Test. Potential 1 was tested against a variety of surface structures at 300 K by using a LAMMPS input script generated by SAI. The prompt that was given was: 'Can you give me a LAMMPS input script to check the stability of my potential at 300 K, I already have the POSCAR, I also want the bottom layer of atoms to be frozen while the rest are mobile. The script should also give me an output file for the entire simulation.' This resulting script simulated the given structures at the correct reaction conditions but required an additional energy minimization step to relax the system. Inclusion of this command let the structure find the configuration that is most stable by minimizing the energy. Once this configuration was found, the simulation then began to run at a constant temperature of 300 K. Additionally, the bottom layer of atoms were frozen to replicate the existence of infinite material found in solids. As for the structures this test was conducted on, it was the surfaces of each element type at the (100), (110), and (111) facets. These simulations ran to completion and did not show any incorrect physics of the system. This is proven both visually as well as quantitatively through the radial distribution functions (RDFs).

Figure 3.1 illustrated how the structures maintained their shape during the stability tests, as seen by structure images and RDFs. These images, captured at the final frames of the simulation, did not display any black holes or anomalies indicative of an inaccurate potential that needs further optimization. The RDFs provided insights on how the atoms are shifting as time passes. Although it is clear that the atoms do shift from their original positions, the overall structure remained consistent. This is noticed when the biggest difference between the RDFs before and after simulation is the rounding of the peaks, indicating

that the locations of the atoms changed but still kept the crystal-like state. This serves as evidence that the currently developed potential is able to accurately predict surface physics of Ni and Au.

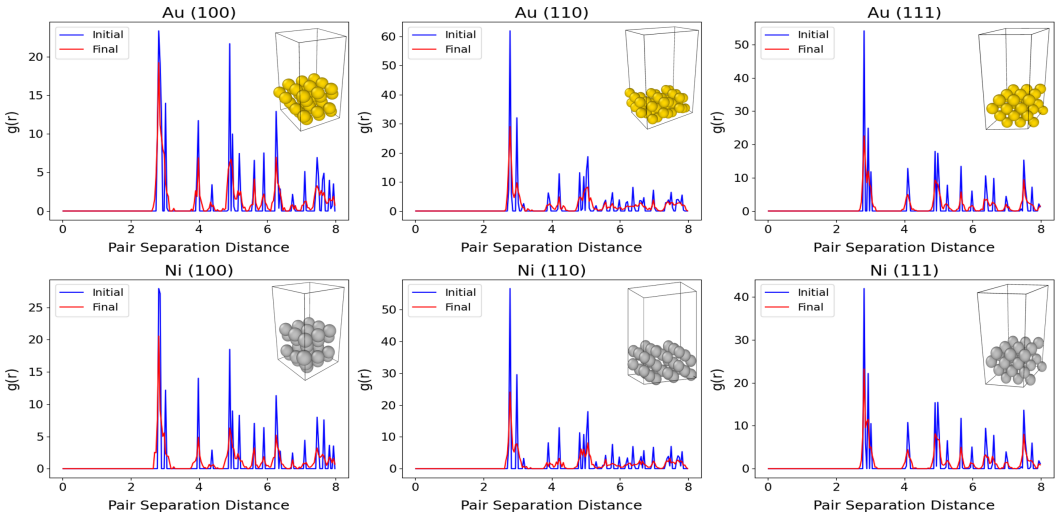


FIG. 3.1. *RDFs for Ni and Au surfaces before (blue) and after (red) the MD simulations at 300 K performed to test the stability of the optimum nonmagnetic Ni-Au ML-IAP.*

4. Conclusions. In this study, we developed a nonmagnetic Ni-Au ML-IAP that successfully predicted the stable surface behavior of pure Ni and Au despite the absence of surface structures in the training data set. This demonstrated the remarkable capability of machine learning techniques to construct accurate interatomic potentials, even where some extrapolation from the training set is present. Throughout this process, we were able to add thousands of data points to our training data and provide structures the ML-IAP had never seen before, covering a larger amount of the descriptor space. This then lead to improvements in the accuracy of the ML-IAP, but was later hindered by the slow rate of the hand-tuning parameter optimization process performed to decrease errors. Manually shifting the values of the adjustable parameters raised the need for a tool that could automate this process. A genetic algorithm was then employed to gather larger sets of sample parameter values and produce more candidate potentials.

Future work for this project should involve more simulation tests of the best candidate potential against different structures to determine what predictions it is unable to make. From here, we could then decide what type of structures should be added to the training data set as this aspect of the project is always open for further enhancement. Continuous refinement of this potential will then lead to a comparative analysis between fully optimized nonmagnetic and magnetic potentials. This would enable us to asses how influential magnetism is on ML-IAP accuracy and determine if the magnetic property should included or excluded in the potential development process.

REFERENCES

[1] B. M. ADAMS, W. J. BOHNHOFF, K. R. DALBEY, M. S. EBEIDA, J. P. EDDY, M. S. ELDRED, R. W. HOOPER, P. D. HOUGH, K. T. HU, J. D. JAKEMAN, ET AL., *Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification,*

and sensitivity analysis: version 6.13 user's manual, tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.

[2] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, *Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons*, Physical review letters, 104 (2010), p. 136403.

[3] R. Drautz, *Atomic cluster expansion for accurate and transferable interatomic potentials*, Physical Review B, 99 (2019), p. 014104.

[4] T. B. Ferriday and P. H. Middleton, *Alkaline fuel cell technology-a review*, International journal of hydrogen energy, 46 (2021), pp. 18489–18510.

[5] L. Floridi and M. Chiriatti, *Gpt-3: Its nature, scope, limits, and consequences*, Minds and Machines, 30 (2020), pp. 681–694.

[6] J. M. Goff, C. Sievers, M. A. Wood, and A. P. Thompson, *Permutation-adapted complete and independent basis for atomic cluster expansion descriptors*, Journal of Computational Physics, 510 (2024), p. 113073.

[7] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, et al., *Commentary: The materials project: A materials genome approach to accelerating materials innovation*, APL materials, 1 (2013).

[8] J. Liu, B. Zhang, Y. Fo, W. Yu, J. Gao, X. Cui, X. Zhou, and L. Jiang, *Electrochemically induced dilute gold-in-nickel nanoalloy as a highly robust electrocatalyst for alkaline hydrogen oxidation reaction*, Chemical Engineering Journal, 464 (2023), p. 142692.

[9] W. Liu, Y. Zhu, Y. Wu, C. Chen, Y. Hong, Y. Yue, J. Zhang, and B. Hou, *Molecular dynamics and machine learning in catalysts*, Catalysts, 11 (2021), p. 1129.

[10] A. O. Lyakhov, A. R. Oganov, H. T. Stokes, and Q. Zhu, *New developments in evolutionary structure prediction algorithm uspex*, Computer Physics Communications, 184 (2013), pp. 1172–1182.

[11] N. C. Nguyen and A. Rohskopf, *Proper orthogonal descriptors for efficient and accurate interatomic potentials*, Journal of Computational Physics, 480 (2023), p. 112030.

[12] A. R. Oganov and C. W. Glass, *Crystal structure prediction using ab initio evolutionary techniques: Principles and applications*, The Journal of chemical physics, 124 (2006).

[13] A. R. Oganov, A. O. Lyakhov, and M. Valle, *How evolutionary crystal structure prediction works and why*, Accounts of chemical research, 44 (2011), pp. 227–237.

[14] A. G. Oshchepkov, G. Braesch, A. Bonnefont, E. R. Savinova, and M. Chatenet, *Recent advances in the understanding of nickel-based catalysts for the oxidation of hydrogen-containing fuels in alkaline media*, ACS Catalysis, 10 (2020), pp. 7043–7068.

[15] S. Plimpton, *Fast parallel algorithms for short-range molecular dynamics*, Journal of computational physics, 117 (1995), pp. 1–19.

[16] A. Rohskopf, C. Sievers, N. Lubbers, M. Cusentino, J. Goff, J. Janssen, M. McCarthy, D. M. O. de Zapiain, S. Nikolov, K. Sargsyan, et al., *Fitsnap: Atomistic machine learning with lammps*, Journal of Open Source Software, 8 (2023), p. 5118.

[17] J. Stöhr and H. C. Siegmann, *Magnetism*, Solid-State Sciences. Springer, Berlin, Heidelberg, 5 (2006), p. 236.

[18] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, *LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales*, Comp. Phys. Comm., 271 (2022), p. 108171.

[19] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker, *Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials*, Journal of Computational Physics, 285 (2015), pp. 316–330.

[20] L. Van der Maaten and G. Hinton, *Visualizing data using t-sne.*, Journal of machine learning research, 9 (2008).

[21] Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csányi, A. V. Shapeev, A. P. Thompson, M. A. Wood, et al., *Performance and cost assessment of machine learning interatomic potentials*, The Journal of Physical Chemistry A, 124 (2020), pp. 731–745.

DECISION TREE MACHINE LEARNING MODEL CONSTRUCTION FOR PARTICLE SIMULATION

QIMORA M. MASON* AND KATHRYN A. MAUPIN†

Abstract. This research demonstrates the use of machine learning techniques into particle clustering methods for enhancing uncertainty quantification (UQ) and validation analysis. Our study explores the development and application of a machine learning-based particle clustering code designed to improve the accuracy processes. By using advanced machine learning algorithms, including supervised learning techniques like decision tree, support vector machines, and neural networks, more precise identification and characterization of uncertainty in simulation models can be achieved. This paper suggests that combining machine learning with particle clustering holds significant potential for advancing UQ methodologies and validation analysis in many engineering and scientific implementations.

1. Introduction. High-fidelity computational codes are implementations of physics-based, first-principles models, the cost of which may vary but are typically computationally expensive. Reduced-cost simulation techniques, such as surrogate modeling, have been extensively used to study and analyze complex real-world systems. Surrogate-based models ease the computational cost of exploring parameter and design space for engineering tasks, such as model calibration, uncertainty quantification (UQ) and sensitivity analysis (SA). Model calibration is the process of tuning model parameters to enhance the capability of the model to reproduce observed data. In applications with a large number of parameters, sensitivity analysis may be used to reduce the size of the parameter space, thereby easing the computational burden of parameter studies, e.g. calibration and UQ. Uncertainty quantification is the science of quantifying estimations of the uncertainties associated with constructing a computational model and using it in real-world applications.

Machine learning (ML) is a branch of artificial intelligence (AI) which enables us to tackle complex tasks with fixed programs and algorithms developed by human beings. The concept of learning in ML refers to the ability to perform specific tasks through the machine. It focuses on using the provided data and selected algorithms to process the way that humans learn, while improving predictions and accuracy. An example of ML is a collection of features (inputs) that have been quantitatively measured from a given dataset or process that we want the ML system to follow. A computational program learns from experience with respect to a specified class of tasks, and the performance of the machine learning model is then measured.

Machine learning models empower computational predictions in a broad variety of applications, including engineering [15], health care [4, 12], and sciences [11, 16]. In particular, ML has been used to enhance UQ methodologies through the use of Gaussian process regression and neural networks [20]. To examine the soundness of ML implementations, formal metrics may be used to assess the overall quality of predictive accuracy in regression and classification problems. Validation analyses in machine learning and surrogate modeling can help evaluate the overall performance of the model to produce sufficient certainty of its predictions. In the evolving realm of machine learning, generating the accuracy for a predictive model is a tedious but necessary process for informed decision making. By integrating UQ and validation into the growth and learning process, an ML model will provide a better understanding of the high-fidelity model's robustness, limitations, and reliability.

This paper focuses on the construction of a decision tree ML model for a particle clustering simulation. An overview of the background mathematics of the regression algorithm

*Elizabeth City State University, qmmason342@students.ecsu.edu
†Sandia National Laboratories, kmaupin@sandia.gov

can be found in Section 2. Section 3 details an exemplar application and the process used to determine the validity of the constructed model. Concluding remarks can be found in Section 4.

2. Background. To effectively create our model, we use a decision tree model, as it breaks our data down into more attainable parts. Decision trees are popular as ML models due to their ability to handle complex and mixed data and their reliability and simplicity [17]. In regression tasks, a statistical model between the inputs and outputs is estimated according to some data. Then, the model is asked to predict a numerical value given some input.

Machine learning algorithms have many hyper-parameters, which act as settings that we can use to control the algorithm's behavior. The values of these hyper-parameters are not adapted by the learning decision tree algorithm itself [20]. Implementing our decision tree regressor, we were able to observe these inputs of our data and train our model accordingly in the structure of a tree.

We use the scikit-learn library in python [13] to construct our supervised decision tree model. The decision tree regressor is a supervised learning algorithm. This allows our model to learn over time, as our dataset includes the correct independent and dependent variables.

2.1. Regression Tree Algorithm. In many real-world domains, the task of machine learning algorithms is to learn models that predict numerical values [10]. A regression tree is a type of decision tree used for predicting continuous outcomes [9]. It divides the data into subsets using a series of splits based on feature values with the ultimate goal of predicting a numerical target variable. Each terminal nodes, branches, and leaf represents a predicted outcome.

At each node, the dataset is split into two or more groups based on a feature X_i that minimizes the variance of the target variable within each group. A common approach for selecting the best split is minimizing the absolute error. The goal is to minimize the sum of absolute differences between the predicted and actual values.

In a regression decision tree, the primary mathematical concept used is the calculation of standard deviation reduction (SDR). The SDR is based on the decrease in standard deviation after a dataset is split on an attribute [19]. The variance σ^2, which measures the average distance of all data points x_i from the mean μ is mathematically calculated as,

$$\sigma^2 = \sum_{i=1}^{n} (x_i - \mu)^2 / n, \tag{2.1}$$

where n is the total number of data points. The standard deviation σ is the square root of the variance,

$$\sigma = \sqrt{\sum_{i=1}^{n} (x_i - \mu)^2 / n}, \tag{2.2}$$

The goal of a regression tree is to minimize the total variance within the resulting subgroups after each split [2]. The formula for the total variance after splitting a node can be expressed as,

$$\sigma^2_{total} = \frac{n_L}{n}\sigma^2_L + \frac{n_R}{n}\sigma^2_R, \tag{2.3}$$

where, n_L and n_R are the number of samples in the left and right child nodes and σ^2_L and σ^2_R represent the variance in the left and right child nodes [6]. The reduction in the standard

deviation or SDR is then computed as the difference between the standard deviation of the parent node and the standard deviation of its children,

$$SDR = \sigma_{total} - (\frac{n_L}{n}\sigma_L + \frac{n_R}{n}\sigma_R) \tag{2.4}$$

The split that minimizes this SDR is chosen as the best split at each node.

By recursively splitting the dataset based on features that minimize the variance in the target variable, regression trees provide a clear and understandable model structure. The main advantages of regression trees include their ability to handle both numerical and categorical data, their ease of interpretation, and their capability to capture complex, non-linear relationships in data [2].

2.2. Limitations of Regression Models. Decision tree regression models have notable limitations, such as overfitting, where trees become overly complex and fit to noise in the data rather than the underlying trends. To avoid overfitting, a grid search with cross validation was used to tune our hyper-parameters [13]. This approach combines an exhaustive grid search of the parameter space with a cross validation calculation to identify the best combination of hyper-parameters for a given training data set. For example, if the max depth parameter is set too high, the model will learn fine details of the training data and eventually learns from the noise. This means that the model would overfit. Thus, a grid search cross validation helps prevent this problem and reduces the risks of overfitting by making sure that the model remains consistent and reliable across various splits.

Decision tree models are also sensitive to small changes in the data, which can result in different tree structures [14]. Compared to linear regression, which assumes a linear relationship between variables, decision trees are more flexible since they can model non-linear patterns. Random forests and gradient boosting algorithms also offers better performance by averaging or boosting trees, reducing overfitting and improving extrapolation [14].

Moreover, decision trees may struggle with outliers, unlike ridge or lasso regression, which use regularization techniques to reduce the impact of multicollinearity and prevent overfitting. These regularized models provide simpler, more interpretable solutions for high-dimensional data [18]. Regression tree models have been used in a variety of applications, ranging from quantum mechanics [3], fraud detection [5], quality control [1] and many more.

2.3. Validation Metrics. To quantify the efficiency, accuracy, error, and overall quality of an ML model, evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are calculated [20]. The MAE is the average absolute difference between our predicted values and our actual values,

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{2.5}$$

Here, n is the total number of observed data points in the testing set, $\hat{y}$ are the predicted values, and y are the true values within our dataset. We use this to assess the effectiveness of our regression model. MAE is more sensitive to outliers and treats all errors equally, which penalizes large deviations.

The root mean square error (RMSE) is an error metric similar to the MAE. Intuitively, RMSE is the square root of the mean of the squared error. Calculating the RMSE is a crucial metric for evaluating our model's performance [7]. Mathematically, it can be written as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{2.6}$$

where, as before, n is our total number of data points, y_i are the actual observations within our dataset, and $\hat{y}_i$ are the ML model predictions.

3. Example. As an exemplar application, we consider a particle-based current model. This model takes as inputs the voltages of three areas of the device being simulated. The emitter, base, and collector voltages are denoted by V_e, V_b, and V_c, respectively. As quantities of interest, we take the corresponding current output. The emitter, base, and collector currents are similarly denoted by I_e, I_b, and I_c, respectively.

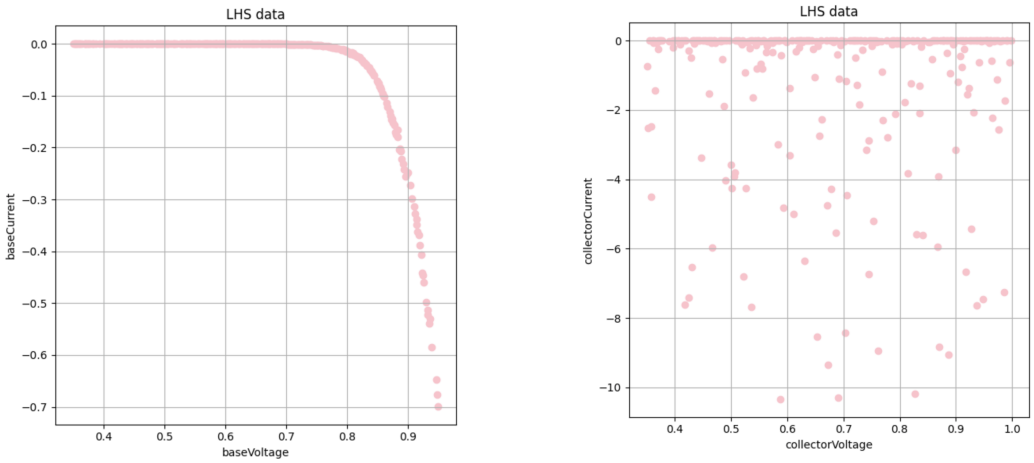
The parameter space is sampled using Latin-hypercube sampling (LHS). LHS uses a stratified sampling method to improve coverage of the space being sampled. When sampling a d-dimensional space of random variables $\{x_i\}_{i=1,2,\ldots,d}$, stratification in LHS is attained by dividing each dimension into n non-overlapping intervals of equal probability, where n is the total number of computational runs produced. One value is randomly selected from each of the extreme intervals [8]. This method is used in computational modeling of reliability analyses due to the fact that the samples fill the space better than a purely random sampling algorithm.

For the application considered here, 500 LHS samples were drawn from the parameter space. The emitter voltage is set to zero, and the base voltage and collector voltage are both assigned a uniform distribution,

$$V_e = 0, \quad V_b \sim \mathcal{U}[0.35, 1], \quad V_c \sim \mathcal{U}[0.35, 1]. \tag{3.1}$$

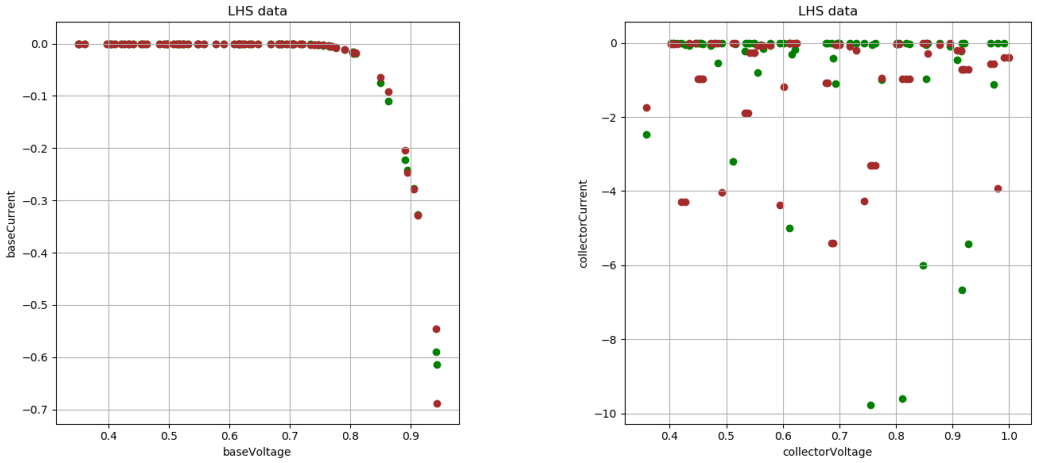
The high-fidelity computational code was then run at each of the LHS sample points, and the current values I_e, I_b, and I_c were extracted.

Given a set of data from a high-fidelity physics code, it is a generally accepted practice to use 80 percent of the data for training an ML model and reserve the remaining 20 percent for testing its performance. Here, we randomize and shuffle our data before splitting it by setting our random state equal to 42. For simplicity, we construct two separate models. The first takes as input the base voltage and predicts the base current, while the other takes as input the collector voltage and predicts the collector current. Figure 3.1 showcases the



(a) (b)

FIG. 3.1. *Scatterplots of the training data generated by randomly selecting 80 percent of the high-fidelity data produced through an LHS study of the parameter space. On the left, (a) shows the base current compared to the input base voltage values. Similarly, on the right, (b) shows the collector current compared to the input collector voltage values.*



(a) (b)

FIG. 3.2. *Scatterplots of the decision tree predictions compared to the true testing data. These data points represent the remaining 20 percent of the high-fidelity data produced through an LHS study of the parameter space. The green dots represent the true data, and the brown dots represent the decision tree model prediction. On the left, (a) shows the base current compared to the input base voltage values. Similarly, on the right, (b) shows the collector current compared to the input collector voltage values. Note: These figures will be updated. Some data was lost during Qimora's separation from SNL and figures will have to be recreated.*

visualization of our training data. The two cases shown in Figure 3.1 are quite different. The data for the base voltage and base current displays a strong dependency, while that for the collector voltage and collector current appears to be more random. This training data is used to train our decision tree regressor, and the testing data is used to evaluate the accuracy of the trained model.

To evaluate the performance of the decision tree regressor model, we conducted a series of validation tests. Using 5-fold cross-validation, the base voltage yielded an RMSE of 0.01 and an MAE of 0.02. With these scores, we can say that we would be able to trust our decision tree regressor model to emulate the base voltage of this system. Again using 5-fold cross-validation, the collector voltage yielded an RMSE of 3.28 and an MAE score of 1.58. These results are too high to consider the model a trustworthy predictor of the collector voltage. This is likely due to the fact that the correlation between the base voltage and the base current was much stronger than that between the collector voltage and the collector current. The collector voltage and current data is much more random. The decision tree regressor was unable to capture the relationship between the independent and dependent variables because the relationship was too weak.

Scatterplots of the predictions from the decision tree model compared to the true testing data are shown in Figure 3.2. Here, the green dots represent the true data, and the brown dots represent the decision tree model prediction. We can qualitatively see what the metrics tell us quantitaively. The decision tree model performs quite well in predicting the base current, where the relationship between the input and output are clear. However, the model is unable to predict the collector current, where the relationship between the input and output appears to be more random.

4. Conclusion. In previous studies, decision trees have been used to model and predict outcomes by leveraging the model's ability to handle very complex, non-linear relationships

within data. In this study, decision trees were successfully used and validated as an emulator for the base current as a function of base voltage in an advanced circuit model. However, in the scenario where the relationship appears to be more random, the decision tree model performed poorly. Through validation techniques, such as grid search cross validation and the use of validation metrics, the robustness and reliability of the predictions of the ML model have been ensured.

REFERENCES

[1] A. O. Alnahit, A. K. Mishra, and A. A. Khan, *Stream water quality prediction using boosted regression tree and random forest models*, Stochastic Environmental Research and Risk Assessment, 36 (2022), pp. 2661–2680.

[2] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, Wadsworth International Group, 1984.

[3] E. Farhi and S. Gutmann, *Quantum computation and decision trees*, Physical Review A, 58 (1998), p. 915.

[4] H. Habehh and S. Gohel, *Machine learning in healthcare*, Curr. Genomics, 22 (2021), pp. 291–300.

[5] M. Hammed and J. Soyemi, *An implementation of decision tree algorithm augmented with regression analysis for fraud detection in credit card*, International Journal of Computer Science and Information Security (IJCSIS), 18 (2020), pp. 79–88.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2nd ed., 2009.

[7] T. O. Hodson, *Root mean square error (rmse) or mean absolute error (mae): When to use them or not*, Geoscientific Model Development Discussions, 2022 (2022), pp. 1–10.

[8] R. L. Iman, *Latin hypercube sampling*, in John Wiley & Sons, Ltd, 2008.

[9] G. James, D. Witten, T. Hastie, and Tibshirani, *An Introduction to Statistical Learning with Applications in R*.

[10] S. Kramer, *Structural regression trees*, in AAAI/IAAI, Vol. 1, 1996, pp. 812–819.

[11] P. Langley et al., *The changing science of machine learning*, Machine learning, 82 (2011), pp. 275–279.

[12] V. Nemani, L. Biggio, X. Huan, Z. Hu, O. Fink, A. Tran, Y. Wang, X. Zhang, and C. Hu, *Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial*, Mechanical Systems and Signal Processing, 205 (2023), p. 110796.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[14] J. R. Quinlan, *Induction of decision trees*, in Machine learning, 1986, pp. 81–106.

[15] Y. Reich and S. Barai, *Evaluating machine learning models for engineering problems*, Artificial Intelligence in Engineering, 13 (1999), pp. 257–272.

[16] F. A. R. Rodríguez, L. G. Flores, and A. A. Vitón-Castillo, *Artificial intelligence and machine learning: present and future applications in health sciences*, in Seminars in Medical Writing and Education, vol. 1, 2022, pp. 9–9.

[17] J. Su and H. Zhang, *A fast decision tree learning algorithm*, in Aaai, vol. 6, 2006, pp. 500–505.

[18] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society: Series B (Methodological), 58 (1996), pp. 267–288.

[19] A. Tripathi, *Decision tree for regression models in machine learning*, 2021.

[20] R. K. Tripathy and I. Bilionis, *Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification*, Journal of computational physics, 375 (2018), pp. 565–588.

BREAKING BAD STRUCTURE GENERATION: METHODS FOR SYSTEMATIC, DATA-DRIVEN ATOMISTIC STRUCTURES FOR ML MODEL TRAINING

COREEN MULLEN*, EMBER SALAS 2†, AND JAMES GOFF 3‡

Abstract. In this paper we discuss the development of the Generalized Representative Structure (GRS) software, which aims to represent large atomistic environments using smaller representative structures called "fingerprints". Current methods for representing large atomistic environments are restricted to simple, fixed-lattice, crystal structures. This is an issue when it comes to trying to represent more complex systems that can be found in the natural world. They also focus on predicting material properties based on structures, using inverse machine learning we are able to generate structures given specified properties. Our method uses the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) molecular dynamics program and Atomic Cluster Expansion (ACE) descriptors to create structures without these restrictions. The findings of our research show that the GRS method can accurately predict some of the properties of its target environment. The results of our method will help to diversify the data used for Density Functional Theory (DFT) simulations and reduce the computational resources needed for DFT.

1. Introduction. Computational materials research leverages many algorithms and software tools to predict properties of materials. Some methods have been established for doing the inverse of this, yielding atomic structures with specific properties[24] . This is of particular interest for bridging the many length scales accessed with different atomistic simulation methods and for curating atomistic materials databases. Materials databases and many computational research efforts, such as machine-learned interatomic potential (MLIP) training, often rely on highly-accurate but computationally expensive Density functional theory (DFT) calculations. The DFT method falls under the electronic structure length and time scale, shown in Fg. 1.1 [13]. DFT calculations offer fairly accurate predictions of thermodynamic and electronic properties, and more recently have served as training data in a wide variety of data-driven research efforts. These include materials informatics, database development, machine-learned model development, and more[22]. Given the computational expense for DFT and electronic structure theory methods, it is important to ensure that they are performed only for the necessary atomic structures for the particular research.

Molecular Dynamic (MD) is a very important computational simulation method that aids in research, modeling the movement of atoms and the interactions of a molecular system at the microscopic level. The MD technique solves Newton's equation of motion to observe this behavior by calculating the particles position in reference to the forces being enacted on the particle[21]. The Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is a software package that is well known for use in MD simulations. Its popularity can be attributed to its ability to curate a wide range of models, both simple and complex, for a multitude of materials. MLIPs can be integrated with LAMMPS providing the ability to predict different variables of a system such as the force and energy based on the input of molecular dynamics simulations.

Procuring atomic structure data and developing atomistic structure databases has become a mainstay in computational research for the fields of materials science, physics, and chemistry. While databases containing comprehensive high-fidelity information have been obtained for many materials systems, they often exclude more exotic or complex phases such as liquid phases or phases that lack a lot of crystal symmetry[9, 7, 17]. The atomic structures in these databases are useful for some applications, however, they do not always reflect the properties of the material in real life systems. For example, they often are perfect crystals without defects, while real life structures are never perfect, and often have defects. These databases are also often centered around well-ordered, low-energy structures, leading to limited or even redundant information in them. This presents a problem for data-driven research that requires comprehensive, diverse data.

*Elizabeth City State University, cmmullen879@students.ecsu.edu
†Sandia National Labs, elsikor@sandia.gov
‡Sandia National Labs, jmgoff@sandia.gov

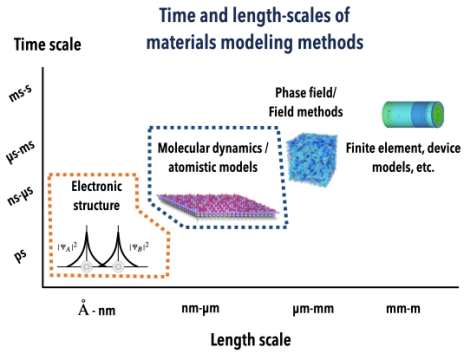


FIGURE 1.1. *Typical time and length scales accessible with different materials modeling methods.*

Though some computational resources allow for 'brute-force' research with large amounts of DFT, vast computational resources aren't available to all. It would be more useful for the broader scientific community if the expensive calculations could be tailored to different computational budgets. Data redundancy is another issue that can come from 'brute-force' research, having redundant data in a dataset increases the amount of time and computational resources needed for the process[10][23].

1.1. Background. For the purposes of discussing challenges with atomic structure generation in practice, the development of MLIPs is highlighted. However, these challenges are shared with many other data-driven materials models. While MLIPs are a compelling method to extend the accuracy of expensive ab-initio MD simulations to more relevant time and length scales, MLIPs often require an expert curated database of atomistic structures to train on. MLIPs can handle large systems, adapt with new data input, and are significantly faster than ab-initio methods. However, curating training data for accurate and efficient MLIPs is one of the key challenges of these methods.

To train MLIPs, many DFT calculations are often performed on atomic structures. From the scale of DFT, sizes are limited to ~ 100 atom systems, classical or ML-driven MD simulations are limited to 10^{12} atoms on the upper end, and in real life systems, things are on the scale of 10^{23}. This disparity is a key focus of this work. For MLIPs to accurately predict properties of realistic or even MD-sized systems, the potential needs to be able to extrapolate and interpolate. Recently developed advanced ML models are expected to be good at this, but require adequate training data. Such models often require that some of the characteristics of the more realistic systems be reflected in the training database. This is a challenge in and of itself, because the training database typically needs to be comprised of 1-100 atom structures to be compatible with DFT.

Some elegant solutions for this type of problem have been presented for simple systems like alloys, such as the Special Quasi-random Structure (SQS) method. Small atomic structures may be generated such that they have the same correlation functions of a true bulk alloy[25]. The atomic structures that possess the same chemical ordering characteristics as the bulk alloy referenced are referred to as special quasi-random structures. While this is useful for atomistic modeling of alloys, it doesn't work for all possible phases of the material. For example, this method isn't suitable for representing the melting process of an alloy. This is because the SQS method relies on finding structures that best reproduce the fixed-lattice cluster correlations of a target system. The target system in SQS is typically a completely random (chemically uncorrelated) alloy. In this context, SQS may be characterized by structures that have fingerprints (fixed-lattice cluster correlations) that closely resemble the fingerprints of the target. These fingerprints in SQS only encode information about the chemical occupation of lattice sites. The goal of this research is to develop and validate a method that is suitable for representing more complex materials systems and physical/chemical processes

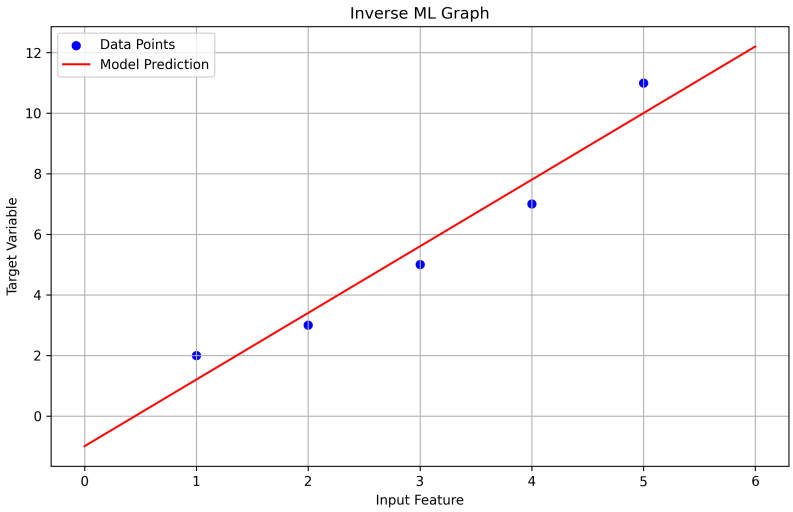


FIGURE 1.2. *Inverse machine learning graph that shows relationship between input and target variable. The blue points represent the data, the red line represents the predicted values from the linear regression model.*

than just alloys on a fixed-lattice. This method, referred to within as the Generalized Representative Structure (GRS) method, may be used for generating atomic structures that represent (in theory) arbitrary atomic configurations and chemistries. The method is developed such that GRS may accurately represent alloy phases, but also liquids, amorphous systems, ordered phases, and more. This is accomplished with GRS by generating structures that either collectively or independently reproduce a target fingerprint distribution. These target fingerprints may correspond to a large atomic structure, a set of atomic structures, an alloy, a liquid, etc. This can be accomplished if fingerprints are sensitive to positions as well as chemistries. Thus, the GRS method relies on the Atomic Cluster Expansion (ACE) descriptors to do this. These are a natural extension of the fixed lattice cluster correlations used in traditional SQS[1]. It is also advantageous to use these given the direct connection to MLIPs[1, 4]. A key goal of this work is to develop and validate this GRS method to reduce the amount of DFT used in training ML models, and to provide DFT-sized representations of complex materials.

It is proposed that GRS may be generated analogously to how SQS structures are generated. In this work, GRS are made to reproduce complicated target distributions beyond alloys on a perfect crystal lattice.The ability to do so and the accuracy with which GRS may be generated is assessed. The ability to generate smaller atomic structures and reduce atomistic ML datasets is also assessed to see if GRS can reduce the amount of DFT needed for computational materials research or speed the DFT calculations by reducing their size (while retaining similar amounts of information).

While other structure generation methods have been developed, these methods often rely on an external energy model[11]. Data-driven structure generation methods offer compelling alternatives, such as the entropy maximization method[8]. This method aims to globally sample descriptor space, which is an effective, systematic way for getting a comprehensive span of descriptor space. However, globally sampling descriptor space can be very resource-intensive, sometimes resulting in hundreds of thousands of structures. The massive computational cost this could incur for DFT calculations is not compatible with all computational budgets, and may still cause concerns for redundancy.

The GRS method is data-driven and does not require an a-priori energy model like the entropy

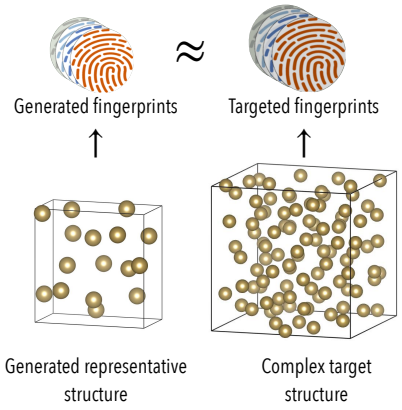


FIGURE 1.3. *A generalized representative structure (GRS) of a liquid Tantalum (Ta) system. A small structure may be produced that matches (as closely as possible) the features of a larger system.*

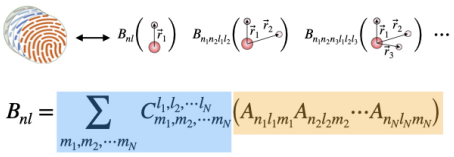


FIGURE 2.1. *Illustration of the N-body ACE descriptors and defining equation. The key features to highlight about the ACE descriptors are the rotational invariance, ensured by the blue highlight of the Clebsch-Gordan product, and the invariance of ACE descriptors with respect to permutations of bonds (accomplished by permutation-invariant functions in the orange highlight).*

maximization method. Rather than aiming to span a window of descriptor space, it provides (1) a way to obtain structures resembling a target or (2) a way to systematically find structures that deviate from a target distribution. Because the GRS method does allows one to tailor how far from a target to search or how accurately to represent a target, it is possible to systematically search over descriptor space while balancing computational cost. The ability to do this with GRS will be tested in some applied cases.

Using our GRS method is also beneficial from an ethical research standpoint [6]. Considering that when using a scientists intuition for what training data structures will be used, there is likely to be some bias in their selection. For example, it is common to fit MLIPs on bulk ground state data, and if it were to make up the majority of the training dataset, there will likely be poor extrapolation to other phases and conditions[16]. It is shown in this paper that the GRS method may be used to systematically add training data to MLIP datasets.

2. Methodology. To highlight the key methodology for GRS, some concepts for the SQS method are important to recall. The SQS method relies on minimizing a loss function that matches the fingerprints of a random alloy in a smaller crystal unit cell.

The fingerprints used in SQS only encode information about atomic structures on a fixed lattice. The GRS method relies on the use of Atomic Cluster Expansion (ACE) descriptors to encode atomic information on or off-lattice. Detailed definitions and derivations of ACE descriptors may be found elsewhere [5, 3, 2]. For the purposes of this work, it is sufficient to say that the ACE descriptors encode the information needed to generalize the SQS method off-lattice. The ACE descriptors may be constructed such that they depend on both the collection of atomic positions in a structure, $\boldsymbol{R} = (\vec{r_1}, \vec{r_2}, \cdots \vec{r}_{N_{at}})$, and chemical occupations of atomic sites, $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots \mu_{N_{at}})$. These

ACE fingerprints can be generated for arbitrary body order by taking higher order products of atomic basis functions, as illustrated in Fig. 2.1.

With the ACE descriptors presented, the loss function central to the GRS method may be defined.

$$\begin{aligned} Q_{GRS} &= \alpha_1 M_1^{GRS} + \alpha_2 M_2^{GRS} + \cdots \\ M_1^{GRS} &= \sum_{\boldsymbol{\nu}} \alpha_1 |\bar{B}_{\boldsymbol{\nu}} - \bar{B}_{\boldsymbol{\nu}}^{target}|^2 \\ M_2^{GRS} &= \sum_{\boldsymbol{\nu}} \alpha_2 |\mathrm{var}(B_{\boldsymbol{\nu}}) - \mathrm{var}^{target}(B_{\boldsymbol{\nu}})|^2 \end{aligned} \tag{2.1}$$

The loss function for GRS, Eq. (2.1) is defined by the difference between the average ACE fingerprint values of the GRS and those of the target. This is the M_1^{GRS} term. The difference between higher order moments in the distribution, such as the variance with the M_2^{GRS} term, may be added for minimal additional computational cost. The GRS is then defined by the set of atomic coordinates and chemistry's that minimize this loss function.

$$\min_{(\boldsymbol{R},\boldsymbol{\mu})} \left[Q_{GRS} \right] \tag{2.2}$$

The evaluation of the input structures is done by scoring the M_1^{GRS} term. It is used as our metric to assess the quality of a structure. We use it because if the difference between the fingerprints for the GRS and the target structure is minimal, then that means the GRS is accurately representing the target.

$$MAE\_1_{metric}^{GRS} = \frac{1}{n_{desc}} \sum_{\boldsymbol{\nu}} |\bar{B}_{\boldsymbol{\nu}} - \bar{B}_{\boldsymbol{\nu}}^{target}| \tag{2.3}$$

When the DFT process is being run on large sets of data, there can be an uptick in the time it takes to complete. These multi-structure solutions are used in the GRS method as a way to accurately depict more complex systems. Structures and targets can both be made of smaller structures, and still benefit from the GRS method being used.

2.1. Computational Tools & Software. LAMMPS plays an important role in the minimization of the distance between our generated structures and our target structure. We use the Conjugate Gradient minimization method from LAMMPS which combines the current and previous force gradient to start searching in a new direction over the space[19]. Our method is being run with the Python version of LAMMPS (pyLAMMPS), as it is convenient to our Python script to include the work needed by LAMMPS without needing to run it separately in the LAMMPS software.

The Atomic Cluster Expansion (ACE) provides the fingerprints for our representative structures.

$$V^{\text{eff}}(R, \mu) = V_{\text{core}}(R, \mu) + Q_{\text{GRS}}(R, \mu)$$

2.2. Data Generation. Training data is used in various fields of machine learning research and is arguably the most important piece of the process. The training data that we are observing in this research includes elements that are transition metals or metalloids. During our testing of the software, we worked with four different data sets:

1. A single-element dataset of tantalum (Ta) from the FitSNAP paper [18], serving as a simple and small starting example.
2. A dataset of nickel (Ni) and gold (Au) provided by researchers to test a multi-element system with our software.
3. A dataset of magnesium (Mg) and calcium (Ca).
4. A dataset of iron (Fe), chromium (Cr), silicon (Si), and vanadium (V).

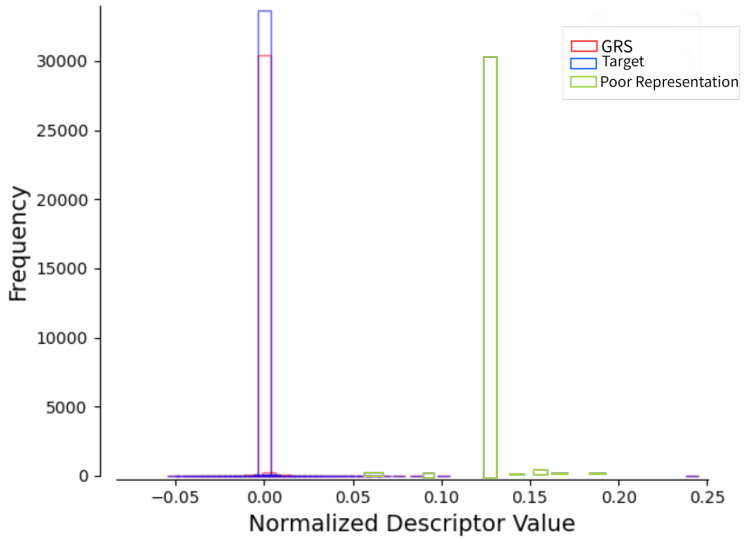


FIGURE 2.2. *A table depicting a 16 atom NiAu as the target and a good and poor-representation of GRS comparatively.*

2.3. Procedures of Software. Candidate structures are different unique configurations of atoms that we use as a foundation for our GRS method to generate its structures. For different atomic structures, with different atomic elements, there would need to be a variety of candidate structures to represent the vast array of unique configurations a system can be formed as. It is intended that the GRS method can work for arbitrary structures as well as perfect candidate structures. If all of the candidate structures given are perfect structures, or vice-versa, where the candidate structures would all be unique, then the GRS method would not be able to efficiently represent the other side.

There are various ways to create candidate structures, which is beneficial considering the various types of structures we need to represent. One way is using crystal structures with variable density, and with a chemical composition that can either remain constant or be fixed. Including supercells of specified parent lattices, such as body-centered cubic (BCC), face-centered cubic (FCC), and hexagonal-close-packed (HPC) lattices with random chemical compositions. Symmetrically distinct supercells are also used for these crystal structures. Another type of candidate structure needed for accurate GRS are amorphous/quasi-crystalline candidates with variable density and variable or fixed chemistry. These structures are constructed from supercells with a specific number of atoms and without a parent lattice structure allowing for randomly initialized positions. These structures are useful when simulating disordered systems.

Algorithm 1 Pseudocode for evaluating MAE^{GRS}

```
Define run_lammps
for each atom in atoms do
   set elements
   Set pair style and coefficients
   Compute descriptors
   Run Simulation
   Call run_lammps function
   Return descriptor gradients
end for
Set target
Set candidate
descriptors_target_all = lammps_ace_compute(target)
descriptors_candidate_all = lammps_ace_compute(candidate)
per_descriptor_residuals = average_columns(descriptors_target_all) - average_columns(descriptors_candidate_all)
M1_MAE = average(per_descriptor_residuals)
```

For testing the quality of the generated structures we created a Python script [1] to run the LAMMPS conjugate gradient minimization algorithm over our structures, to then be able to compare their similarity to the target. The minimization algorithm puts the atoms at their lowest potential energy where they are stable, which is why it is used for the comparison. We take the average and variance for the target and candidate structures into account when running the minimization. After this we get the average between the target and candidate and compare to ensure that the candidate was able to accurately represent the target. Fg. 2.2 shows an example histogram that would be made for to determine if the GRS method was able to cover the same distribution of space as the target it is intended to represent. It represents the target and GRS method aligning almost perfectly, while the poor representations, which derive from using inaccurate candidate structures for the system, are highly unaligned with the target. The final metric we use to evaluate the quality of the GRS method results is the total_mae score. The total_mae represents the total Mean Average Error (MAE) score between the candidate and target.

3. Results & Discussion . Providing a solution to the computational costs of using large training datasets for MLIPs, is one of the main goals of the GRS method. It intends to reduce the size of the datasets while retaining a sufficiently similar amount of information comparative to a larger training data set.

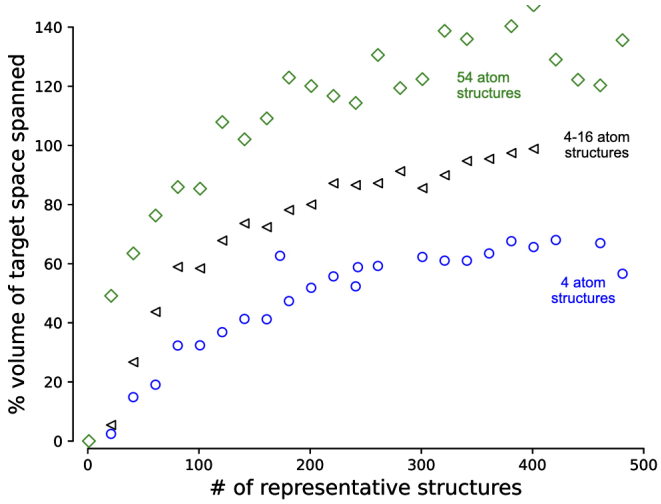


FIGURE 3.1. *Distribution of a four atom structure of tantalum (Ta) over a target space.*

As represented in Figure 3.1, we are looking at the number of representative structures it takes to cover the equivalent amount of environment that the target structure does. In this example the target structure was made of 373 structures to achieve 100% of space covered, while our representative structures could reach 100% with 100 GRS at a size of 128 atoms.

Some of the properties we are comparing include lattice constants, Poisson ratio, and Bulk Modulus. These properties are often used in the materials science field to describe the behavior and characteristics of materials, especially crystalline materials. As seen in Fig. 3.1, the GRS method is able to reasonably predict these properties of interest quantitatively. To compare we look at the ACE (GRS from QE) and SNAP columns, the determination of having ”good” or ”bad” results involves the GRS scores matching to the SNAP scores. The lattice constant, shear modulus, and Poisson ratio properties are all sufficiently predicted with our GRS method. The GRS predictions have a difference of .001 for the lattice constant property, .13 for the Poisson ratio property, and 10.68 for the shear modulus. Fg. 3.1 shows an example of the GRS method being used on a set of Ta data where with approximately 150 54-atom structures, you can represent an equivalent amount of the descriptor space, comparative to one large Ta structure.

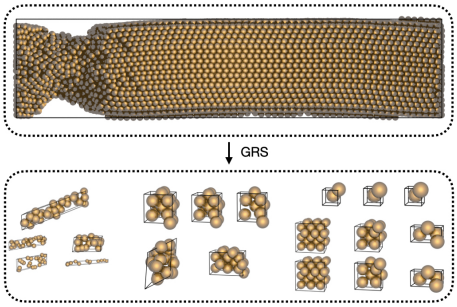


FIGURE 3.2. *A BCC Iron tensile test using an EAM interatomic potential, the loading direction is along [111] in a cylindrical wire geometry. Snapshot is post mechanical yielding showing significant necking (diameter reduction) and corresponding non-crystalline regions where plastic deformation has occurred.*

There is more space covered when using the GRS due to the fact that they are representing more unique small pieces of a structure, which provides more diversity to what the data represents.

In Fg. 1.3, a simple single element Tantalum (Ta) structures fingerprints are closely represented by a single GRS. It however is not always the case that a single GRS can accurately represent a system. In these cases, we aim to reproduce the target with multiple GRS that collectively reproduce the target..An example of a more complicated structure can be seen in Fg. 3.2. This example represents a system of body-centered cubic (BCC) Iron undergoing a tensile test causing deformations in the structure. The GRS that were made from this system represent multiple sections of the larger structure, it generates fingerprints for the structured, more "perfect" area, fingerprints at the cross-section of the deformation, as well as fingerprints for the area that is deformed.

	ACE GRS from QE (fit 1)	ACE GRS from QE (fit 2)	SNAP	DFT (snap paper)	Expt.
shear modulus (GPa)	63.08	53.51	73.76	∅	69.2
bulk modulus (GPa)	164.12	262.23	190.8	195.4	185.7
Poisson Ratio	0.33	0.4	0.46	∅	0.34
C11 (GPa)	188.83	279.04	270.28	∅	267
C44 (GPa)	63.08	53.51	73.4	75.3	82.5
lattice constant ($\mathring{A}$)	3.315	3.331	3.316	3.32	3.3

TABLE 3.1

A table showing the property scores of two ACE potentials trained on a 160-structure GRS set before hyperparameter optimization, SNAP, DFT from [18], and Experiment. Note: The Poisson's ratio is a dimensionless quantity.

To demonstrate that the structures generated by the GRS method aren't irrelevant energetically, DFT calculations were performed using the Quantum Espresso package on the 160 structure GRS set and small ACE MLIPs were trained on it. The potentials we used for this test were not optimized, but used as beginning estimations for the properties that could be obtained using the GRS set. It is reasonable to suspect that with further optimization on the potentials, the accuracy of the predicted properties would increase.

While the GRS method is able to go beyond perfect crystalline alloy systems, there are still restrictions and unknowns of what can be done using this method. The boundaries of this study include both the boundaries of the research field as a whole and the constraints of our software. There is a limitation to the type of data that the GRS system can work with, for example, running tests on structures of organic systems with elements such as oxygen and hydrogen would be one barrier. This is due to the large degree of angular resolution needed for the ACE fingerprints to capture the information of organic systems. In more organic systems other methods such as course-graining are more useful, as it is able to bridge the gap between the larger length and time scales of organic systems.

The GRS method however has shown promise to be useful as a new method for providing more complex and diverse sets of atomistic structure data.

4. Conclusion. In our research project we developed tools to benchmark and test our GRS method and validate the results we received from it. The main motivation of our research has been to reduce the amount of computationally and time expensive DFT needed to get important properties of materials systems. With the results from the testing, we have seen that our GRS are sufficient for use in this environment.

Another limitation of SQS is that it may be challenging to generate for materials with very high chemical degrees of freedom. This is because the evaluation of the fixed-lattice cluster correlations scales exponentially with the number of vertices in a cluster[14]. The efficient implementations of ACE fingerprints retain linear scaling with the number of neighbor atoms[12]. For example, high-entropy alloys and high-entropy materials have garnered a lot of attention in recent works[20, 15]. However, the ability to sample the random alloy phase of a 5-element alloy is limited with the

exponential scaling of the fingerprints. There may be additional motivation for the GRS method when it comes to fixed-lattice systems of this kind.

In conclusion, the validation work done has shown that there is promise for the GRS method to be useful in providing more complex and diverse sets of atomistic structure data for researchers. Our future work on this project would include attempting to add charge and magnetism in the properties of the generated structures, increasing the variety of the datasets created from this method. We will also consider evaluating the gradient of our effective potential using back propagation through the ML framework PyTorch.

5. Acknowledgements. This research was funded by the Minority Serving Institution Partnership Program Nuclear Security Advanced Manufacturing Enhanced by Machine Learning (MSIPP NSAM-ML) program. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly-owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] R. DRAUTZ, *Atomic cluster expansion for accurate and transferable interatomic potentials*, Physical Review B, 99 (2019), p. 014104. Publisher: American Physical Society.

[2] ———, *Atomic cluster expansion for accurate and transferable interatomic potentials*, Phys. Rev. B, 99 (2019), p. 014104.

[3] G. DUSSON, M. BACHMAYR, G. CSANYI, R. DRAUTZ, S. ETTER, C. VAN DER OORD, AND C. ORTNER, *Atomic cluster expansion: Completeness, efficiency and stability*, 2021.

[4] J. M. GOFF, C. SIEVERS, M. A. WOOD, AND A. P. THOMPSON, *Permutation-adapted complete and independent basis for atomic cluster expansion descriptors*, Journal of Computational Physics, 510 (2024), p. 113073.

[5] J. M. GOFF, C. SIEVERS, M. A. WOOD, AND A. P. THOMPSON, *Permutation-adapted complete and independent basis for atomic cluster expansion descriptors*, 2024.

[6] D.-A. HO AND O. BEYAN, *Biases in data science lifecycle*, 2020.

[7] A. JAIN, S. P. ONG, G. HAUTIER, W. CHEN, W. D. RICHARDS, S. DACEK, S. CHOLIA, D. GUNTER, D. SKINNER, G. CEDER, AND K. A. PERSSON, *Commentary: The Materials Project: A materials genome approach to accelerating materials innovation*, APL Materials, 1 (2013), p. 011002.

[8] M. KARABIN AND D. PEREZ, *An entropy-maximization approach to automated training set generation for interatomic potentials*, 2020.

[9] S. KIM, J. CHEN, T. CHENG, A. GINDULYTE, J. HE, S. HE, Q. LI, B. A. SHOEMAKER, P. A. THIESSEN, B. YU, L. ZASLAVSKY, J. ZHANG, AND E. E. BOLTON, *Pubchem 2023 update*, Nucleic Acids Research, 51 (2023), pp. D1373–D1380.

[10] K. LI, D. PERSAUD, K. CHOUDHARY, B. DECOST, M. GREENWOOD, AND J. HATTRICK-SIMPERS, *Exploiting redundancy in large materials datasets for efficient machine learning with less data*, Nature Communications, 14 (2023), p. 7283.

[11] A. O. LYAKHOV, A. R. OGANOV, H. T. STOKES, AND Q. ZHU, *New developments in evolutionary structure prediction algorithm uspex*, Computer Physics Communications, 184 (2013), pp. 1172–1182.

[12] Y. LYSOGORSKIY, C. V. D. OORD, A. BOCHKAREV, S. MENON, M. RINALDI, T. HAMMERSCHMIDT, M. MROVEC, A. THOMPSON, G. CSÁNYI, C. ORTNER, AND R. DRAUTZ, *Performant implementation of the atomic cluster expansion (PACE) and application to copper and silicon*, npj Computational Materials, 7 (2021), pp. 1–12.

[13] S. MOHR, L. E. RATCLIFF, L. GENOVESE, D. CALISTE, P. BOULANGER, S. GOEDECKER, AND T. DEUTSCH, *Accurate and efficient linear scaling dft calculations with universal applicability*, Phys. Chem. Chem. Phys., 17 (2015), pp. 31360–31370.

[14] J. M. SANCHEZ, F. DUCASTELLE, AND D. GRATIAS, *Generalized cluster description of multicomponent systems*, Physica A: Statistical Mechanics and its Applications, 128 (1984), pp. 334–350.

[15] A. SARKAR, L. VELASCO, D. WANG, Q. WANG, G. TALASILA, L. DE BIASI, C. KÜBEL, T. BREZESINSKI, S. S. BHATTACHARYA, H. HAHN, ET AL., *High entropy oxides for reversible energy storage*, Nature communications, 9 (2018), p. 3400.

[16] E. L. SIKORSKI, M. A. CUSENTINO, M. J. MCCARTHY, J. TRANCHIDA, M. A. WOOD, AND A. P. THOMPSON,

Machine learned interatomic potential for dispersion strengthened plasma facing components, The Journal of Chemical Physics, 158 (2023), p. 114101.

[17] L. Talirz, S. Kumbhar, E. Passaro, A. V. Yakutovich, V. Granata, F. Gargiulo, M. Borelli, M. Uhrin, S. P. Huber, S. Zoupanos, C. S. Adorf, C. W. Andersen, O. SchÃ¼tt, C. A. Pignedoli, D. Passerone, J. VandeVondele, T. C. Schulthess, B. Smit, G. Pizzi, and N. Marzari, *Materials cloud, a platform for open computational science*, Scientific Data, 7 (2020), p. 299.

[18] A. Thompson, L. Swiler, C. Trott, S. Foiles, and G. Tucker, *Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials*, Journal of Computational Physics, 285 (2015), pp. 316–330.

[19] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, *LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales*, Comp. Phys. Comm., 271 (2022), p. 108171.

[20] K.-K. Tseng, H. Huang, W. ren Wang, J.-W. Yeh, and C. Tsai, *Edge-dislocation-induced ultrahigh elevated-temperature strength of hfmonbtaw refractory high-entropy alloys*, Science and Technology of Advanced Materials, 23 (2022), pp. 642 – 654.

[21] M. E. Tuckerman and G. J. Martyna, *Understanding modern molecular dynamics: Techniques and applications*, J. Phys. Chem. B, 104 (2000), pp. 159–178.

[22] T. van Mourik, M. Bühl, and M.-P. Gaigeot, *Density functional theory across chemistry, physics and biology*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 372 (2014), p. 20120488.

[23] Z. Wang, J. Ma, and L. Zhang, *Finite element thermal model and simulation for a cylindrical li-ion battery*, IEEE Access, 5 (2017), pp. 15372–15379.

[24] T. Xie and J. C. Grossman, *Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties*, Phys. Rev. Lett., 120 (2018), p. 145301.

[25] A. Zunger, S.-H. Wei, L. G. Ferreira, and J. E. Bernard, *Special quasirandom structures*, Phys. Rev. Lett., 65 (1990), pp. 353–356.

OLLAMA-ASSISTED FUNCTION CALLS IN LEAP

PAT MUTIA* AND JACOB DAVIS†

Abstract. This project focuses on enhancing the operators' experience of using the ground station scheduling satellite software with the use of large language models (LLM). Operators are often faced with managing multiple tasks in time-constrained situations. The integration of a pre-trained transformer to perform function calling within the application promotes the ease of performing multi-step tasks and a user-friendly experience for all operators by performing the task with a well-formulated prompt. For the handling of sensitive information, *Ollama* was the optimal choice as an open-source API to well-trained LLMs such as *Llama3.1* and *Mistral3*, which would be responsible for interpreting the natural language of a user's query. *Ollama* allows for local hosting that opens up control over published information with a tradeoff of lowered processing speed depending on the number of parameters for the selected model. Moreover, LangChain and LangGraph were integrated into the generative chat interface to perform function calls after interpreting the operator's request embedded with the required arguments. Experimentation of the generative chat interface called "LEAP Chat" was performed manually to verify the accuracy. Future experiments to fine-tune prompting the model, sequential chaining, and human-in-the-loop integration would need to be performed to provide consistent and accurate results. Despite the early nature of this work, there is high potential that an LLM could be integrated to provide new capabilities and simplify workflows.

1. Introduction. Generally, pre-trained large language models (LLMs) are trained on substantial amounts of documents and data with methods like supervised and unsupervised learning to improve the stability and accuracy of the models. Adaptations of the LLM with proper prompts and fine-tuning have many possible capabilities (Figure 1.1), such as text generation, image generation, or pattern recognition [1].

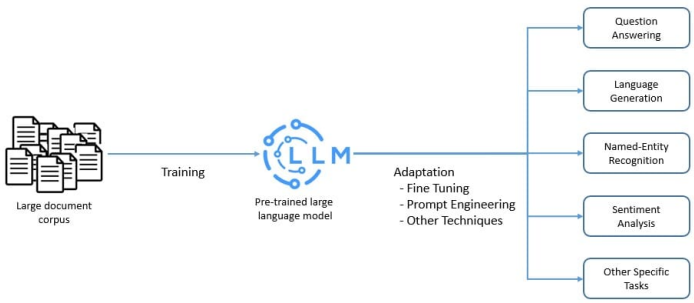


FIG. 1.1. *Available Functionalities with a Pre-trained LLM through Adaptation*

Recent developments within the machine learning community have progressed from generative and recognition models to multi-modal capabilities that can function call, retrieve information from databases, and leverage external tools [1]. In this work, we explore the use of a LLM to improve usability of a legacy satellite scheduling application termed as LEAP for Leading Edge Advancement Project. This application has a complex user interface (UI) with multi-step tasks that may call for a few or many parameters to properly work but can otherwise be easily described in a sentence. We aimed to improve the UI through allowing a function calling language model to parse an operator's prompt with any described parameter and perform the specified task(s) with their respective arguments. This would ease the user

*California Institute of Technology, pmutia@caltech.edu
†Sandia National Laboratories, jacdavi@sandia.gov

experience for operators by allowing multi-step tasks to be performed upon a single or a few queries and allow new operators to interact with the software with minimal familiarity.

2. Components Research. The LEAP software potentially handles sensitive information, so while the user interface can be enhanced with large language models, it must be done locally without returning input and output data back to the model for improvement. *Ollama*, an open-source API of high performing LLMs, allows local hosting of the models on instantiation. However, only some models have the capabilities to perform tool calling accurately [7].

2.1. Models. For the language model, compatibility of the LLM to perform function-calling was imperative to the task for the "LEAP Chat." In collaboration with the AI-researching organization Glaive, an option for the model was *Llama-3-Groq-Tool*. Built based on the Meta *Llama 3* model, it scored the highest on the Berkeley Function Calling Leaderboard in July 2024 at 90.76% overall accuracy for 70B (billion parameters) and 89.06% for 8B. With the same structure as *Llama3*, the performance of the model was enhanced with full fine-tuning and Direct Preference Optimization (DPO) to optimize for function-calling. From the developers, the recommended approach was a routing system to analyze user queries for their nature and requirements. For tool use or API interactions, the model should be selected whereas general queries should be directed to a general-purpose language model like *Llama3* [10].

Additionally, the *Ollama* database has access to the Meta *Llama 3.1* model (8B, 70B, 405B), which is one of the highest performing closed source AI models that competes with top AI models such as GPT-4 and Claude 3.5 Sonnet in capabilities of general knowledge, flexibility, math, and tool use. The recent upgraded versions have longer context length of 128K, advanced tool use, and stronger reasoning [2].

Comparing the extensive testing and scoring of the Llama3.1 versus other models in Figure 2.1, Llama 3.1 model performs better in the general category and is comparable in tool use, which are the main capabilities crucial to the "LEAP Chat." The model utilizes a decoder-only transformer architecture with minor adaptations for training stability while also using iterative post-training of supervised fine-tuning (SFT) and DPO to improve performance.

2.2. LangChain and LangGraph. For the adaptation of the model to perform function calls for LEAP, components from LangChain and LangGraph are extremely useful. LangChain provides the standard interface to interact with models and other components, which is useful for simple straight-forward chains and retrieval flows[4]. The library contains necessary building blocks to templatize prompts and to dynamically select and manage model inputs[1].

LangGraph is also a helpful library for state-based applications with LLMs to create agents that determine the flow of the LLM. The low-level framework allows for precise control over the flow and state of the application. Moreover, LangGraph also enables access to human-in-the-loop and memory features that enable the model to perform supervised function calling and recall previous values starting from the initialization of the model [5].

For the intended structure of "LEAP Chat" (Figure 2.2), the model must process the user's input as a query and use the LangChain components to understand the nature of the query and the arguments. Then the model must determine what tools with various function calls are at it's disposal and decide which to call based on the nature of the task and the prompt fed into the model. An agent created using LangGraph would help choose what function call operations to perform next such as using a tool or asking for more user input. The memory would store previous chat history between the user and the model. This

Category / Benchmark	Llama 3.1 405B	Nemotron 4 340B Instruct	GPT-4 (0125)	GPT-4 Omni	Claude 3.5 Sonnet
General MMLU (0-shot, CoT)	88.6	78.7 (non-CoT)	85.4	**88.7**	88.3
MMLU PRO (5-shot, CoT)	73.3	62.7	64.8	74.0	**77.0**
IFEval	**88.6**	85.1	84.3	85.6	88.0
Code HumanEval (0-shot)	89.0	73.2	86.6	90.2	**92.0**
MBPP EvalPlus (base) (0-shot)	88.6	72.8	83.6	87.8	**90.5**
Math GSM8K (8-shot, CoT)	**96.8**	92.3 (0-shot)	94.2	96.1	96.4 (0-shot)
MATH (0-shot, CoT)	73.8	41.1	64.5	**76.6**	71.1
Reasoning ARC Challenge (0-shot)	**96.9**	94.6	96.4	96.7	96.7
GPQA (0-shot, CoT)	51.1	-	41.4	53.6	**59.4**
Tool use BFCL	88.5	86.5	88.3	80.5	**90.2**
Nexus	**58.7**	-	50.3	56.1	45.7
Long context ZeroSCROLLS/QuALITY	**95.2**	-	**95.2**	90.5	90.5
InfiniteBench/En.MC	**83.4**	-	72.1	82.5	-
NIH/Multi-needle	98.1	-	**100.0**	**100.0**	90.8
Multilingual Multilingual MGSM (0-shot)	**91.6**	-	85.9	90.5	**91.6**

(a) Comparison of *Llama 3.1* 405B to Other Models

Category / Benchmark	Llama 3.1 8B	Gemma 2 9B IT	Mistral 7B Instruct	Llama 3.1 70B	Mixtral 8x22B Instruct	GPT 3.5 Turbo
General MMLU (0-shot, CoT)	**73.0**	72.3 (5-shot, non-CoT)	60.5	**86.0**	79.9	69.8
MMLU PRO (5-shot, CoT)	**48.3**	-	36.9	**66.4**	56.3	49.2
IFEval	**80.4**	73.6	57.6	**87.5**	72.7	69.9
Code HumanEval (0-shot)	**72.6**	54.3	40.2	**80.5**	75.6	68.0
MBPP EvalPlus (base) (0-shot)	**72.8**	71.7	49.5	**86.0**	78.6	82.0
Math GSM8K (8-shot, CoT)	**84.5**	76.7	53.2	**95.1**	88.2	81.6
MATH (0-shot, CoT)	**51.9**	44.3	13.0	**68.0**	54.1	43.1
Reasoning ARC Challenge (0-shot)	83.4	**87.6**	74.2	**94.8**	88.7	83.7
GPQA (0-shot, CoT)	**32.8**	-	28.8	**46.7**	33.3	30.8
Tool use BFCL	**76.1**	-	60.4	84.8	-	**85.9**
Nexus	**38.5**	30.0	24.7	**56.7**	48.5	37.2
Long context ZeroSCROLLS/QuALITY	81.0	-	-	90.5	-	-
InfiniteBench/En.MC	65.1	-	-	78.2	-	-
NIH/Multi-needle	98.8	-	-	97.5	-	-
Multilingual Multilingual MGSM (0-shot)	**68.9**	53.2	29.9	**86.9**	71.1	51.4

(b) Comparison of *Llama 3.1* 8B and 70B to Other Models

FIG. 2.1. *Scores of* Llama 3.1 *Accuracy Depending on Task Type*

structure was proposed based on the rigid structure of the satellite API, which requires parsing through the user's query for required parameters in a specific format. Moreover, the memory is helpful for allowing reference to previously mentioned information from the user and/or the model.

3. Experimentation. For the LEAP scheduling satellite software, the initial task was to experiment and focus on the ease, accuracy, and consistency of the LLM when checking on the status of a specific or multiple satellites, which requires date/time and satellite name arguments to be processed and formatted accordingly. These parameters would then be used in a subsequent API invocation to the satellites. Therefore, the tools for this experimentation included the following functions: format the time, get an end time, and check the status of

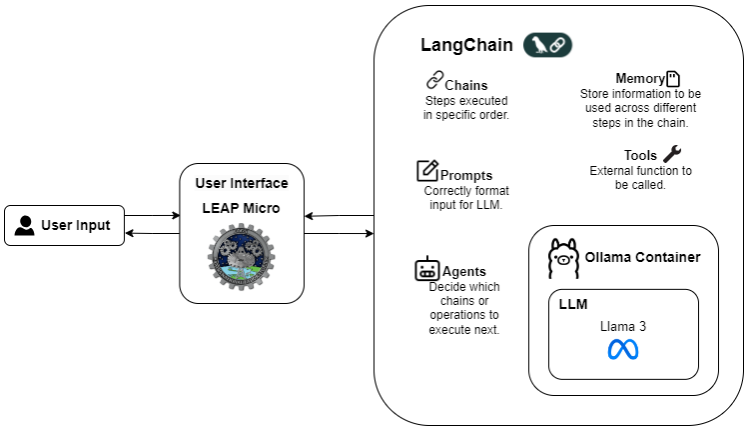


FIG. 2.2. *LEAP Chat architecture*

the satellites with those parameters.

For the model for "LEAP Chat", the *Llama-3-Groq-Tool* performed well on single tool calls in all versions of the "LEAP Chat" but was unable to recognize sequential tool calls, so the model selected for further fine-tuning was *Llama 3.1*.

3.1. LangChain LEAP Chat. For the initial model proposal, LangChain was the only structure used as the task appeared strictly sequential; i.e. obtain a start time, obtain an end time, format the two times, and then feed those times into a tool for API invocation. However, prompting the model demonstrated that very specific queries, which would allow the LLM to recognize multiple function calls, were required such as listing each tool to select. With such queries triggering multiple function calls, the ease and functionality of the "LEAP Chat" was minimized. Moreover, there was no memory specified in the structure of the chain, so the model was unable to recall outputs from previous tools in the chain. This version of "LEAP Chat" highlighted the importance of the queries and memory in the chain, but the multiple, sequential tool calls may need to be coded into the structure of the model for this task.

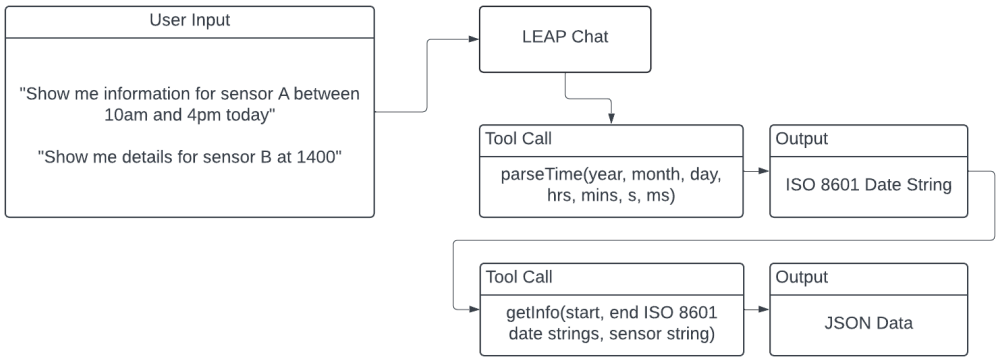
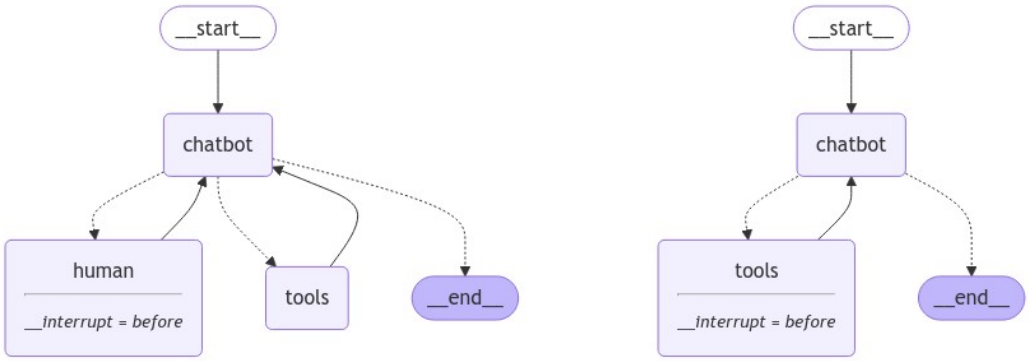


FIG. 3.1. ***Example of LangChain Flow.*** *The user can prompt the LEAP Chat like demonstrated. The model will parse the arguments within the query. The LangChain should perform multiple tool calls to reach an output for the user.*

3.2. LangGraph LEAP Chat. With the addition of LangGraph, the model gained an agent to cycle through the multiple tools for the "LEAP Chat," which proved more successful in calling multiple function calls. However without human intervention, the model would continue performing the function calls that were unnecessary and yielded inaccurate results as seen in Figure 3.2(a). Even with a node to allow the model to rely on human feedback, it would not detect that that its responses were erroneous, so it would not end up using the human node. Adding a human-in-the-loop structure to interrupt before each function call was another strategy to promote accuracy and unnecessary calls at the expense of losing the autonomy of the "LEAP Chat" to perform the tool calls on its own and consequently the speed of the query in Figure 3.2(b). With this strategy, the tool calls were more intentional and accurate with the option to correct arguments, but if the model recognized that it needed to perform multiple tool calls and then was interrupted, the model failed to continue calling all the tools even when the arguments were accurate.



(a) LangGraph Agent with Human Node

(b) LangGraph Agent with Human-In-The-Loop Structure

FIG. 3.2. *LangGraph Agents of LEAP Chat*

In Figure 3.2, the flow of LEAP Chat is demonstrated with the use of a human node for dependency on a human for content versus human-in-the-loop interruptions for corrections. In Figure 3.2(a), the chat interface starts up with a chatbot to request user input or queries. With a prompt and a query, the chatbot that represents the model would decide on whether to request for content from the human node or to perform a tool call. It would cycle through those until it believes it has reached completion. Then the model will prompt the user again and repeat the steps until an end condition is found. In Figure 3.2(b), the chat interface starts up similarly. The chatbot will only be able to decide between tools to call and be interrupted after trying to set up the tool call. The interface will let the user accept the tool call with the parameters the model has parsed or deny it and subsequently edit the arguments. Then the model will perform that cycle until an end condition is called as well.

4. Future Research. The initial structures and experiments demonstrated some common issues with performing multiple function calls, making sequential function calls, and having general inaccuracies that cannot be edited at the expense of functionality. Therefore, the future challenge is fine-tuning of the LLM to perform multiple and/or sequential function calls, which may be resolved with additions and modifications to the chat interface's model structure and prompt engineering.

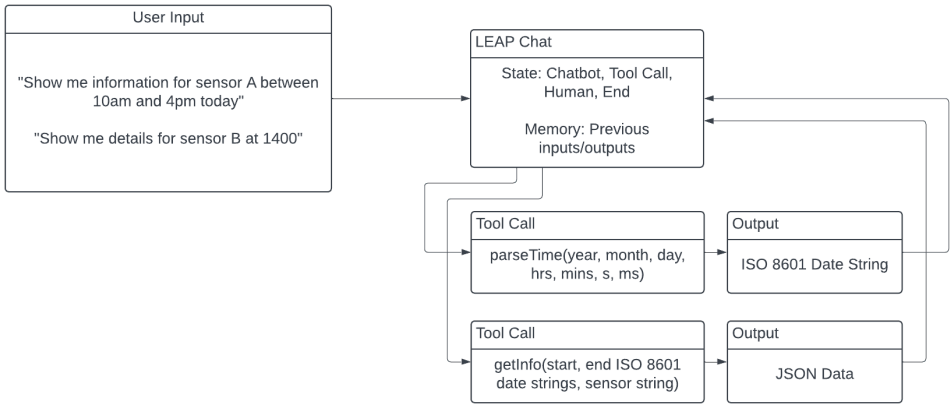


FIG. 3.3. ***Example of LangGraph Flow.*** *The user can prompt the LEAP Chat like demonstrated. The model will parse the arguments within the query. Unlike the LangChain flow in Figure 3.1, there is a LangGraph agent that will determine which state that model should enter to perform the necessary tool calls, rather than sequential tool calls, to reach an output for the user. For Human-In-The-Loop, there will be pauses before each tool call asking the human whether to continue.*

4.1. Prompting. The prompt was shared for all versions of "LEAP Chat." It had set a persona for the model as a helpful assistant responsible for answering questions with explicit mentions of a default argument for formatting date and time if it was not mentioned in the query. Further experimentation in prompting for the "LEAP Chat" could include mentioning steps to complete a task or providing examples of a query and how it should be interpreted as input. Based on operators' preferences, the output could also be formatted to be more succinct [8]. Overall, these choices would allow the code to become simpler and cleaner while increasing the accuracy and experience of the chat for the operators since the model is being requested to be more conscious of specific aspects of query.

Occasionally, the model would fail to be able to accurately parse through the query for arguments required for the function calls. For instance, the time "1200" that represents the time "12:00" could be seen as 1,200 hours. Apart from explicit mention in the prompt, default values were integrated into the tool calls rather than the model deciding on whether to reprompt the operator or the model choosing default values, leading to overly complex prompting and code. This challenge can be resolved with libraries Instructor and Pydantic. Instructor works with Pydantic, a data validation library [9], to extract arguments from complete and incomplete messages [6]. Pydantic uses schema validation to ensure the correct values and their types are passed into the function calls [9]. These libraries would prove beneficial in creating a robust model that can take incomplete arguments like time and date values of many formats (e.g. colon inclusion, 12-hour format) and raise less errors.

4.2. Sequential Chains with Memory. For the difficulty with multiple sequential tool calls, the use of sequential chains, a specific type of LangChain, would allow each step in the chain to pass on its outputs as respective inputs for the next step. Moreover, these chains have memory that can be integrated into them to retain information of previous inputs and outputs [3]. In the context of "LEAP Chat", the multi-stepped tasks would occur more consistently with assurance that the correct previous output would be the argument for the proceeding function call. In Figure 4.1, an example of a sequential chain is demonstrated, where input after parsing for parameters is passed through a sequential chain of tool calls. Then the parameters are passed into another chain of tool calls. Then the final chain outputs

the result.

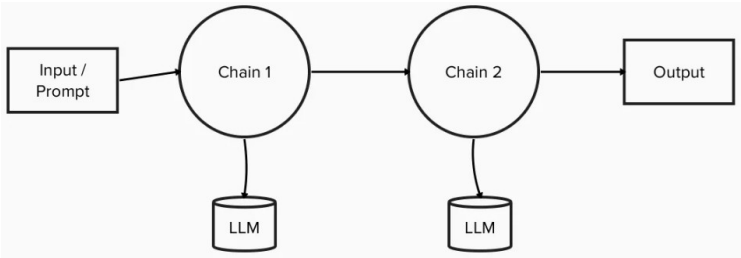


FIG. 4.1. *Example of Sequential Chain Structure*

4.3. Expandability of LEAP Chat. Although the LEAP software has a multitude of tasks per application, the feasibility and ease of creating a chat interface for the application is possible. Multi-stepped tasks can be made into different sequential chains with commonly shared tools such as formatting time and getting start and end times. The LangGraph can then be additionally expanded to have a node run these sequential chains as well as the human node and tool node. Therefore, the complexity of LEAP can be minimized through a function calling LLM to perform multi-step tasks, especially as the model gains greater consistency with structure adjustments of sequential chains to ensure multiple function calls occur.

5. Conclusion. The purpose of this research was to show the potential of improving the LEAP satellite scheduling software through the integration of an LLM. The potential was qualitatively evaluated through the ease of converting one function of an application within the LEAP software with the aid of LangChain and LangGraph in addition to the LLM's consistency and accuracy from a variety of prompts. We found that *Llama 3.1* model had more flexibility in breaking down the query into the proper parameters for the tool calls than the *Llama-3-Groq-Tool* model. Moreover, we observed LangGraph worked well on performing multi-step tasks independent of LangChain, but LangChain was crucial for reaching consistent results for tool calls that needed to be called sequentially. Although consistency and accuracy must still be improved, LEAP Chat demonstrates how LEAP software can incorporate a chat interface to perform well-defined multi-step tasks that would improve the learning curve of operating the software and may allow the operator to perform trivial tasks easily with a good query and interaction with a model.

REFERENCES

[1] K. CHANDRAKANT, *Introduction to LangChain*, Baeldung, (2024).
[2] A. DUBEY ET AL., *The Llama 3 Herd of Models*, Meta, (2024).
[3] P. KRAMPAH, *Langchain — Sequential LLM Calls*, Medium, (2023).
[4] LANGCHAIN AI, *LangChain*, LangChain, (2024).
[5] ———, *LangGraph*, LangChain, (2024).
[6] J. LIU, *Instructor*.
[7] OLLAMA, *Ollama*.
[8] OPENAI, *Prompt engineering*, OpenAI Platform, (2024).
[9] PYDANTIC, *Pydantic*.
[10] N. A. RICK LAMERS ET AL. AND S. CHAUDHARY, *Introducing Llama-3-Groq-Tool-Use Models*, groq, (2024).

SCIENTIFIC MACHINE LEARNING FOR SURROGATE MODELING

KWESI A. OHENE-OBENG* AND KATHRYN MAUPIN†

Abstract.
In computational simulations, the advancement of surrogate models is paramount for the efficient prediction of experimental outcomes. Previous studies have demonstrated the efficacy of Gaussian processes in modeling the discrepancies between simulation predictions and experimental observations. However, there is a growing interest in harnessing machine learning (ML) and artificial intelligence (AI) techniques to further enhance predictive accuracy and computational efficiency. This study aims to explore the application of various ML algorithms, specifically Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN), in the construction of surrogate models, with a particular focus on identifying the optimal combination of model parameters (density, velocity, and location) for accurate prediction of experimental data.Using simulation data, MLP, RNN, and CNN models were trained and evaluated. The MLP model demonstrated superior performance, which was further refined through calibration to reduce discrepancies with experimental data. This methodology was employed to discern the optimal parameter combinations and elucidate the underlying physical mechanisms, thereby enhancing prediction accuracy. Incorporating domain-specific knowledge into ML models holds promise for achieving even greater precision and reliability in predictions. This study underscores the potential of advanced ML techniques to advance the accuracy of surrogate models in complex simulation environments.

1. Introduction. The quantification of uncertainty of a computational model is critical to establishing its credibility and trustworthiness for predicting the quantities of interest. Uncertainty quantification (UQ) studies, whether addressing inverse or forward problems, rely on repeated evaluations of the computational model. A significant challenge in UQ arises from the computational demands of these models, which can be computationally expensive. Additionally, these models often do not perfectly mirror reality, and uncertainties are introduced by the imprecision in choosing a functional form and determining the value of input parameters associated with the models.

Numerous studies have addressed these challenges by developing surrogate models, particularly in situations where the computational models are prohibitively expensive to evaluate repeatedly. Approaches such as Gaussian Processes have been employed to capture model form error and provide insights into the underlying physics that may be absent from traditional physics-based computational models [3, 5, 6]. This study focuses on constructing surrogate models using machine learning algorithms, specifically neural networks, and subsequently quantifying model form error to shed light on the underlying physical phenomena.

Following this introduction, the paper presents a brief background of the study in Section 2, the methods employed in Section 3, and the results obtained in Section 4. Concluding remarks are provided in the final section.

2. Background. High-fidelity models, which are often treated as black-box representations of first-principle models, may not always offer sufficient insights or explanations of the phenomena within the studied domain. When these models fail to adequately represent or explain the phenomena, this deficiency is known as model discrepancy. Addressing such errors may require a more in-depth examination of the model's underlying assumptions and representations to improve its accuracy and reliability [7]. Numerous methodologies have been developed to quantify discrepancies in models, often relying on statistical techniques to compare model outputs with actual data. While these approaches can effectively identify errors or biases in predictions based on observed data, they fall short in addressing how these inaccuracies affect predictions for unobserved or future scenarios. This is a signifi-

*University of Texas at El Paso, kaohene@sandia.gov
†Sandia National Laboratory, kmaupin@sandia.gov

cant limitation, as the primary objective of many modeling efforts is to provide accurate predictions for situations where data is not yet available.

To construct a meaningful representation of model inadequacy, it is essential to integrate both qualitative and quantitative insights into the phenomena being modeled. This means that any framework designed to quantify uncertainty in predictions must be informed by our existing knowledge of the system and the mechanisms through which errors arise. By acknowledging the underlying causes of model discrepancies and incorporating this understanding into the modeling process, we can develop more robust methods for representing uncertainty. Such an approach not only enhances the reliability of predictions but also contributes to a more comprehensive understanding of the model's limitations and strengths.

The computational challenges often encountered in modelling complex physical phenomena stem from the costly and time-consuming nature of detailed analysis and simulations. Achieving a level of accuracy comparable to real experimental data can be particularly taxing. To alleviate this burden, surrogate modeling has emerged as a powerful tool, offering approximations that effectively mimic high-fidelity models [11] and to make predictions that closely align with reality.

2.1. Existing Literature. This study builds upon the foundational work of [5], which utilizes Gaussian Processes to construct surrogate models within a Bayesian framework for model calibration and correction. A first-order Taylor series expansion is used to address parameter non-identifiability and explore how different model discrepancy formulations affect Bayesian calibration and prediction accuracy [4]. This research demonstrates that simpler formulations can improve extrapolation beyond the calibration domain. Maximum Likelihood Estimation (MLE) can effectively quantify model form uncertainty, as demonstrated in [6]. The MLE is shown to offer reliable composite predictions by integrating model-to-model variance and prediction errors, as compared with Bayesian methods using uninformative priors. This is demonstrated with applications to concrete creep and laser peening.

In our study, we incorporate scientific machine learning techniques, specifically neural networks, to develop surrogate models. We apply Monte Carlo dropout methods for prediction and subsequently perform model calibration. We then identify model discrepancies and compare our findings with the methods proposed by [5].

3. Methods. In the methods section, we outline the approach employed to analyze both experimental and simulated data, providing a comprehensive overview of our experimental setup and machine learning methodologies. To analyze these datasets, we applied several machine learning algorithms, including Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNN), to capture and predict the complex relationships inherent in the data. The performance of these algorithms was evaluated through rigorous statistical analysis and validation procedures to ensure accuracy and robustness.

3.1. Neural Networks. This study utilized neural networks, specifically Multi-Layer Perceptrons (MLPs), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs), for surrogate modeling. Neural networks (NNs) are a powerful class of algorithms designed for function approximation through data-driven methods. They structure information hierarchically, with features distributed across multiple layers. Recent research has increasingly focused on the potential of neural networks for uncertainty quantification, with some studies employing deep neural networks to address challenges associated with numerical simulations [10].

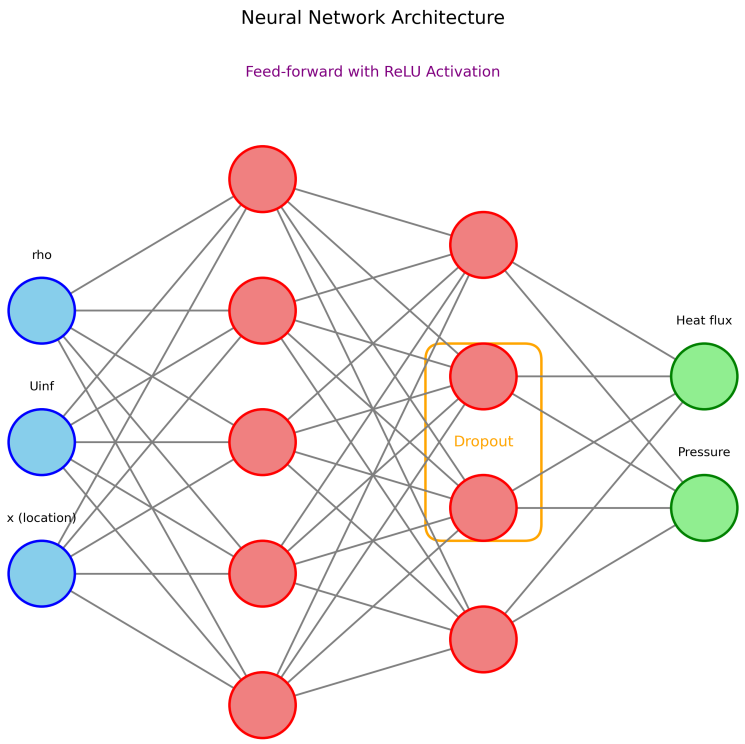


Fig. 3.1: Neural Network Architecture: The input neurons, shown in blue correspond to the the three input parameters: density(rho), velocity(Uinf), and spatial location (x). These inputs are processed through the hidden layers, shown by red circles. The green output neuron correspond to the model outputs: heat flux and pressure.

The surrogate model in the present study is a feed-forward neural network with ReLU activation. A schematic of the NN architecture is shown in Figure 3.1. The network consists of three input neurons, which correspond to the variables: density, velocity, and location. These input neurons are represented by blue circles on the left side of the diagram. The neural network includes two hidden layers, with neurons depicted as red circles. These hidden layers are where the data is processed and feature extraction occurs, allowing the network to capture complex patterns in the data. Each neuron in these hidden layers is connected to neurons in adjacent layers, forming a fully connected architecture.

One of the hidden layers includes a dropout layer, highlighted by an orange rectangle. Dropout regularization technique is used to prevent overfitting by randomly setting a fraction of the input units to zero during the training process. The output layer of the network consists of two neurons, represented by green circles on the right side, corresponding to the target variables: heat flux and pressure. These output neurons generate the final predictions of the network after processing the inputs through the hidden layers.

3.2. Monte Carlo Dropout. Monte Carlo (MC) Dropout combines Monte Carlo methods with dropout regularization techniques to enhance predictive performance in neural

networks. This approach aims to mitigate overfitting and underfitting by minimizing the loss function, thus improving the model's robustness, particularly in the context of uncertainty estimation. Further details can be found [2].

When fitting a Multi-Layer Perceptron (MLP) model, MC Dropout introduces randomness into the neural network's forward passes, even during the prediction phase. Dropout is applied during training to prevent overfitting by randomly deactivating a fraction of each neuron's outputs in each forward pass. MC Dropout extends this dropout mechanism to the inference phase, where each prediction is subject to random neuron deactivation. By performing multiple forward passes with different dropout configurations, a distribution of predictions is generated. That is, given multiple realizations of the MLP, $\hat{y}_t$, the final prediction is given by

$$\hat{y} = f_{MLP}(x; \theta) \tag{3.1}$$

$$\hat{y} = \frac{1}{N} \sum_{t=1}^{N} \hat{y}_t, \tag{3.2}$$

and the corresponding uncertainty is calculated as

$$\sigma_{\hat{y}} = \frac{1}{N} \sum_{t=1}^{N} (\hat{y}_t - \hat{y})^2. \tag{3.3}$$

Here, N is the number of predictions. This method offers several key advantages. First, it provides a straightforward yet effective means of estimating prediction uncertainty without necessitating major alterations to the model architecture or the use of complex Bayesian methods. Secondly, it allows for a probabilistic interpretation of the network's outputs, which is valuable in contexts where understanding the confidence in predictions is as critical as the predictions themselves. Lastly, MC Dropout builds upon existing dropout mechanisms, enabling its efficient integration into most neural network frameworks with minimal computational overhead.

3.3. Parameter Calibration. The principal objective of model calibration is to refine the parameters of a model to ensure that its predictions align closely with empirical data. This process involves systematically adjusting the model parameters to minimize discrepancies between predicted outcomes and observed measurements. Typically, this is accomplished through the minimization of a loss function that quantifies the deviation between model predictions and actual data [1, 9]. Let θ denotes the model parameters. With experimental data denoted by y_{exp} and model predictions denoted by y_{pred}, we define the loss function as $L(\theta)$,

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(y_{pred,i}(\theta) - y_{exp,i} \right)^2. \tag{3.4}$$

Here, N is the number of data points, $y_{\text{pred},i}(\theta)$ is the predicted value for the i^{th} data point, and $y_{\text{exp},i}$ is the corresponding experimental measurement. The goal is to find the parameter set θ^* that minimizes this loss function:

$$\theta^* = \arg\min_{\theta} L(\theta). \tag{3.5}$$

In practice, optimization algorithms such as gradient descent, stochastic gradient descent, or more sophisticated techniques like Bayesian optimization are employed to find the optimal parameters. These algorithms iteratively adjust the parameters to reduce the loss function, improving the model's fit to the experimental data.

4. Example Application. The data utilized in this study comprise both experimental and simulated datasets. The simulated dataset was divided into training and testing sets, with 80% allocated for training and 20% reserved for testing. Additionally, from the training data, 20% was further set aside to create a validation set for tuning the neural network model.The experimental data were obtained from the Large Energy National Shock (LENS) Tunnel at the Calspan-University at Buffalo Research Center, focusing on a laminar hypersonic double-cone experiment. In this experiment, a 25°/55° double-cone was exposed to conditions defined by freestream parameters, including density (ρ), velocity (U), and temperature (T). Measurements were taken for heat flux (q) and pressure (p) at multiple locations along the cone, under two distinct scenarios referred to herein as 'Run 06' and 'Run 07' [8]. Each scenario involved distinct experimental conditions, with a single test conducted per condition set. Simulated data were generated using the high-fidelity code, Sandia Parallel Aerodynamic Reentry Code (SPARC). This simulated data served as a basis for developing surrogate models for further analysis.

4.1. Preliminary Analysis. The data consisted of parameters; Density (rho), Velocity (Uinf), Vibrational Temperature and Temperature (Tinf). Relationships between the key parameters and the target outcomes for Run 06 scenario are shown in Figure 4.1. Velocity demonstrates the strongest correlation with heat flux, with a correlation coefficient of 0.384. This moderate positive correlation indicates that as velocity increases, heat flux also tends to increase. This suggests that velocity is likely a critical predictor of heat flux and should be given significant attention in any predictive modeling efforts. In contrast, the temperature parameters show an extremely weak correlation with heat flux (correlation coefficient of 0.003), indicating that these parameters may not significantly influence heat flux predictions. Density also exhibits a weak correlation with heat flux (correlation coefficient of 0.018), suggesting a minimal linear relationship with heat flux and therefore a lesser role as a predictor. Location shows a small negative correlation with heat flux (correlation coefficient of -0.049). However, it is important to note that correlations capture only the linear relationships.

For pressure, location is the most strongly correlated parameter, with a correlation coefficient of 0.822. This strong positive correlation suggests that location plays a significant role in influencing pressure, indicating that as the distance from the tip of the cone increases, pressure also increases, in contrast to the behavior observed with heat flux. Velocity shows a weak positive correlation with pressure (correlation coefficient of 0.080), suggesting a minor but positive relationship between the two. The temperature parameters again show very weak negative correlations with pressure (correlation coefficient of -0.021), indicating minimal impact on pressure predictions.

The corresponding correlation heatmap for Run 07 is shown in Figure 4.2. Velocity remained the most strongly correlated parameter with heat flux, with a correlation coefficient of 0.534. This stronger correlation compared to Run 06 suggests that velocity is even more critical in predicting heat flux in this scenario. Location also shows a positive correlation with heat flux, though weaker (correlation coefficient of 0.122), indicating that location has some influence on heat flux, albeit less than velocity. The temperature parameters exhibit an extremely weak and negative correlation with heat flux (correlation coefficient of -0.001), further supporting the notion that temperature has little to no impact on heat flux. Density also shows a very weak negative correlation with heat flux (correlation coefficient of -0.011),

reinforcing its limited role as a predictor.

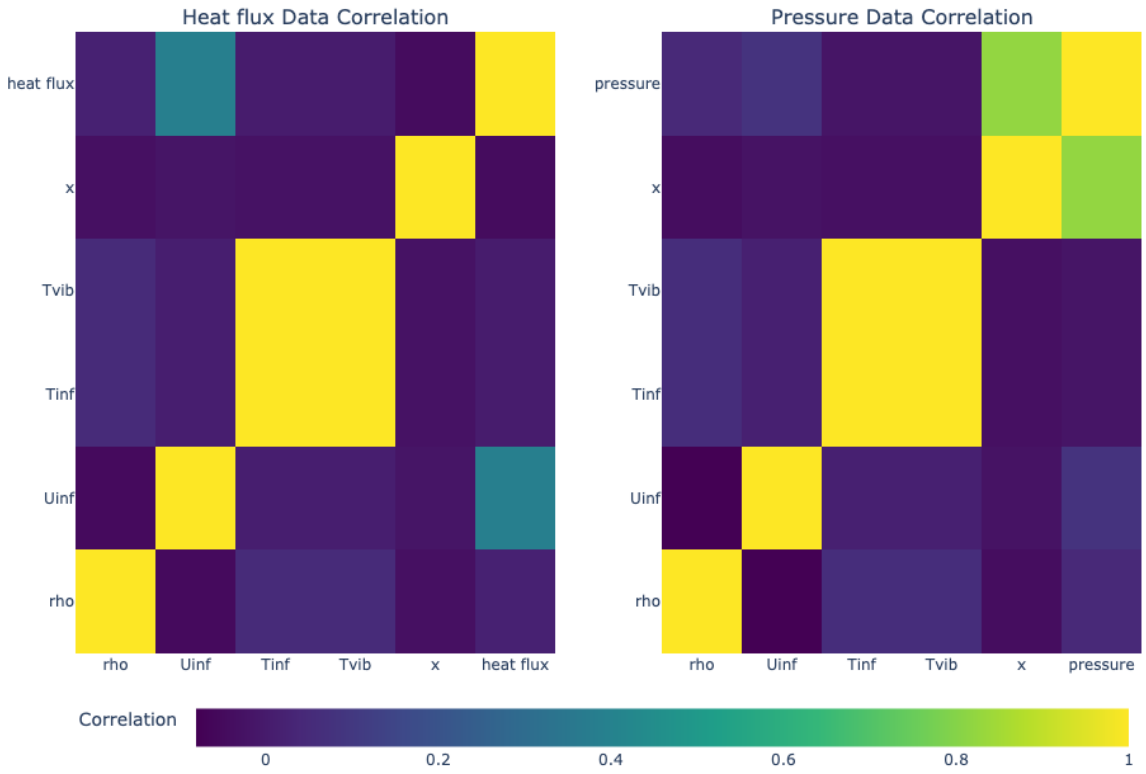


Fig. 4.1: Heatmap of correlation coefficients, calculated between inputs and outputs of Run 06. The yellow color indicates a perfect correlation (correlation coefficient of 1), while deeper shades transitioning towards purple signify a strong negative correlation (correlation coefficient approaching 0).

For pressure, location again emerges as the most strongly correlated parameter (correlation coefficient of 0.801), highlighting its importance in pressure prediction. Velocity shows a moderate positive correlation with pressure (correlation coefficient of 0.389), suggesting a meaningful relationship where increases in velocity correspond to increases in pressure. The temperature parameters, similar to heat flux, display extremely weak positive correlations with pressure (correlation coefficient of 0.007), indicating negligible linear relationships. Density has a very weak negative correlation with pressure (correlation coefficient of -0.023), suggesting it plays a minor role in predicting pressure. The analysis across both scenarios indicates that velocity consistently shows a meaningful correlation with both heat flux and pressure, underscoring its importance as a predictor. Conversely, temperature parameters consistently exhibit near-zero correlations, suggesting their negligible impact. Consequently, this study will exclude temperature from further modeling efforts, focusing instead on parameters with stronger predictive power, such as velocity, density (due to its significance on heat flux), and location.

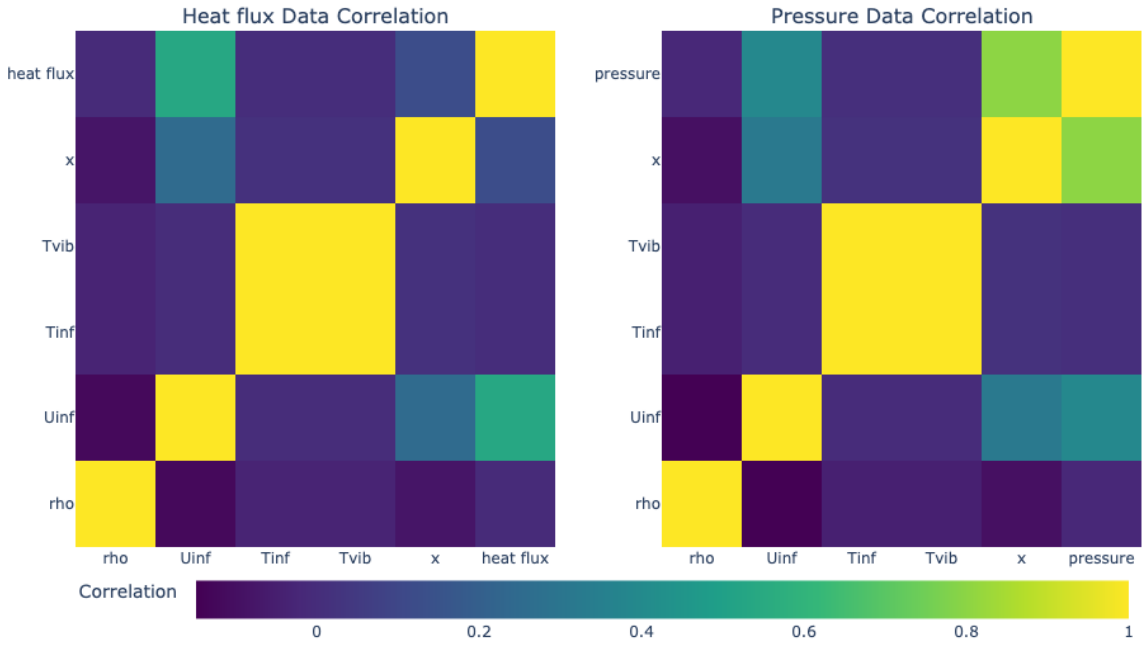


Fig. 4.2: Heatmap of correlation coefficients, calculated between inputs and outputs of Run 07. The yellow color indicates a perfect correlation (correlation coefficient of 1), while deeper shades transitioning towards purple signify a strong negative correlation (correlation coefficient approaching 0)

4.2. Relationship between Inputs and Outputs. To gain a qualitative understanding of the relationships between the inputs and the outputs in the Run 06 scenario we show 3D scatterplots in Figure 4.3. The left plot shows that heat flux generally increases with distance from the tip of the cone. The regions farther from the tip (yellow) tend to have higher heat flux values compared to the regions closer to the tip (purple). Higher heat fluxes are associated with specific ranges of density and velocity. The data points tend to cluster around certain values of density and velocity, suggesting a non-linear relationship where the heat flux depends on the interplay between density and velocity. Pressure (right plot) also appears to increase with distance from the tip of the cone. However, the increase in pressure is less steep than the increase in heat flux as you move farther from the tip. The distribution of pressure shows a more complex relationship with density and velocity compared to heat flux. The points appear more spread out across the ranges of density and velocity, indicating that pressure might be influenced by additional factors or that the relationship is non-linear.

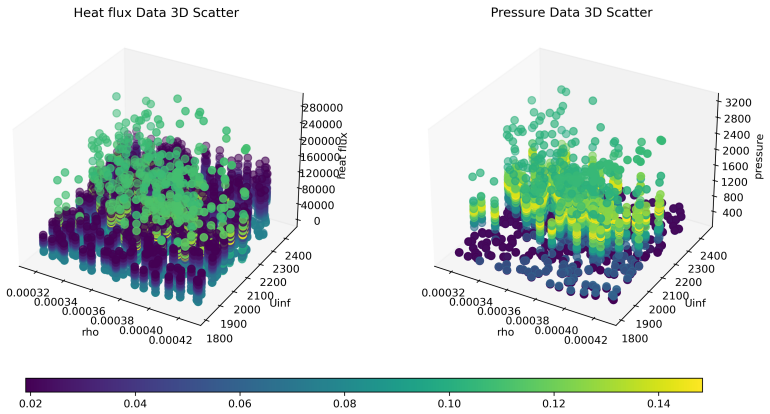


Fig. 4.3: 3D Scatterplot for Run 06 Scenario. The yellow color indicates a greater distance from the cone, while a dimmer hue suggests proximity to the cone.

We similarly analyze a 3D scatterplot of Run 07. Figure 4.4 depicts a different relationship, the heat flux values still increase with distance from the tip of the cone, but the overall range of heat flux appears to be lower compared to the Run 06. The maximum heat flux value is around 200,000, indicating that this dataset or the specific run being analyzed might involve less extreme conditions. The clustering of points suggests that the relationship between heat flux, density, and velocity is consistent with Run 06, where heat flux values are influenced by specific combinations of density and velocity. The scatter is more concentrated, indicating perhaps a more uniform distribution of conditions.

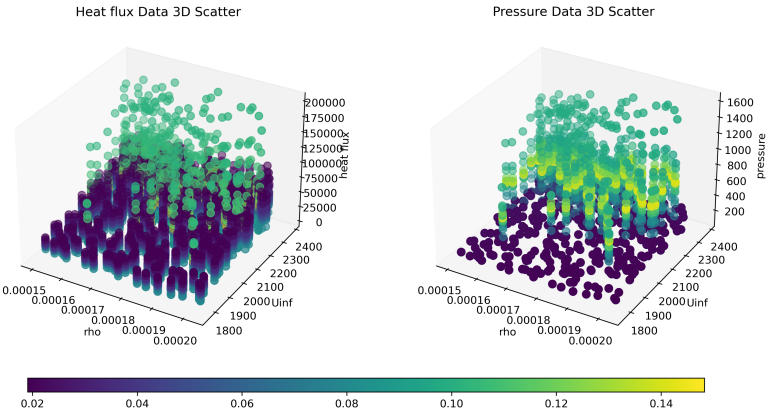


Fig. 4.4: 3D Scatterplot for Run 07 Scenario. The yellow color indicates a greater distance from the cone, while a dimmer hue suggests proximity to the cone.

Pressure values increase with distance from the tip but show a more noticeable gradient compared to the previous dataset. The maximum pressure here is around 1600, which is

lower than in the previous figure. The data points show a clearer division between low-pressure and high-pressure regions, with more pronounced clustering at lower pressures near the tip.The spread of pressure values across density and velocity appears tighter in this figure, suggesting a stronger correlation or a more controlled range of conditions.

4.3. NN Construction. In this section, we present the performance evaluation of different neural network architectures, including Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN), in predicting heat flux and pressure within our dataset. The models were trained using a dataset comprising the predictors (density, velocity, and location) to estimate the target variables (heat flux and pressure).

To ensure robust and reliable model evaluation, the dataset was divided into training and testing subsets. The training set was further split into training and validation subsets to facilitate model tuning and prevent overfitting. The validation data was crucial in monitoring the model's performance during training, allowing for the fine-tuning of hyperparameters and early stopping strategies. This approach ensured that the models generalized well to unseen data, as reflected in the test set performance metrics.

4.3.1. Model Selection. Each of the three NN architectures featured in the present study- MLP, RNN and CNN- were constructed for the two quantities of interest: heat flux and pressure. The results are reported in Table 4.1 and include the coefficient of determination (R^2) and mean squared error (MSE) for both the training and test sets. These metrics provide a comprehensive assessment of each model's predictive accuracy and reliability. The findings reveal the strengths and limitations of each neural network architecture, offering insights into their suitability for the task at hand.

	heat flux			**PRESSURE**		
Model	**Train** R^2	**Test** R^2	**MSE**	**Train** R^2	**Test** R^2	**MSE**
MLP	0.8184	0.6120	0.0798	0.8959	0.8117	0.0723
RNN	0.9011	0.7534	0.0638	0.9982	0.9959	0.0105
CNN	0.9669	0.9205	0.0361	0.9658	0.9356	0.04226

Table 4.1: Model performance metrics for heat flux and pressure using three neural network architectures.

For heat flux, the MLP model achieved a training R^2 of 0.8184 and a test R^2 of 0.6120, with an MSE of 0.0798 on the test set. The RNN model performed better, with a training R^2 of 0.9011 and a test R^2 of 0.7534, along with a lower test MSE of 0.0638. The CNN outperformed both with a training R^2 of 0.9669 and a test R^2 of 0.9205, and an MSE of 0.0361, indicating its superior ability to generalize and accurately predict heat flux. In pressure prediction, the MLP model again demonstrated solid performance with a training R^2 of 0.8959 and a test R^2 of 0.8117, and an MSE of 0.0723. The RNN model, however, excelled with near-perfect results, showing a training R^2 of 0.9982 and a test R^2 of 0.9959, with a very low MSE of 0.0105, making it the most accurate model in this task. The CNN, while still performing well, showed a slightly lower test R^2 of 0.9356 and a higher MSE of 0.4226 compared to its performance on heat flux prediction.

While the RNN and CNN models demonstrated superior accuracy in both tasks, as indicated by their higher R^2 values and lower MSEs, the MLP model was ultimately chosen for further use. This decision was based on the simplicity and ease of implementation associated with MLPs, despite their slightly lower performance metrics. The MLP offers

a balance between performance and computational efficiency, making it a more practical choice, especially when the application context does not demand the highest possible precision. Moreover, the simpler architecture of the MLP makes it easier to train and deploy, requiring less computational power and time. This trade-off between simplicity and performance is often crucial in real-world applications, where the marginal gains in accuracy offered by more complex models may not justify the additional complexity and resource requirements. Therefore, despite the higher accuracy, the MLP was selected as the preferred model due to its straightforward implementation and adequate performance in both heat flux and pressure prediction tasks.

4.4. Model Calibration. The impact of model calibration on the surrogate models is shown in Figures 4.5 and 4.6. The plots are organized into three rows, with the left column depicting the pre-calibration results and the right column presenting the outcomes following calibration.

In the top row, which compares predicted versus actual values, the dashed red line represents the ideal scenario where predictions perfectly match the actual values. Prior to calibration, the data points are scattered significantly below this line, indicating a considerable underestimation by the model, with nearly all predictions falling short of the actual values. Following calibration, the plot on the right shows a marked improvement, as the points are positioned much closer to the dashed red line, indicating that the model's predictions are now more accurately aligned with the actual values. This demonstrates that calibration mitigates the underestimation issue, resulting in predictions that better reflect the true values. Note that the post-calibration results still suffer from model discrepancy, resolving which is reserved for future work. The middle-left plot highlights the uncertainty in the model's predictions before calibration. The blue shaded area represents the uncertainty band around the mean predictions (depicted by the blue line), while the red line indicates the actual values. The plot reveals a consistent underestimation by the model, with the uncertainty band failing to encompass the actual values, particularly around the peak at the 30^{th} location. Additionally, the uncertainty band is relatively narrow, signifying an unjustified level of confidence in predictions that are, in reality, inaccurate. After calibration, as illustrated in the middle-right plot, the model's predictions exhibit significant improvement. Though still underpredicting, the uncertainty band more accurately captures the range of actual values. The band has appropriately widened where necessary, especially around the peak, suggesting that the model now provides a more realistic assessment of its uncertainties. The bottom-left plot shows the discrepancy before calibration, while the bottom-right plot, which depicts the post-calibration results, reveals a significant reduction in errors.

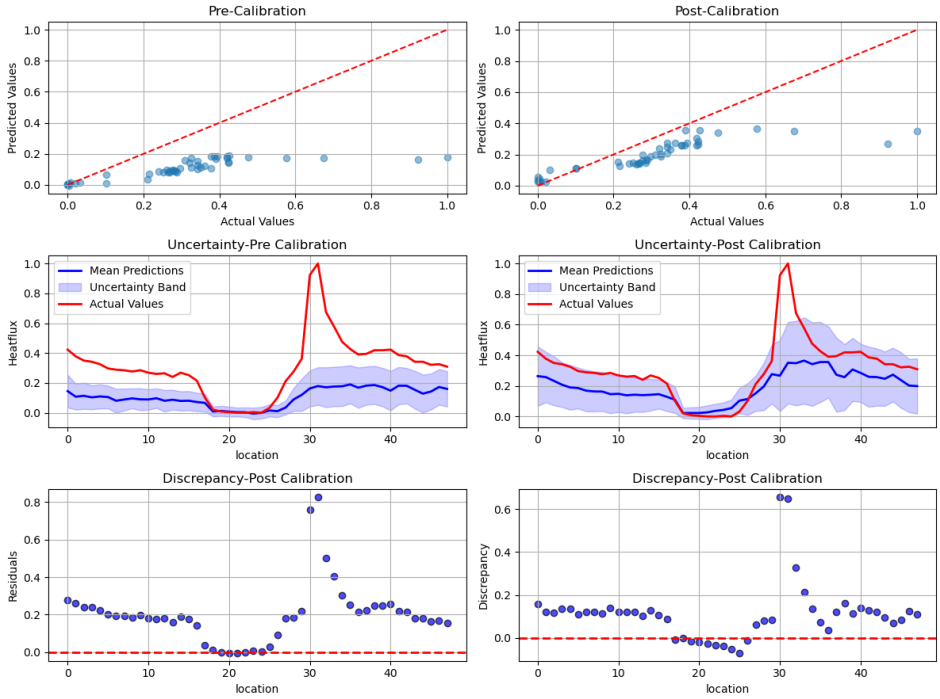


Fig. 4.5: Assessment of MLP surrogate performance before (left) and after (right) calibration for Run 06. These three visualizations capture how the model improved following calibration.

Similar plots for Run 07 are shown in Figure 4.6, in the pre-calibration plot on the left, the data points are noticeably scattered below this line, indicating that the model is significantly underestimating the actual values. After calibration, the plot on the right exhibits a clear improvement: the points are much closer to the dashed red line, suggesting that the model's predictions now more accurately align with the actual values. Calibration has successfully addressed the underestimation issue, enhancing the accuracy of the model's predictions. As with the results for Run 06, some model discrepancy remains and is intended for future studies. The pre-calibration uncertainty, shown in the middle plot on the left, illustrates a narrow uncertainty band as before, indicating that the model is overly confident in its inaccurate predictions. After calibration, as depicted in the middle-right plot, there is again an improvement. In the bottom row, the post-calibration plot reveals a significant reduction in residuals across most locations. This reduction highlights that calibration has effectively minimized errors, resulting in a more accurate and dependable surrogate model. In summary, the post-calibration results in the figures demonstrate that the calibration process has improved the model's predictive accuracy, enhanced the reliability of uncertainty quantification, and reduced discrepancies between predicted and actual values, making the model more robust and reliable for predicting heat flux and pressure.

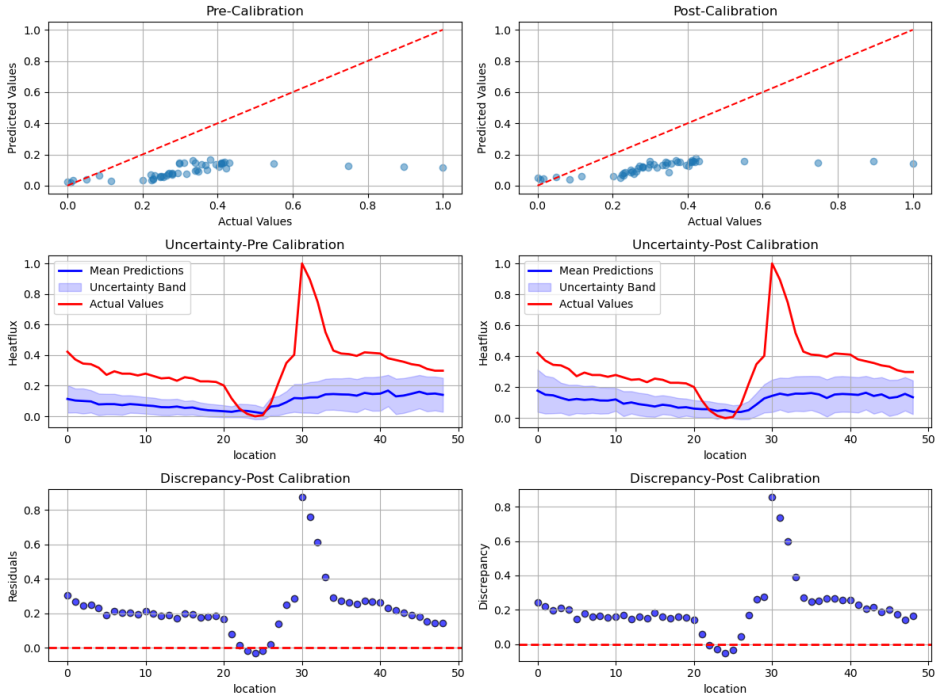


Fig. 4.6: Assessment of MLP surrogate performance before (left) and after (right) calibration for Run 07. These three visualizations capture how the model improved following calibration.

4.5. Discrepancy. The study employed the additive dependency approach as outlined in the framework established by Kennedy and O'Hagan. Despite the efforts dedicated to calibrating the MLP surrogate model, discrepancies persist in comparisons to the experimental data. Figure 4.7 compares the discrepancies between the high-fidelity model (SPARC) and its corresponding surrogate model for heat flux (top row) and pressure (bottom row) in the Run 06 scenario. The left column displays the discrepancies from the high-fidelity model, while the right column shows the discrepancies from the surrogate model. The primary objective of this comparison is to evaluate how well the surrogate model approximates the high-fidelity model.

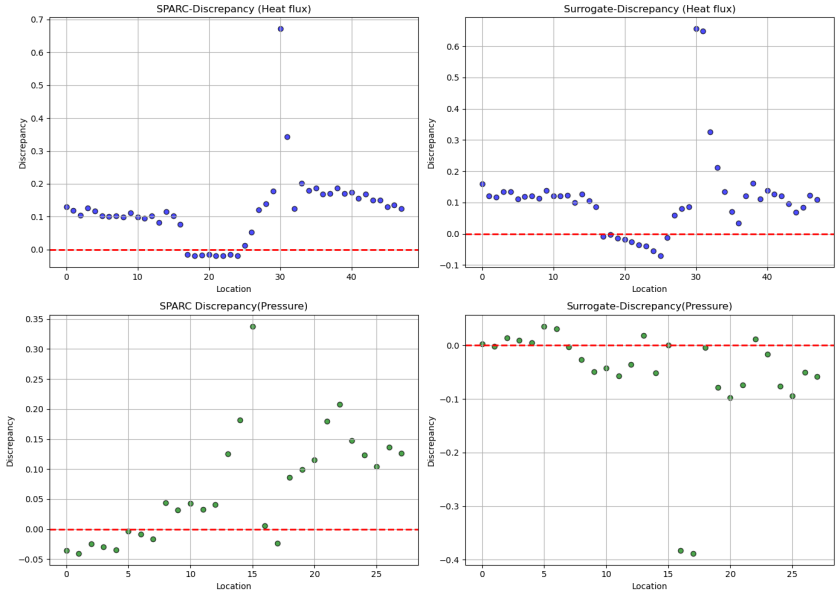


Fig. 4.7: Comparing the discrepancy between SPARC and the MLP surrogate for Run 06. This is to investigate if the MLP can be used as a surrogate to SPARC.

In the top row, the discrepancies for heat flux are plotted against location. The high-fidelity model's discrepancies (left plot) exhibit a range of values, with most points remaining close to the zero line, though there are notable deviations, particularly around the 30^{th} location, where a significant spike is observed. The surrogate model's discrepancies (right plot) similarly show a general alignment with the zero line, with deviations that closely mirror those of the high-fidelity model. The most notable feature is that the surrogate model also captures the spike around the 30^{th} location, though with slight variations in magnitude and shape. The fact that the surrogate model reflects the key patterns and deviations seen in the high-fidelity model, particularly the spike, indicates that the surrogate is a strong approximation for predicting heat flux. The discrepancies between the two models are generally small, suggesting that the surrogate model effectively replicates the behavior of the high-fidelity model across various locations. In the bottom row, the discrepancies for pressure are plotted against location. The high-fidelity model's discrepancies (left plot) are distributed around the zero line, with a few noticeable outliers, particularly between the 10^{th} and 20^{th} locations. These outliers indicate locations where the high-fidelity model detects variations or anomalies in pressure that deviate from the expected values.

Similar plots for Run 07 are shown in Figure 4.8. The surrogate model's discrepancies (right plot) also show a distribution around the zero line but with some differences compared to the high-fidelity model. Notably, the surrogate model captures the overall trend of the discrepancies but exhibits some systematic deviations, particularly in the range from the 5^{th} to the 15^{th} locations, where the surrogate model's discrepancies tend to be negative. This suggests that while the surrogate model approximates the high-fidelity model's behavior, it may slightly underestimate or overestimate pressure at certain locations. However, the general pattern and magnitude of the discrepancies indicate that the surrogate model remains a reasonable approximation for pressure prediction. The comparison between the high-fidelity and surrogate models for both heat flux and pressure indicates that the surrogate model

provides a good approximation of the high-fidelity model. For heat flux, the surrogate model captures the key features and discrepancies observed in the high-fidelity model, particularly the significant spike around the 30^{th} location. For pressure, while there are some systematic deviations, the overall trend and magnitude of discrepancies suggest that the surrogate model reasonably approximates the high-fidelity model's behavior. Overall, the surrogate model effectively replicates the key patterns of the high-fidelity model, making it a suitable and efficient alternative for scenarios where computational resources are limited, or when quick predictions are needed.

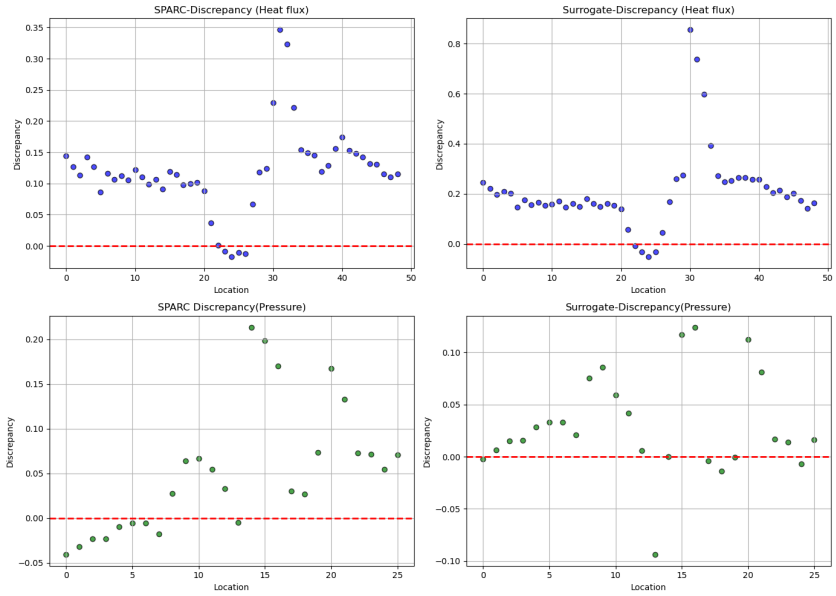


Fig. 4.8: Comparing the discrepancy between SPARC and the MLP surrogate for Run 07. This is to investigate if the MLP can be used as a surrogate to SPARC

5. Discussion and Conclusion. The study presented here builds upon previous research by investigating the relationship between key parameters and their predictive influence on heat flux and pressure in a laminar hypersonic double-cone experiment. This investigation leverages both experimental data from the LENS-I and high-fidelity simulated data generated by the SPARC code. The core of this research lies in the application of advanced neural network architectures—namely, Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN)—to predict heat flux and pressure, followed by an exploration of model calibration and discrepancy analysis.

Through a correlation analysis, the temperature parameters were determined to be unnecessary for the accurate representation of the quantities of interest. The MLP, RNN and CNN were therefore trained over velocity, density, and location. The performance evaluation of the neural network architectures revealed distinct differences in their ability to predict heat flux and pressure. The CNN outperformed other models in predicting heat flux, achieving the highest R^2 and lowest MSE on the test dataset. Similarly, the RNN demonstrated exceptional accuracy in predicting pressure, with near-perfect R^2 values. However, despite these superior metrics, the MLP was ultimately chosen for further application due to its balance between simplicity and adequate performance. The decision to prioritize simplicity

and ease of implementation over marginal gains in accuracy reflects a pragmatic approach to model selection, particularly relevant in real-world applications where computational resources and time are often constrained.

Model calibration further refined the predictive accuracy of the selected models. The calibration process addressed the initial underestimation issues in the surrogate model's predictions. The reduction in residuals and the alignment of uncertainty bands with actual values following calibration underscore the importance of this step in enhancing model reliability. Discrepancy analysis between the high-fidelity and surrogate models reveals that the surrogate model closely approximates the behavior of the high-fidelity model. This finding suggests that the surrogate model is a suitable alternative for scenarios requiring efficient and quick predictions without sacrificing significant accuracy.

The neural network architecture employed in the present study, while effective, do not explicitly consider the correlated nature of the output variables. A multi-output framework could potentially offer an avenue for further improvement in predictive modeling and will be explored in future work.

REFERENCES

[1] S. Bi, S. Prabhu, S. Cogan, and S. Atamturktur, *Uncertainty quantification metrics with varying statistical information in model calibration and validation*, AIAA journal, 55 (2017), pp. 3570–3583.

[2] Y. Gal and Z. Ghahramani, *Dropout as a Bayesian approximation: Representing model uncertainty in deep learning*, in international conference on machine learning, PMLR, 2016, pp. 1050–1059.

[3] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63 (2001), pp. 425–464.

[4] Y. Ling, J. Mullins, and S. Mahadevan, *Selection of model discrepancy priors in Bayesian calibration*, Journal of Computational Physics, 276 (2014), pp. 665–680.

[5] K. A. Maupin and L. P. Swiler, *Model discrepancy calibration across experimental settings*, Reliability Engineering & System Safety, 200 (2020), p. 106818.

[6] I. Park and R. V. Grandhi, *A Bayesian statistical method for quantifying model form uncertainty and two model combination methods*, Reliability Engineering & System Safety, 129 (2014), pp. 46–56.

[7] S. Peitz and M. Dellnitz, *A survey of recent trends in multiobjective optimal control—surrogate models, feedback control and objective reduction*, Mathematical and computational applications, 23 (2018), p. 30.

[8] J. Ray, S. Kieweg, D. Dinzl, B. Carnes, V. G. Weirs, B. Freno, M. Howard, T. Smith, I. Nompelis, and G. V. Candler, *Estimation of inflow uncertainties in laminar hypersonic double-cone experiments*, AIAA journal, 58 (2020), pp. 4461–4474.

[9] C. Soize, *Uncertainty quantification*, Springer, 2017.

[10] R. K. Tripathy and I. Bilionis, *Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification*, Journal of computational physics, 375 (2018), pp. 565–588.

[11] G. G. Wang and S. Shan, *Review of metamodeling techniques in support of engineering design optimization*, in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 4255, 2006, pp. 415–426.

QUANTIFYING ALEATORIC UNCERTAINTY IN OPERATOR LEARNING USING GENERATIVE NETWORKS

JULIO C. PÁEZ* AND RAVI G. PATEL†

Abstract. The standard implementation of deep neural networks yields deterministic behavior after training, where each input always yields the same prediction. This can be problematic when the data used to train the network has high uncertainty. We use generative networks to produce randomized outputs of deep neural networks. Focusing on aleatoric uncertainty, these networks are then applied to regression as well as operator learning to yield a range of predictions rather than one static predictions, all while bypassing the need to compute the posterior distribution required for Bayesian neural networks.

1. Introduction. Operator learning is a recently developed generalization of regression to mappings between functions that promises to drastically replace expensive numerical integration of partial differential equations (PDEs) with fast evaluations of mappings between functional states of a system. However, even in the ideal case where operator learning recovers the training data, there will be epistemic uncertainty in the interpolatory and extrapolatory regimes. A trustworthy model is one that is transparent about its shortcomings. Single-point estimates such as the maximum likelihood estimate (MLE) and maximum a posteriori (MAP) point hide the broad range of fits that reasonably match the training data while presenting vastly different predictions. Trustworthy models include UQ which exposes analysts and designers to the full range of surrogate predictions. Moreover, models that accurately decompose uncertainty into aleatoric and epistemic parts more faithfully capture inherent stochastic dynamics and enable decision makers and algorithms to seek new data to improve the models.

The standard approach to modeling epistemic and aleatoric uncertainty in neural networks is Bayesian inference with heteroscedastic noise. However, computing the posterior is a computationally expensive task. For high dimensional parameter spaces as encountered in neural network models, the posterior is intractable and approximations are necessary. In the limit of large chains with large enough step sizes, Markov Chain Monte Carlo (MCMC) can produce samples of the posterior distribution, but this convergence is slow. For neural networks, the posterior is typically approximated with variational inference (VI). However, VI does not provide convergence guarantees to the true posterior distribution.

Even with the posterior distribution, UQ of new predictions requires the posterior predictive distribution, which involves an integral over the parameter space. Again, for high-dimensional parameter spaces, this computation is intractable, and must be approximated. As neural operators are composed of deep neural networks, they also suffer from the computational limitations of the Bayesian approach to UQ.

Additionally, while Bayesian inference of phenomenological parameters provides useful information about a physical process, Bayesian inference of neural network weights does not. Within a compact domain, there are many sets of neural network weights that yield the same function, within some approximation. For neural networks, the posterior predictive distribution provides the most useful UQ information. Since it requires one to marginalize out the parameter space from the posterior distribution anyway, we attempt to avoid computing posterior distribution and focus on computing a predictive distribution directly. In this work, we investigate an alternative approach to UQ leveraging techniques from generative modeling. Our approach is to construct a generative model to produce model parameters whose actions on data match the data distribution. We demonstrate our method by quan-

*University of Texas Rio Grande Valley, Sandia National Laboratories, julio.paez01@utrgv.edu
†Sandia National Laboratories, rgpatel@sandia.gov

tifying the aleatoric uncertainty from a scalar regression problem and an operator learning problem.

2. The Structure of Deep Neural Networks. The prototypical deep neural network (DNN) is a function composed of other functions, called *layers*, which approximates a function of choice f. Each layer takes an input $\boldsymbol{x}_{i-1}$, performs a linear transformation W_i on $\boldsymbol{x}_{i-1}$, adds a bias vector $\boldsymbol{b}_i$, and evaluates a function σ (called the *activation function*) at the result to get its output $\boldsymbol{x}_i$:

$$\boldsymbol{x}_i = \sigma(W_i \boldsymbol{x}_{i-1} + \boldsymbol{b}_i).$$

If we define

$$L_i(\boldsymbol{x}) \coloneqq \sigma(W_i \boldsymbol{x} + \boldsymbol{b}_i),$$

then a DNN comprised of d layers can be represented by the composition

$$N(\boldsymbol{x}_0; \xi) = L_d\Big(L_{d-1}\Big(\cdots L_2\big(L_1(\boldsymbol{x}_0)\big)\cdots\Big)\Big), \tag{2.1}$$

where $\boldsymbol{x}_0$ represent the initial input of the DNN and ξ is the set of all the weight matrices W_i and biases $\boldsymbol{b}_i$:

$$\xi = \{W_1, W_2, \ldots, W_d, \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_d\}.$$

We also use ξ to denote the set of all the particular elements of each weight matrix and bias vector, also called the *parameters* of the DNN.
The signature aspect of deep neural networks is how these parameters are determined. In general, another function M (called the *objective* or *loss function*) is chosen and the parameters are defined to be the arguments where M reaches its minimum given a training dataset of observations X:

$$\xi \coloneqq \underset{\xi^*}{\operatorname{argmin}}\, M\big(f(\boldsymbol{x}), N(\boldsymbol{x}; \xi^*)\big),$$

where $\boldsymbol{x}$ are the individual observations from X.
Usually, gradient-based optimization is used to obtain approximations of these parameters [2].

2.1. Regression with Deep Neural Networks. The quintessential use case for DNNs is regression: approximating a function given a set of n points $\big\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\big\}_{i=1}^n$.
Typically, the objective function used for regression is the mean squared error (MSE):

$$\xi \coloneqq \arg\min_{\xi^*} \sum_{i=1}^{n} \big\|\boldsymbol{y}_i - N(\boldsymbol{x}_i; \xi^*)\big\|^2.$$

3. Problems with Quantifying Uncertainty in Deep Neural Networks. Deep neural networks can run into problems when trained on data with high uncertainty. For example, consider a dataset of n points $\big\{(x_i, y_i)\big\}_{i=1}^n$ with high aleatoric uncertainty as in Figure 3.1. After a DNN is trained on such a dataset, the output of the network at any given test point x^* will always produce the same output y^*. This produces a single line over the testing data set (Figure 3.2). The determinism of the DNN structure after training is a problem as we cannot quantify the uncertainty of the output. Furthermore, high aleatoric

FIG. 2.1. *A deep neural network trained using the plotted points.*

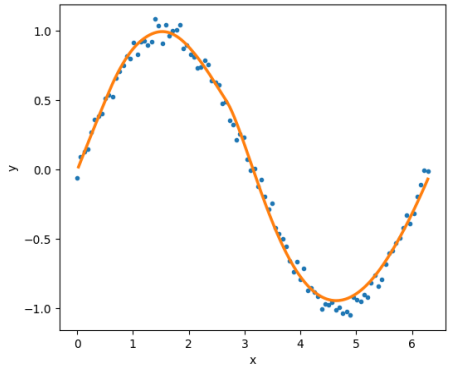
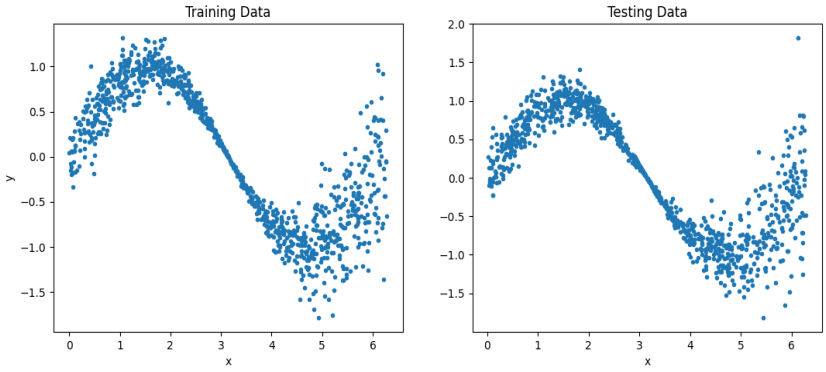


FIG. 3.1. *Training and testing datasets with high aleatoric uncertainity at both ends.*



uncertainty can be caused by a process's inherent stochastic behavior, behavior which would ideally be replicated in the model.

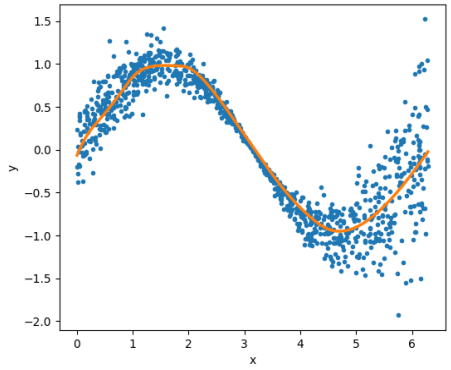
A way to combat this problem is to use Bayesian neural networks [1]. However, implementing a Bayesian neural networks comes with its own problems. Namely, the reliance on calculating the posterior probability distribution. This distribution includes an integral over the parameter space, a space which is high-dimensional for all but the most trivial neural networks. This integral is thus intractable and computationally expensive to estimate. Hence, we will use a different approach to introduce stochasticity to DNNs.

4. Generative Networks. A generative network (GN) has an architecture similar to a DNN, but whose input is a sample $\boldsymbol{z}$ from a probability distribution $P_{\boldsymbol{z}}$ over the space $\mathcal{Z}$. Because the input of the GN is random, the outputs are randomized in turn. Thus, we can use a GN to generate randomized parameters of a DNN, which leads to randomized outputs even when receiving the same input.

Say we have a deep neural network N as described in Section 2 and a generative network G with hyperparameters θ. Then, to produce the output $\boldsymbol{y}^*$ of the network N, we first sample $P_{\boldsymbol{\psi}}$ to get $\boldsymbol{\psi}$, input $\boldsymbol{\psi}$ into the generative network $G(\,\cdot\,;\theta)$, feed the parameters ξ produced by $G(\boldsymbol{\psi};\theta)$ into N, and finally input $\boldsymbol{x}^*$ into $N(\,\cdot\,;\xi)$:

$$\begin{array}{ccccccc} & & \boldsymbol{x}^* & \longrightarrow & N(\boldsymbol{x}^*;\xi) & \longrightarrow & \boldsymbol{y}^* \\ & & & & \uparrow & & \\ \boldsymbol{\psi} \sim P_{\boldsymbol{\psi}} & \longrightarrow & G(\boldsymbol{\psi};\theta) & \longrightarrow & \xi & & \end{array}$$

FIG. 3.2. *A deep neural network trained on the training data and acting on the testing data.*



By repeating this process multiple times for each input, we obtain a range of predictions rather than being limited to a single one.

Of course, we cannot expect fully random ξ parameters to yield reasonable predictions. Hence, we need to tune the hyperparameters θ in G. We will do so using scoring rule minimization [3].

A *scoring rule* S is a function over two distributions P and Q which provides a sense of distance between the two distributions:

$$S(P, Q) \coloneqq \mathbb{E}_{\boldsymbol{X}\sim Q} S(P, \boldsymbol{Z}).$$

A scoring rule S is *strictly proper* relative to a set of distributions $\mathcal{P}$ if $P = Q$ is the unique minimum of S in the set $\mathcal{P}$. In the context of generative networks, we will compare the concatenation of the model input and outputs to the input/output data pairs, $z = [x, y]^T$.

From now on, we will use the energy score, with unbiased estimate

$$\hat{S}_E^{(\beta)}(P, \boldsymbol{z}) = \frac{2}{m}\sum_{j=1}^{m} \|\tilde{\boldsymbol{z}}_j - \boldsymbol{z}\|_2^\beta - \frac{1}{m(m-1)}\sum_{j,k=1}^{m} \|\tilde{\boldsymbol{z}}_j - \tilde{\boldsymbol{z}}_k\|_2^\beta, \quad \tilde{\boldsymbol{z}}_j \sim P,$$

and let the input $\boldsymbol{\psi}$ to the generative network have each of its elements be distributed according to the random normal distribution, $\mathcal{N}(0, 1)$.

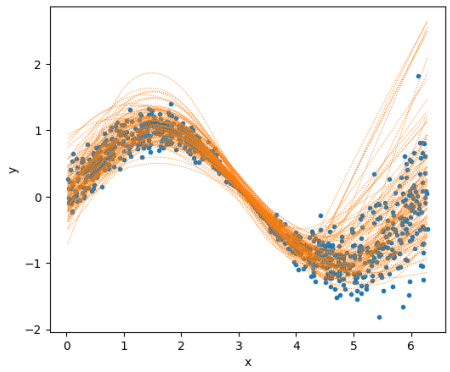
5. Aleatoric Uncertainty Quantification in Regression. Consider the data described in Section 3. Using the same training data of size 1000, with $m = 10$ draws, 100 batches of size 10, and 125 epochs, we can fit the DNN and GN pair to the data and arrive at a range of outputs. The same size was used for the DNN in this case as in Figure 3.2, as well as the same training data, batches, and epochs. We train a generative network to produce the weights for a neural network fit to the data using the scoring rule to match the predictions to the data in distribution. Our results are shown in Figure 5.1.

6. Operator Learning. Another task DNNs can perform is operator learning: Instead of having input points/vectors $\boldsymbol{x}_i$, corresponding output points/vectors $\boldsymbol{y}_i$ and trying to learn a function $f(\boldsymbol{x}) = \boldsymbol{y}$, we have input *functions* $\boldsymbol{u}_i$, corresponding output *functions* $\boldsymbol{v}_i$ and try to learn an *operator* $L[\boldsymbol{u}] = \boldsymbol{v}$.

While this can be done in various ways, we will use the parameterization described in [4]: given an input function $\boldsymbol{u}^*$, the output $\boldsymbol{v}^*$ is given by

$$\boldsymbol{v}^* = \mathcal{F}^{-1}\Big[g(\kappa; \xi_1) \cdot \mathcal{F}\big[h(\boldsymbol{u}^*; \xi_2)\big]\Big],$$

FIG. 5.1. *A deep neural network aided by a generative network acting on the same data as before.*



where $g(\,\cdot\,;\xi_1)$ is a DNN with parameters ξ_1, $h(\,\cdot\,;\xi_2)$ is another DNN with parameters ξ_2, κ are the wavenumbers, and $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform and the inverse Fourier transform, respectively.
A generative network G can be introduce in this process to add stochasticity to the output, this time generating weights for both $g(\,\cdot\,;\xi_1)$ and $h(\,\cdot\,;\xi_2)$:

$$\begin{array}{ccccccccc} & & \kappa & \longrightarrow & g(\kappa;\xi_1) & \xrightarrow{\times} & g(\kappa;\xi_1)\cdot\mathcal{F}\big[h(\boldsymbol{u}^*;\xi_2)\big] & \xrightarrow{\mathcal{F}^{-1}} & \boldsymbol{v}^* \\ & & & & \uparrow & & \uparrow\mathcal{F} & & \\ \boldsymbol{\psi}\sim\mathcal{N}(0,1) & \longrightarrow & G(\boldsymbol{\psi};\theta) & \longrightarrow & \xi_1,\xi_2 & \longrightarrow & h(\boldsymbol{u}^*;\xi_2) & & \\ & & & & & & \uparrow & & \\ & & & & & & \boldsymbol{u}^* & & \end{array}$$

We will now quantify uncertainty in the case of operator learning.

7. Aleatoric Uncertainty Quantification in Operator Learning. Consider a dataset of pairs of functions $\{(\boldsymbol{u}_i,\boldsymbol{v}_i)\}_{i=1}^n$ over the domain $(0,1)$. We want to learn the operator L which maps the $\boldsymbol{u}$ functions to their corresponding $\boldsymbol{v}$ functions (Figure 7.1). As in the scalar regression example above, we train a generative network to produce the weights of a neural operator using the scoring rule such that predictions from the neural operator match the data in distribution. Figure 7.2 shows that the predictions from the neural operator match held out test data.

8. Conclusion. This work demonstrates a novel, ensemble approach to UQ in operator learning. We demonstrate with a synthetic benchmark that our approach provides intuitive measures of aleatoric uncertainty. Future work will focus on quantifying epistemic uncertainty in operator learning

REFERENCES

[1] D. M. BLEI, A. KUCUKELBIR, AND J. D. MCAULIFFE, *Variational Inference: A Review for Statisticians*, 2018.
[2] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, 2017.
[3] L. PACCHIARDI AND R. DUTTA, *Likelihood-Free Inference with Generative Neural Networks via Scoring Rule Minimization*, 2022.
[4] R. G. PATEL, N. A. TRASK, M. A. WOOD, AND E. C. CYR, *A physics-informed operator regression framework for extracting data-driven continuum models*, Computer Methods in Applied Mechanics, 373 (2021).

FIG. 7.1. *A few samples from the training dataset. Top row shows the input functions and bottom row shows the output functions.*

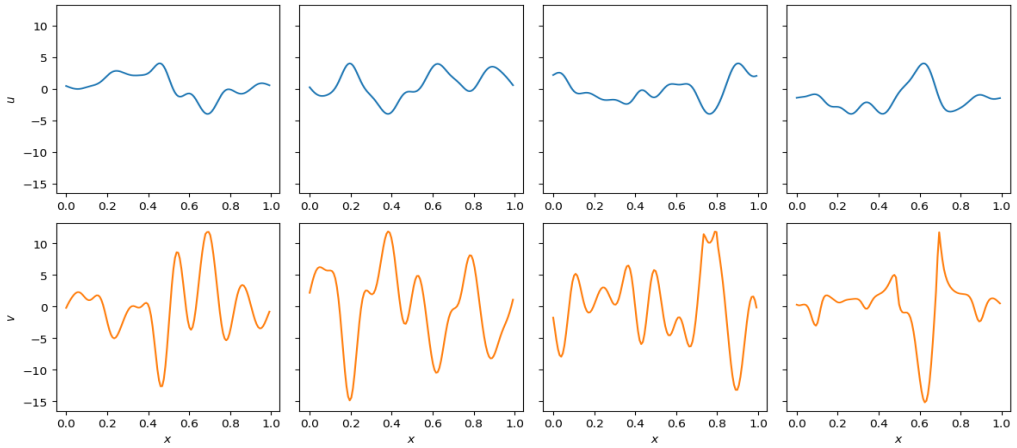
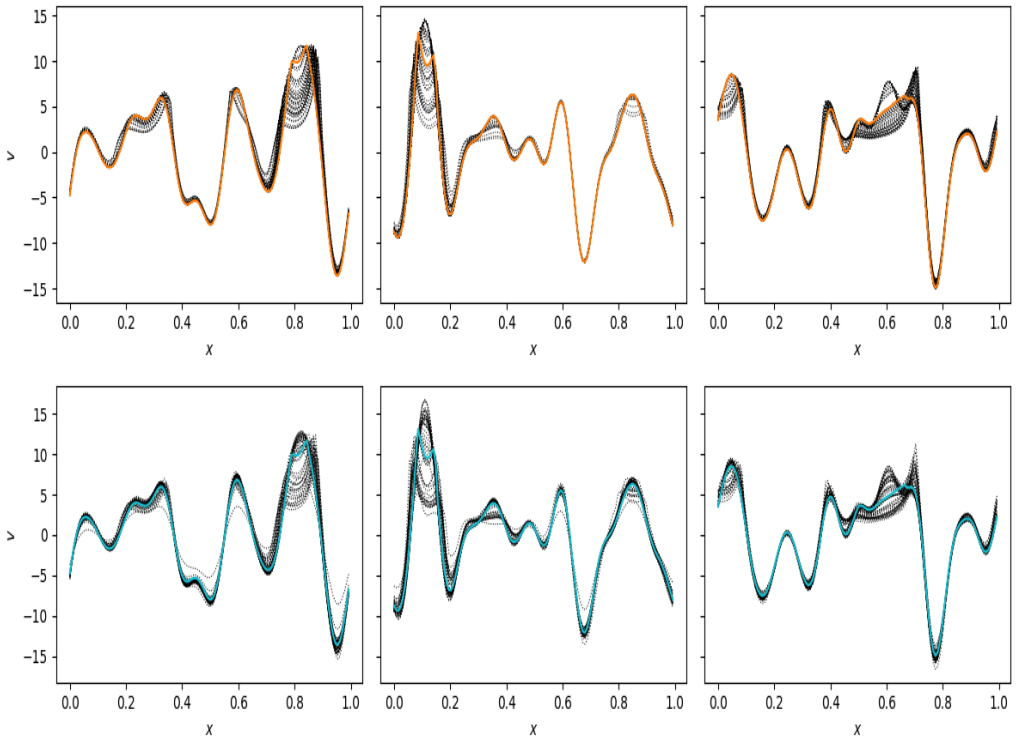


FIG. 7.2. *The top row shows the true operator acting on some samples from the testing data set. The bottom row shows the predictions from the neural operator with weights from the trained generator network*



COUPLED DEEP NEURAL OPERATORS AS A SURROGATE MODEL FOR ICE-SHEET DYNAMICS

DARIO RODRIGUEZ * AND MAURO PEREGO †

Abstract.
Sea level is expected to dramatically increase in the next decades to centuries with severe impact on coastal infrastructure and population. Therefore, sea-level rise projections are critical to inform policy making and adopt strategies to mitigate these changes. However, a large uncertainty in these predictions is due to uncertainty in ice-sheet modeling. Traditionally, ice-sheet dynamics have been modeled using classical discretization methods such as the finite element method (FEM). Nonetheless, FEM-based models tend to be computationally expensive and, therefore, not ideal in the context of uncertainty quantification where a large number of evaluations are required. We hence propose machine-learning-based surrogates as a more efficient alternative to conventional numerical models. Our work builds on previous research where a deep neural operator, the so-called Deep Operator Network (DeepONet), was employed to compute the ice flow velocity while the ice thickness evolution was solved with FEM, generating a hybrid *FEM*-DeepONet ice-sheet model. In the present work, we aim at predicting the evolution of the ice thickness by means of a DeepONet surrogate that maps the ice thickness and parameters at time t to the ice thickness at time $t + \Delta t$, in order to create a fully neural-operator based ice-sheet model. In particular, for the DeepONet that maps the temporal evolution of the ice thickness, two neural operator approaches are considered: i) a *vanilla* architecture, where the operator features both a *branch* and a *trunk* network and ii) a *SVD*-based operator, where the *trunk* net is substituted with a set of basis functions precomputed from the dataset. Moreover, the composition of this neural operator with itself during training is assessed with the main goal of increasing the prediction accuracy and decreasing the training time. The outcomes demonstrate that a four-step composition of the thickness neural operator is ideal in order to obtain highly accurate results. After both the velocity and thickness neural operators are trained independently, they are coupled together to fully characterize the ice-sheet dynamics. Preliminary results demonstrate that the proposed framework is appropriate to handle high-dimensional problems and hence, the model is suitable to conduct uncertainty quantification studies. A brief summary about the hyperparameter selection to find the ideal neural network architectures and configurations is presented. In this work, the Humboldt glacier in Greenland was selected as case study.

1. Introduction. Ice-sheet modeling plays an important role in climate modeling because it is crucial to make probabilistic projections of sea-level rise, an issue that has become a global concern due to the potential catastrophic effects associated with it. A timely and accurate prediction of this phenomenon might help governmental agencies and stakeholders to anticipate the consequences of sea-level rise and thus to inform mitigation policies [1, 2].

To first approximation, the dynamics of ice sheets is governed by Stokes-like flow equations, consisting of a set of partial differential equations (PDE) that is often solved numerically using finite element methods [3, 4, 5]. However, one of the biggest challenges in characterizing ice sheets' physics is related to the inherent presence of high-dimensional uncertainty in the models and observations [6, 1]. As an example, the basal friction field (β), a parameter that describes the sliding conditions at the ice bed and one of the biggest control on ice-flow, can be estimated, together with its probability distribution, by solving a Bayesian inference problem assimilating observations of the surface ice velocity [7, 8]. The uncertainty on quantities of interests such as the total ice-sheet mass loss over time (a proxy of its contribution to sea-level rise) can then be computed with a Monte Carlo method by drawing samples from the parameter distribution and by running, for each sample, the ice-sheet model forward in time. Unfortunately, the cost of conventional PDE-based models and the large number of runs required, makes this approach intractable for realistic ice-sheet

*Department of Aerospace Engineering, University of Illinois Urbana Champaign, darior2@illinois.edu
†Department of Scientific Machine Learning, Sandia National Laboratories, mperego@sandia.gov

problems.

The advent of automatic differentiation (AD) [9] and high-performance computing (HPC) [10] have boosted the embrace of machine learning (ML) as a scientific tool to develop accurate surrogates for modeling such complex problems in a faster manner [11, 12]. In particular, *neural operators* are a type of deep learning models that learn the operator mapping between infinite dimensional function spaces, which allow these ML models to predict relationships between continuous functions [13, 14, 15, 16]. Specifically, one of the most well-known *neural operator* architectures are the deep operator networks (DeepONets) [14], which are a subclass of ML-based surrogates that have proven to work well within a wide range of applications such as gas dynamics [17], hypersonics [18], material science [19] and ice-sheet [20, 21] modeling. In its *vanilla* version [14], a DeepONet relies on uncovering a projection to approximate an operator by using two neural networks, referred to as *branch* and *trunk*. The bases of this projection, which are constructed by the *trunk* network, are assumed of be non-linear functions of the operator's independent variables (time and spatial coordinates), while the coefficients are generated as non-linear functions of the operator's input fields evaluated at a set of discrete sensor points (time and space) by the *branch* network. Then, the DeepONet automatically discovers the appropriate projection by performing the dot product between the output layers from each network, which can be seen as a coefficient-to-input and basis-to-independent-variable mapping [22].

Several works have demonstrated the feasibility of neural operators as surrogates to approximate observation data from large-scale spatio-temporal systems by composing the operator with itself during the training process [23, 24]. This is accomplished by employing a residual network (ResNet) [25] approach to compose the neural operator in a recursive fashion as illustrated in Figure 4.1 [26]. Note that the recurrence is enforced blockwise on the ResNet block, which is itself a deep neural operator. The advantage of this specific architecture stands on that it is capable of approximating unknown dynamical systems using only state variable data, which could be coarsely distributed in time [26].

Nonetheless, the high dimensionality of the training data pose a significant challenge due to the considerable number of trainable parameters to optimize. Therefore, since DeepONets are considered projection-based methods [22], one of the alternatives is to precompute a set of basis functions by means of a singular value decomposition (SVD) of the training data. In other words, this approach aims to replace the conventional *trunk* network of a DeepONet with a set of SVD-generated basis functions. Thus, a single neural network (the *branch* net) is used to learn the coefficients, while the SVD-generated basis is employed instead of the *trunk* network [27, 28].

The overall goal of this work is to generate accurate surrogate models suitable for conducting uncertainty quantification and addressing inverse problems with less computational burden compared to the *FEM* approach. Previous research was focused on the implementation of a hybrid DeepONet-*FEM* approach [20], where a neural operator was trained to map the basal friction field and the ice thickness to the depth-averaged ice velocity. Complementary, the present work aims to substitute the remaining *FEM*-based discretization from the hybrid approach with a DeepONet that maps the ice thickness and parameters at time t to the ice thickness at time $t+\Delta t$, in order to generate a framework of coupled neural operators that fully predicts the ice-sheet dynamics.

In this sense, in section 2, the mathematical models that govern the ice-sheet dynamics are described. Next, in section 3, the numerical approach and model assumptions to generate the training and testing dataset are introduced. Later, in section 4 two different

neural operators (*vanilla* and *SVD*-based) to predict the time-evolution of ice thickness are introduced. This section concludes with a description of how the proposed neural operator is coupled with a previously trained DeepONet to generate a framework of coupled neural operator as a surrogate to assess ice-sheet dynamics. Then, in section 5, a summary of the methods to preprocess the data, the neural networks architectures, as well as the hyperparameter selection, is presented. Afterwards, the loss values and the computational cost for training the operators is compared and discussed. Furthermore, a full prediction of the ice-sheet dynamics obtained by using the two approaches of coupled neural operators is presented and compared with the *FEM*-generated outcomes. Finally, a brief summary and future work research are highlighted in section 8.

2. Ice-sheet models. Figure 2.1 illustrates a cartoon of a vertical section of an ice sheet along the direction of ice flow, where the horizontal coordinates are denoted with x and y, while z represent the vertical coordinates ($z = 0$ is defined as the sea level). The ice domain, at time t, can be approximated as a vertically extruded domain Ω, as shown in Equation 2.1:

$$\Omega := \{(x, y, z) \text{ s.t. } (x, y) \in \Sigma, \text{and } l(x, y, t) < z < s(x, y, t)\} \tag{2.1}$$

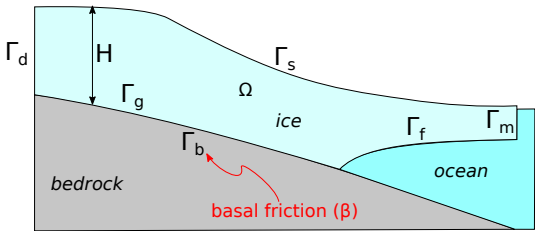


FIG. 2.1. *Scheme of an ice sheet*

where, $\Sigma \subset \mathbb{R}^2$ is the horizontal extension of the ice, $\Gamma_l(t) := \{(x, y, z)\}$ s.t. $z = l(x, y, t)$, $(x, y) \in \Sigma$ represents the lower surface of the ice at time t, and $\Gamma_s(t) := \{(x, y, z)\}$ s.t. $z = s(x, y, t)$, $(x, y) \in \Sigma$ denotes the upper surface of the ice and hence, the ice thickness is obtained by $H(x, y, t) = s(x, y, t) - l(x, y, t)$. Note that the lower surface of the ice is sectioned between grounded (Γ_g) and floating (Γ_f) areas, which determines the boundary conditions of the problem [20].

The ice thickness $H(x, y, t)$ is modeled according to the conservation law shown in Equation 2.2,

$$\partial_t H + \nabla \cdot (\bar{\mathbf{u}} H) = f_H \tag{2.2}$$

where $\bar{\mathbf{u}} := \frac{1}{H} \int_l^s \mathbf{u} dz$ is the depth-integrated velocity and f_H is a forcing term accounting for accumulations (e.g. snow precipitations) and melting in both upper and lower ice surfaces [29].

Since ice sheets behave as a shear thinning fluid, non linear Stokes models, which are described by the set of PDEs shown in Equations 2.3 and 2.4 [29] are often used to model their dynamics. Here, $\mathbf{u} = (u, v, w)$ is the ice velocity vector, σ is the stress tensor, ρ is the ice density and g is the gravity.

$$-\nabla \cdot \sigma = \rho g \tag{2.3}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.4}$$

The Stokes equation is accompanied by the following boundary conditions as shown in Figure 2.1:

$\sigma\mathbf{n} = \mathbf{0}$	on Γ_s	stress free, atmospheric pressure neglected
$\sigma\mathbf{n} = \rho_w g\ min(z, 0)\mathbf{n}$	on Γ_m	boundary condition at the ice margin
$\mathbf{u} = \mathbf{u_d}$	on Γ_d	Dirichlet condition at the internal boundary
$\mathbf{u} \cdot \mathbf{n} = 0,\ (\sigma\mathbf{n})_{\|\|} = \beta\mathbf{u}_{\|\|}$	on Γ_g	impenetrability plus sliding condition
$\sigma\mathbf{n} = \rho_w g z\mathbf{n}$	on Γ_f	back pressure from ocean under ice shelves

Here, $\beta(x, y)$ is the sliding coefficient, ρ_w is the density of the ocean water, $\mathbf{n}$ the unit outward-pointing normal to the boundary and $\mathbf{u}_{||}$ is the component of the velocity $\mathbf{u}$ tangential to the bed. The boundary condition at the margin includes the ocean back-pressure term, when the margin is partially submerged ($z < 0$). For terrestrial margin, $z > 0$, the term becomes a stress-free condition.

3. Computational models. As in previous work [20], the shallow shelf approximation (SSA) [30], a simplification of the Stokes equations which is less expensive to evaluate, was adopted to model the ice-sheet dynamics and to generate an ensemble of data to train the neural operators. In particular, in the SSA model, it is assumed that the velocity is uniform in z (i.e. $\mathbf{u} = \bar{\mathbf{u}}$) and hence, the problem simplifies to a two-dimensional PDE in Σ as shown in Equation 3.1, where the tensor $\hat{\mathbf{D}}$ is associated to the stress tensor σ. See reference [20] for a further explanation of this model and its simplified boundary conditions.

$$-\nabla \cdot \left(2\mu H\hat{\mathbf{D}}\right) + \beta\bar{\mathbf{u}} = -\rho g H \nabla s \tag{3.1}$$

A significant challenge of modeling ice-sheet dynamics is related to the high-dimensional uncertainty. In this work we focus on the uncertainty in the basal friction field (β), which is an unknown infinite dimensional field which greatly affects the ice flow. A Bayesian inference approach can be used to estimate the distribution of β; see, e.g. [7, 8]. However, in this work, a simplified approach is adopted, where β is assumed to have a log-normal distribution with a square covariance k_l, as shown in Equation 3.2

$$\log(\beta) \sim \mathcal{GP}(\log(\bar{\beta}), k_l),\ \text{and}\ k_l(\mathbf{x_1}, \mathbf{x_2}) = a\ \exp\left(-\frac{|\mathbf{x_1} - \mathbf{x_2}|^2}{2l^2}\right) \tag{3.2}$$

Here, $\bar{\beta}$ is the optimal basal friction parameter obtained by solving a PDE-constrained optimization problem [31] to minimize the mismatch between the computed ice velocity and the observed surface ice velocities. The selection of scaling parameters a and l are described in ref. [20].

The finite element discretization of the glacier (based on the SSA approximation) was implemented in FEniCS and solved using PETSc [20]. The samples for β were drawn from the Gaussian process (Equation 3.2). For each β field sample, the *FEM* model was run forward in time to predict both the ice depth-averaged velocity vectors ($\bar{\mathbf{u}}$) and ice thickness (H) to generate the ensemble of data to train the neural operators.

4. Neural operators. The aim of this work is to generate a fully-coupled ensemble of neural operators as a surrogate to predict the ice-sheet dynamics subject to initial conditions. Previous research was focused on the implementation of a hybrid DeepONet-*FEM* approach [20], where a neural operator $\mathcal{G}$ was trained to map the basal friction field (β) and the ice thickness (H) into the depth-averaged velocity vector ($\bar{\mathbf{u}}$). Hence, the operator G, in combination with the *FEM*-based discretization of the ice thickness H over time (Equation 2.2) constituted the *hybrid ice-flow model*. The training, calibration and validation of the operator $\mathcal{G}$ was deeply explored and discussed in ref. [20] and it will not be further addressed in this work. Instead, it is assumed that the operator $\mathcal{G}$ was already trained and it will be directly incorporated as a component of the coupled neural operator surrogate that is proposed in this work.

4.1. Ice thickness neural operator. The emphasis of this work was placed on developing the neural operator $\mathcal{F}$ that substitutes the conventional *FEM* discretization of the ice thickness H over time (Equation 2.2). In this context, the operator $\mathcal{F}$ maps the depth-averaged velocity vector $\bar{\mathbf{u}}$ and ice thickness H states at time $t = n$ to the thickness state at next time step $t + \Delta t = n + 1$ (Δt is assumed as one year), i.e. $H^{n+1} = \mathcal{F}\left(H^n, \bar{\mathbf{u}^n}\right)$, as illustrated in Figure 4.2. Note that the term $\bar{\mathbf{u}}$ depends implicitly on the basal friction field β and hence, the data pairs (shown in Equation 4.1) to train the operator $\mathcal{F}$ were obtained from all the samples of β fields drawn from the distribution 3.2

$$\{(\bar{\mathbf{u}}_i^n, H_i^n), H_i^{n+1}\} \quad n = 0, 1, \ldots, N-1, \quad i = 1, 2, \ldots, Q \tag{4.1}$$

where N is the set of discrete time steps considered and Q is the total number of β samples. Note that the operator $\mathcal{F}$ is a p-time composition of itself where, as an example, p corresponds to 3 on the diagram shown in Figure 4.1.

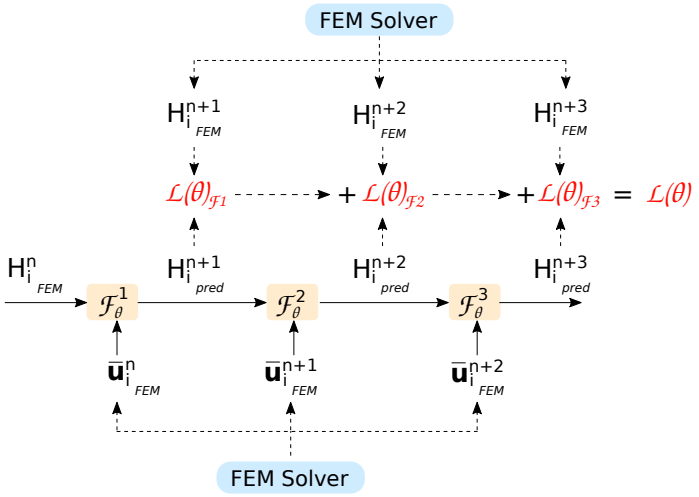


FIG. 4.1. *Composition of neural operator* $\mathcal{F}$

Thus, the training loss function involves summing over the mean squared error in the predictions for each data pair $\{(\bar{\mathbf{u}}_i^n, H_i^n), H_i^{n+1}\}$. Moreover, we included an additional sum over P future time steps, where we compose the $\mathcal{F}$ operator with itself P times. In particular, we employ the loss function $\mathcal{L}_{\theta,\lambda}$ shown in Equation 4.2

$$\mathcal{L}_{(\theta,\lambda)} = \frac{1}{NQ\bar{P}(n)} \sum_{i=1}^{Q} \sum_{n=0}^{N-1} \sum_{\mathbf{x}\in\mathcal{X}} \sum_{p=1}^{\bar{P}(n)} w(\mathbf{x}) \left\| H_i^{n+p}(\mathbf{x}) - \mathcal{F}_\theta^{[p]} \left(H_i^n(\mathbf{x}), \{\bar{\mathbf{u}}_i(\mathbf{x})\}_{j=n}^{n+p-1} \right) \right\|^2 \quad (4.2)$$

where $\mathcal{F}_\theta^{[p]} \left(H_i^n(\mathbf{x}), \{\bar{\mathbf{u}}_i(\mathbf{x})\}_{j=n}^{n+p-1} \right)$ denotes the composition of $\mathcal{F}$ with itself p times, i.e., $\mathcal{F}_\theta^{[p]} \left(H^n(\mathbf{x}), \{\bar{\mathbf{u}}(\mathbf{x})\}_{j=n}^{n+p-1} \right) = \mathcal{F}_\theta(\cdot, \bar{\mathbf{u}}(\mathbf{x})_{n+p-1}) \circ \cdots \circ \mathcal{F}_\theta(\cdot, \bar{\mathbf{u}}(\mathbf{x})_n)(H^n)$ (the i subscript is omitted for readability). Note that the interior sum from $p = 1, \cdots, \bar{P}(n)$ is truncated at $\bar{P}(n) = min(P, N - n)$ to avoid predicting time steps outside the training dataset [23].

Furthermore, $w(\mathbf{x})$ represent an array of spatially-dependent penalizing self-adaptive weights that can lead to a better generalization in presence of localized features [32]. As in previous research [20], we set $w(\mathbf{x}) = m(\lambda(\mathbf{x}))$, where $\boldsymbol{\lambda} := \{\lambda(\mathbf{x})\}$ are the trainable self-adaptive weight parameters dependent on locations $\mathbf{x} \in \mathcal{X}$ and $m(\lambda) = \lambda^d$ is a non-negative monotonically-increasing polynomial mask function with $d = 4$. Note that the loss function $\mathcal{L}(\theta, \lambda)$ is simultaneously minimized with respect to the network hyperparameters θ but maximized with respect to the self-adaptive weights λ i.e., $\min_\theta \max_\lambda \mathcal{L}(\theta, \lambda)$. Finally, $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M\}$ is the finite set of sensor points (spatial coordinates).

From Figure 4.1, it can be noticed that $\mathcal{F}$ acts as a multi-step recurrent residual deep neural network (ResNet) [26], where for each p composition, the *branch* net takes as input the dot product between H and the corresponding spatial components of the vector $\bar{\mathbf{u}}$ (to mimic the argument of the divergence operator from Equation 2.2).

Additionally, the operator $\mathcal{F}$ also consists of either i) a *trunk* network (*vanilla* operator) or ii) a precomputed set of basis functions that substitute the *trunk* net (*SVD*-based operator) as shown in Figure 4.2. In the former approach, the *trunk* net, which is feed-forward neural network, takes as input the sensor points (spatial coordinates) $\mathbf{x} \in \mathcal{X}$ [14] and yields a vector t_k from its output layer. Then, this latter is combined via dot product with the *branch* net's output layer b_k and the resulting value is added up to the ice thickness at the current time step as shown in Equations 4.3 and 4.4. An important remark is that for the first composition, the initial H corresponds to the initial condition, i.e. the ice thickness at time step $t = n$ but for subsequent compositions, the predicted ice thickness H_{pred}, together with the depth-averaged velocity from the *FEM* model ($\bar{\mathbf{u}}_{FEM}$), are combined together via dot product and used as input for the *branch* net of subsequent compositions of the neural operator. Finally, note that for each p-composition, the loss value is computed via the mean square error between $H_{i,FEM}^{n+p}$ and $H_{i,pred}^{n+p}$. In this approach both *branch* and *trunk* networks' hyperparameters are optimized.

$$H_{i,pred}^{n+1} = H_i^0 + \sum_{k=0}^{K} b_k \left(H_{i,FEM}^n, \bar{\mathbf{u}}_{i,FEM}^n \right) t_k(\mathbf{x}) \quad (4.3)$$

$$H_{i,pred}^{n+p+1} = H_{i,pred}^{n+p} + \sum_{k=0}^{K} b_k \left(H_{i,pred}^n, \bar{\mathbf{u}}_{i,FEM}^n \right) t_k(\mathbf{x}), \quad p = 1, \cdots, \bar{P}(n) \quad (4.4)$$

On the other hand, for the *SVD*-based operator, the set of basis functions (also called as t_k in Figure 4.2 for simplicity) are precomputed from the matrix $H_{n,i}^m \in \mathbb{R}_M^{N\times Q}$ which contains the ice thickness data, where the M columns are the number of grid nodes and the product $N \times Q$ rows, that we call S, represents a stacked combination of the time steps

n and β samples considered. Therefore, the *SVD* of the matrix $H^m_{n,i}$ is used to generate a lower-dimensional linear subspace [27] given by the approximation of $H^m_{n,i}$ as the product of the truncated matrices $\tilde{U}\tilde{\Sigma}\tilde{V}^T$, where $\tilde{\Sigma} \in \mathbb{R}^{r\times r}$ contains the largest r singular values associated to the truncated singular vectors $\tilde{U} \in \mathbb{R}^{S\times r}$ and $\tilde{V}^T \in \mathbb{R}^{r\times M}$. Hence, the right singular vector $\tilde{V} \in \mathbb{R}^{r\times M}$ was used as the set of basis functions [22], where r was set heuristically to retain the most amount of thickness data information. The output layer vector b_k from the *branch* net is then combined with the set of basis functions t_k for each p-composition following the same approach as described for the *vanilla* case.

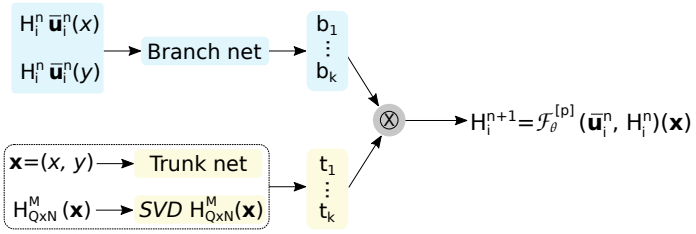


FIG. 4.2. *Architecture for the neural operator* $\mathcal{F}$

4.2. Coupled neural operators. Once the two operators, $\mathcal{G}$ (that predicts the depth-averaged ice velocity) and $\mathcal{F}$ (that maps the temporal evolution of the ice thickness), were trained independently, they were coupled together in such a way that for a given β_i field distribution and initial ice thickness H^0_i, the first operator computes the deep-averaged velocity vector $\bar{\mathbf{u}}^0_i$ at time $t = 0$ and then, this latter value, in conjunction with the initial ice thickness H^0_i, are fed into the $\mathcal{F}$ operator so it predicts the ice thickness at the next time step H^1_i. This cycle is repeated until the states $\bar{\mathbf{u}}$ and H are completely determined for a desired time span $t = N$. Although the coupled model was used to predict the dynamics of aforementioned states (H and $\bar{\mathbf{u}}$), the ultimate goal is to predict the ice mass added to the ocean which is associated to the ice-sheet mass loss m_{loss} that we compute as shown in Equation 4.5

$$m^n_{loss} = \rho_{ice} \sum_{\mathbf{x}\in\mathcal{X}} M_{lump}(\mathbf{x}) \left(H^0_i(\mathbf{x}) - H^n_i(\mathbf{x})\right) \tag{4.5}$$

where M_{lump} is the lumped mass matrix (obtained from the *FEM* model) and ρ_{ice} is the ice density.

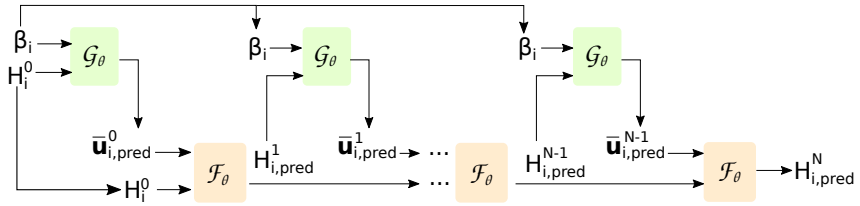


FIG. 4.3. *Coupled neural operators* $\mathcal{G}$ *and* $\mathcal{F}$

Finally, it is worth mentioning that all the considered neural operators and thus, the corresponding deep neural networks, were fully implemented in JAX [33], while the open-source Optax [34] library was used to set up the stochastic gradient-based optimizer and the learning rate scheduler.

5. Data preprocessing and training details. The *FEM* model of the Humboldt glacier in Greenland was run forward in time based on the SSA model considering $Q = 200$ basal friction samples, $\beta_i(\mathbf{x}),\ i = 1, \cdots, Q$ drawn from the distribution 3.2. For each β_i sample, the depth-averaged velocity $\bar{\mathbf{u}}$ and ice thickness H were determined over a period of $N = 50$ years, $n = 0, \cdot, N$ evaluated at $M = 1426$ grid points $\mathbf{x} \in \mathcal{X}$ corresponding to the unstructured mesh of the glacier. Thus, the training and testing dataset were organized as a triplet of multidimensional vectors of the form 5.1, where the bracket-enclosed subscripts denote the number of data samples in each dimension of the vector.

$$\left[\{\bar{\mathbf{u}}H\}_{\left[(N-\bar{P}(n))\times Q, M\right]}, \{\mathcal{X}\}_{\left[(N-\bar{P}(n))\times Q, M\right]}, \{H\}_{\left[(N-\bar{P}(n))\times Q, M, p=1,\cdots,\bar{P}(n)\right]}\right] \tag{5.1}$$

As in previous work [20], the data corresponding to the first 20 β samples were used for testing, whereas the rest of data (β samples from 21 to 200) were used for training the neural operator $\mathcal{F}$.

The training parameters for the two approaches (*vanilla* and *SVD*-based) of the operator $\mathcal{F}$ are listed in Table 3.2. For the former, the architecture of both the *branch* net and the *trunk* net consisted of 4 hidden layers and 300 neurons per layer (denoted as 4×300) with ReLU activation function. Conversely, for the *SVD*-based case, a single neural network was required since only the *branch* net is considered for training while the *trunk* net was replaced with a precomputed set of basis functions obtained from an *SVD* of the data. In specific, by following an heuristic approach, we found that most of thickness data information is retained with 300 eigenmodes as shown in Figure 5.1, where the largest 500 singular values are depicted.

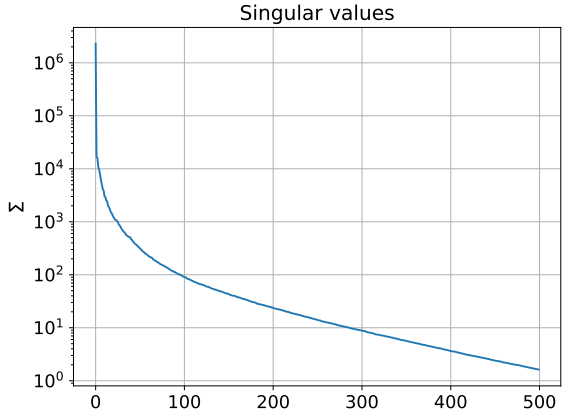


FIG. 5.1. *Largest singular values from the ice thickness H dataset*

Regularization in the loss function 4.2 was introduced with a penalty coefficient of 5×10^{-5} to mitigate possible overfitting. The Adam optimizer was set by default and the learning rate to optimize the networks' hyperparameters was set to 1×10^{-4}, while the learning rate to optimize the self-adaptive weights was set to 5×10^{-5}. For both learning rates, an independent scheduler was set to decay the learning rate exponentially from its initial value at a 0.9 rate every $100k$ epochs. Finally, the neural operator $\mathcal{F}$ was trained with $p = \{1, 2, \cdots, 7\}$ compositions of itself to conduct a composition-dependant analysis such that minimizes the loss function while getting accurate predictions. Table 5.1 summarizes the training details of the neural operators.

TABLE 5.1
Training details

Approach	Architecture	Compositions	Epochs
Vanilla	4×300 (b&t)*	1-7	300k
SVD	4×300 (b)*	1-7	500k

* b refers to branch net and t refers to trunk net

6. Ice thickness DeepONet training. As mentioned in section 5, two approaches for the $\mathcal{F}$ operator (*vanilla* and *SVD*-based) composed p-times with itself were evaluated. Figure 6.1 illustrates the training and testing errors as a function of the iterations (epochs) corresponding to the *vanilla* neural operator for $p = \{2, 4, \text{and } 6\}$ compositions.

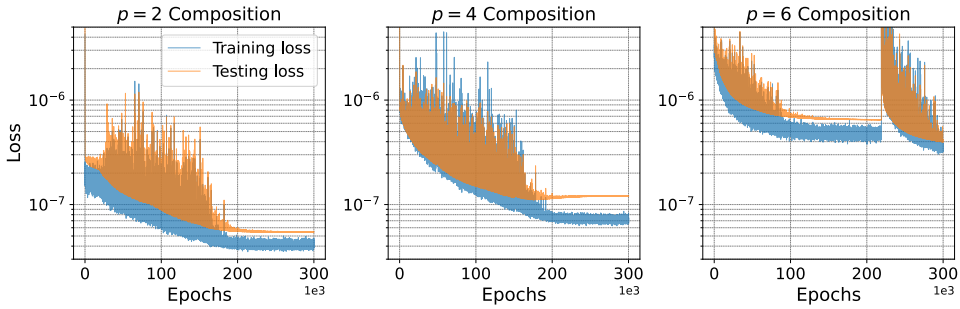


FIG. 6.1. *Loss values for training and testing the vanilla neural operator* $\mathcal{F}$

In a similar fashion, Figure 6.2 depicts the training and testing errors for the *SVD*-based neural operator for the same number of compositions aforementioned. From these results, it can be noticed that the errors flatten at greater loss function values as the number of compositions increases for both the *vanilla* and *SVD*-based approaches. This behavior is attributed to the fact that the loss function is more complex as the compositions augment and hence, the minimization of the loss function becomes less trivial. In addition, note that, when comparing the two approaches, for the same number of compositions, the loss function value decays faster and it is less noisy for the *SVD* operator, which indicates that this latter is faster to train while getting a similar accuracy compared with its *vanilla* counterpart. This trend can be linked to the fact that the *SVD* operator features less trainable parameters since a single neural network is employed and hence, the minimization of the loss function converges faster with less computational burden.

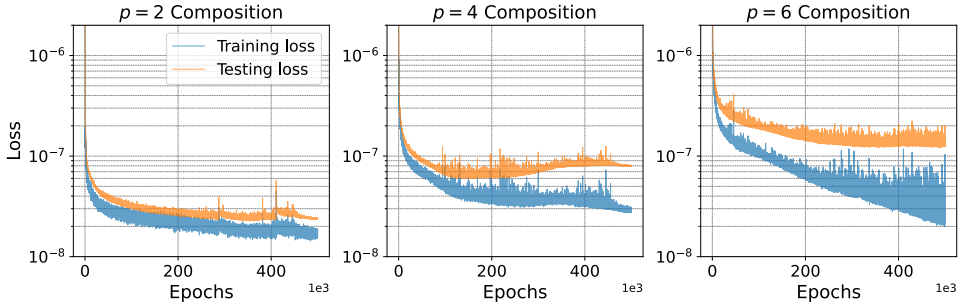


FIG. 6.2. *Loss values for training and testing the SVD-based neural operator* $\mathcal{F}$

Consider that, although the loss function value is greater as the number of compositions of the operator $\mathcal{F}$ augments, composing the $\mathcal{F}$ operator with itself up to 3 or 4 times demonstrates a lessening of the relative $\mathcal{L}_2$ error in the prediction of the ice thickness. Then, after this threshold value of compositions, it is observed that the relative error flattens for both the *vanilla* and *SVD*-based operator as depicted in Figure 6.3, where the relative $\mathcal{L}_2$ of the ice thickness H for a 50-year prediction was computed. Furthermore, note that adding more compositions to the operator results into longer training times as shown in red on the right-hand side axis of the plot from Figure 6.3. For instance, observe that for a 4-step composition of the operator $\mathcal{F}$, training the *vanilla* DeepONet takes around 8 times longer than training the *SVD*-based operator to get similar accuracy.

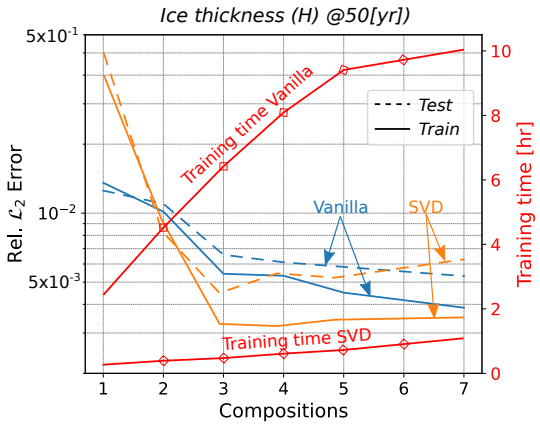


FIG. 6.3. *Comparison between the vanilla and SVD-based neural operator $\mathcal{F}$ for (left) - relative $\mathcal{L}_2$ error of the ice thickness h for a 50-year prediction and (right) - training time*

7. Coupled-operator predictions. In this section, we present the predicted results obtained by coupling the two neural operators: i) $\mathcal{G}$, that predicts the depth-averaged velocity $\bar{\mathbf{u}}$ as a function of the basal friction field β and the ice thickness H at a given time step $t = n$ and ii) $\mathcal{F}$, that maps the ice thickness H and deep-averaged velocity $\bar{\mathbf{u}}$ from a given time step $t = n$ to the next time step $t + \Delta t = n + 1$

7.1. Depth-averaged velocity and ice thickness. The relative $\mathcal{L}_2$ error (relative to the *FEM* models) of the predicted depth-averaged velocity $\bar{\mathbf{u}}$ and the ice thickness H for a time range of 50 years obtained by the framework of coupled neural operators is shown in Figure 7.1. The left-hand side plot shows the relative $\mathcal{L}_2$ error for the $\bar{\mathbf{u}}$ prediction and it can be noticed that the error is considerably stable and low (less than 6% for the training dataset and less than 12% for the testing dataset). This highlights the high accuracy that the $\mathcal{G}$ operator achieves. Note that the two approaches considered for the neural operator $\mathcal{F}$ (*vanilla* and *SVD*) are listed in this subplot. This simply means that, although the $\mathcal{G}$ operator was used to predict the ice flow velocity $\bar{\mathbf{u}}$, it was coupled with one of the two types of the operator $\mathcal{F}$ to evolve the solution of the ice thickness H over time. Hence, the legend shown in 7.1 refers to the coupled framework of operators $\mathcal{G}$ and $\mathcal{F}$ as a single model that characterizes the ice-sheet dynamics.

On the right-hand side of Figure 7.1, the relative $\mathcal{L}_2$ error for the ice thickness H as function of the 50-year time span considered is depicted. Consider that, although the relative $\mathcal{L}_2$ error increases and it is accumulated over time, the maximum value among the training and testing datasets does not exceed the 0.3% for both approaches. Hence, the coupled

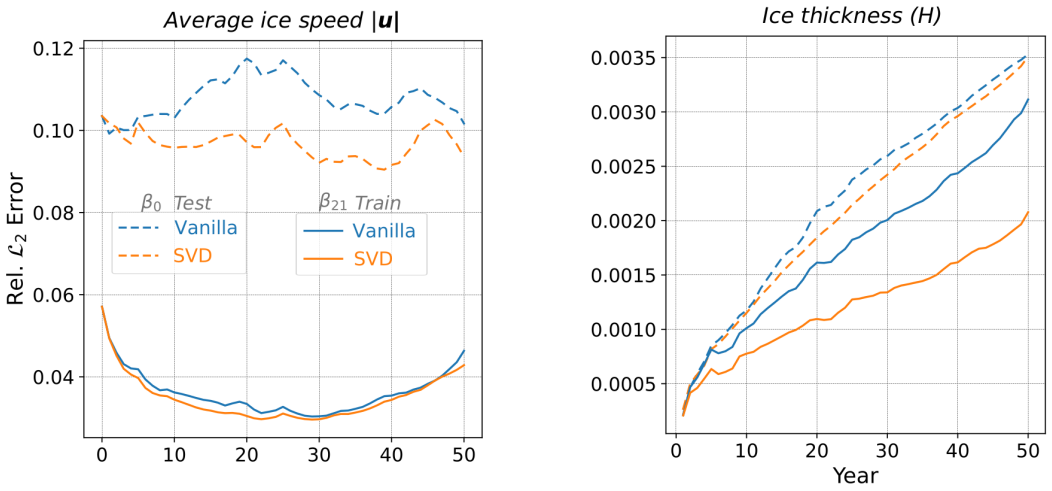


FIG. 7.1. *$\mathcal{L}_2$ Error of ice depth-average velocity $\bar{\mathbf{u}}$ (left) and ice-thickness H (right) with coupled neural operators for a 50 year time span*

operators approach demonstrates to be a highly accurate alternative method able to predict the dynamics of the ice sheet for the given set of initial conditions (β and H^n) at a moderate computational cost.

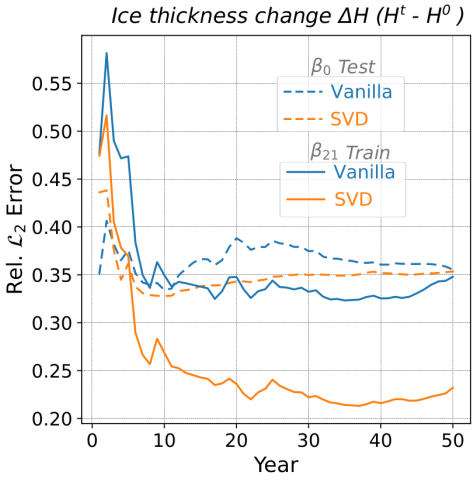


FIG. 7.2. *$\mathcal{L}_2$ Error of the change of ice thickness (ΔH) with coupled neural operators for a 50-year time span*

It is important to mention that it is often the case that predicting the ice-thickness dynamics itself is not the main goal. In fact, one of the most important parameters associated to the sea-level rise is the rate of change of the ice mass (mass loss), which in turn is related to the change of ice thickness ΔH. Thus, we considered important to determine the relative $\mathcal{L}_2$ error of the predicted ΔH relative to the corresponding value obtained by the *FEM* model. We computed the relative error of ΔH for each predicted year, where the difference is always referred to the initial thickness at time $t = 0$, i.e., we defined $\Delta H^n := H^n - H^0$ for a given year n, $n \in \{1, \cdots, 50\}$. The aforementioned errors are shown in Figure 7.2,

where one can note that the error is significantly higher at the beginning and then, after approximately the 10-th year prediction, the error decreases and it flattens. This is because the rate of change of ice thickness relative to the thickness itself at early years is very low, which explains the sudden drop in the relative error.

Finally, Figure 7.3 illustrates the two-dimensional field of the absolute error referred to ΔH for a 50-year prediction for a testing β case for the two $\mathcal{F}$ operators considered. Although, the absolute error for both methods are within a similar range, it can be observed that the largest errors are located on different regions across the two-dimensional field. This can be related to the difference of the basis functions of the DeepONets since they were obtained by different approaches: i) a standard *trunk* network and ii) a precomputed *SVD* basis from the ice thickness data.

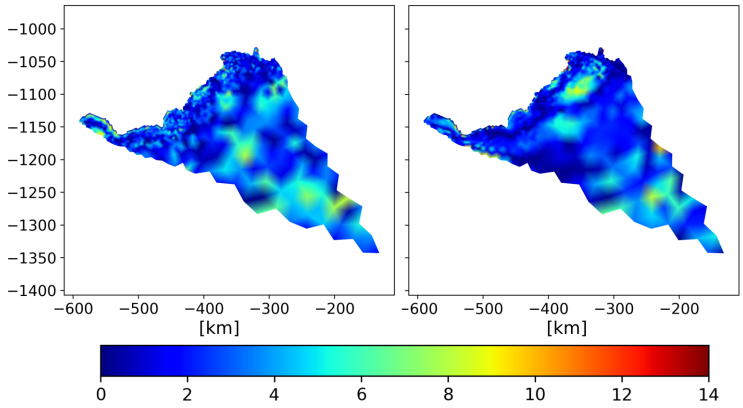


FIG. 7.3. *Absolute error of ΔH field for a 50-year prediction*

7.2. Estimation of ice mass change. As mentioned, the mass loss of a glacier is one of the most important quantities of interest in ice-sheet modeling because it is a proxy for sea-level rise. On the left hand side, Figure 7.4 shows a comparison of the mass change prediction (in gigatons) obtained with a *FEM* model and the *vanilla* coupled operator over the 50-year period. Likewise, on the right hand side, a comparison between the mass change prediction obtained using the *FEM* model and the *SVD*-based coupled neural operator is presented. Observe that the *SVD*-based approach outperforms the *vanilla* DeepONet architecture as the predicted mass change, for the same β sample, better aligns with the *FEM* prediction in the former case.

8. Summary and Future Work. A fully coupled neural operator model was developed and implemented to predict the ice-sheet dynamics of the Humboldt glacier. This model is able to handle high-dimensional parameters spaces, making it suitable to account for large uncertainties like the basal friction. We demonstrated that the model predicts both the velocities and the ice thickness with high accuracy in a faster manner when compared with a standard *FEM* approach. However, the prediction of the rate of change of the ice thickness still presents a relatively high error, which increases the uncertainty of the predicted mass loss. We think that training the coupled neural operators concurrently from their current states (trainable parameters) will increase the accuracy of the models and hence, we expect a reduction of the uncertainty in the mass loss prediction. The next step is to implement a concurrent training of the coupled neural operators by training only the parameters corresponding to the last layer of the neural networks of the neural operators.

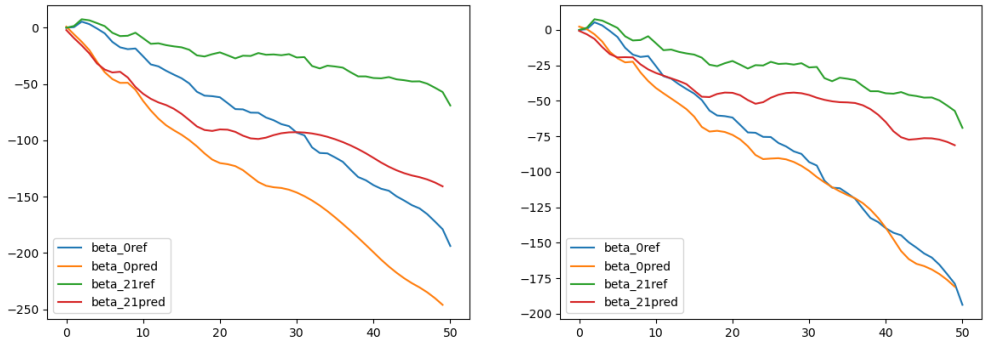


FIG. 7.4. *Mass change ($-M_{loss}$) comparison between the vanilla and SVD-based $\mathcal{F}$ neural operators for a 50-year time span*

Future research involves the implementation of an statistics analysis to determine the accuracy of the model when subject to unseen β samples with the mass loss as figure of merit. Other future research directions contemplate the implementation of sampling strategies to generate a less expensive dataset to train the neural operators with the same accuracy and the implementation of physics-informed coupled neural operators to characterize ice-sheet dynamics.

REFERENCES

[1] Benjamin P Horton, Nicole S Khan, Niamh Cahill, Janice SH Lee, Timothy A Shaw, Andra J Garner, Andrew C Kemp, Simon E Engelhart, and Stefan Rahmstorf. Estimating global mean sea-level rise and its uncertainties by 2100 and 2300 from an expert survey. *npj Climate and Atmospheric Science*, 3(1):18, 2020.

[2] Tamsin L Edwards, Sophie Nowicki, Ben Marzeion, Regine Hock, Heiko Goelzer, Hélène Seroussi, Nicolas C Jourdain, Donald A Slater, Fiona E Turner, Christopher J Smith, et al. Projected land ice contributions to twenty-first-century sea level rise. *Nature*, 593(7857):74–82, 2021.

[3] Mauro Perego, Max Gunzburger, and John Burkardt. Parallel finite-element implementation for higher-order ice-sheet models. *Journal of Glaciology*, 58(207):76–88, 2012.

[4] Wei Leng, Lili Ju, Max Gunzburger, Stephen Price, and Todd Ringler. A parallel high-order accurate finite element nonlinear stokes ice sheet model and benchmark experiments. *Journal of Geophysical Research: Earth Surface*, 117(F1), 2012.

[5] Irina K Tezaur, Mauro Perego, Andrew G Salinger, Raymond S Tuminaro, and Stephen F Price. Albany/felix: a parallel, scalable and robust, finite element, first-order stokes approximation ice sheet solver built for advanced analysis. *Geoscientific Model Development*, 8(4):1197–1220, 2015.

[6] Andy Aschwanden, Mark A Fahnestock, Martin Truffer, Douglas J Brinkerhoff, Regine Hock, Constantine Khroulev, Ruth Mottram, and S Abbas Khan. Contribution of the greenland ice sheet to sea level over the next millennium. *Science advances*, 5(6):eaav9396, 2019.

[7] Tobin Isaac, Noemi Petra, Georg Stadler, and Omar Ghattas. Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the antarctic ice sheet. *Journal of Computational Physics*, 296:348–368, 2015.

[8] B. Recinos, D. Goldberg, J. R. Maddison, and J. Todd. A framework for time-dependent ice sheet uncertainty quantification, applied to three west antarctic ice streams. *The Cryosphere*, 17(10):4241–4266, 2023.

[9] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153):1–43, 2018.

[10] William J Dally, Stephen W Keckler, and David B Kirk. Evolution of the graphics processing unit (gpu). *IEEE Micro*, 41(6):42–51, 2021.

[11] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52(1):477–508, 2020.

[12] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[13] Ravi G. Patel, Nathaniel A. Trask, Mitchell A. Wood, and Eric C. Cyr. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373:113500, 2021.

[14] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[15] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[16] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, pages 1–9, 2024.

[17] Patricio Clark Di Leoni, Lu Lu, Charles Meneveau, George Em Karniadakis, and Tamer A Zaki. Neural operator prediction of linear instability waves in high-speed boundary layers. *Journal of Computational Physics*, 474:111793, 2023.

[18] Zhiping Mao, Lu Lu, Olaf Marxen, Tamer A Zaki, and George Em Karniadakis. Deepm&mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. *Journal of computational physics*, 447:110698, 2021.

[19] Somdatta Goswami, Minglang Yin, Yue Yu, and George Em Karniadakis. A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391:114587, 2022.

[20] QiZhi He, Mauro Perego, Amanda A Howard, George Em Karniadakis, and Panos Stinis. A hybrid deep neural operator/finite element method for ice-sheet modeling. *Journal of Computational Physics*, 492:112428, 2023.

[21] Amanda A Howard, Mauro Perego, George Em Karniadakis, and Panos Stinis. Multifidelity deep operator networks for data-driven and physics-informed problems. *Journal of Computational Physics*, 493:112462, 2023.

[22] Simone Venturi and Tiernan Casey. Svd perspectives for augmenting deeponet flexibility and interpretability. *Computer Methods in Applied Mechanics and Engineering*, 403:115718, 2023.

[23] Joseph Hart, Mamikon Gulian, Indu Manickam, and Laura P Swiler. Solving high-dimensional inverse problems with auxiliary uncertainty via operator learning with limited data. *Journal of Machine Learning for Modeling and Computing*, 4(2), 2023.

[24] Ravi Patel, Indu Manickam, Myoungkyu Lee, and Mamikon Gulian. Error-in-variables modelling for operator learning. In *Mathematical and Scientific Machine Learning*, pages 142–157. PMLR, 2022.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[27] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[28] Katiana Kontolati, Somdatta Goswami, Michael D Shields, and George Em Karniadakis. On the influence of over-parameterization in manifold based surrogates and deep neural operators. *Journal of Computational Physics*, 479:112008, 2023.

[29] Kurt M Cuffey and William Stanley Bryce Paterson. *The physics of glaciers*. Academic Press, 2010.

[30] M Weis, R Greve, and K Hutter. Theory of shallow ice shelves. *Continuum Mechanics and Thermodynamics*, 11:15–50, 1999.

[31] Mauro Perego, Stephen Price, and Georg Stadler. Optimal initial conditions for coupling ice sheet models to earth system models. *Journal of Geophysical Research: Earth Surface*, 119(9):1894–1917, 2014.

[32] Levi D McClenny and Ulisses M Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.

[33] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, et al. Jax: composable transformations of python+ numpy programs, v0. 3.13, 2018.

[34] Optax. Optax documentation, 2024. Accessed: 2024-08-14.