

Computational Methods for Data-Driven Solutions to Inverse Problems in Geophysics

by

Cody Rucker

A dissertation accepted and approved in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science

Dissertation Committee:

Brittany Erickson, Chair

Daniel Lowd, Core Member

Allen Malony, Core Member

Leif Karlstrom, Institutional Representative

University of Oregon

Spring 2025

© 2025 Cody Rucker

This work, including text and images of this document but not including supplemental files (for example, not including software code and data), is licensed under a Creative Commons

Attribution 4.0 International License.



DISSERTATION ABSTRACT

Cody Rucker

Doctor of Philosophy in Computer Science

Title: Computational Methods for Data-Driven Solutions to Inverse Problems in Geophysics

Numerical models are often tasked with resolving multiscale features in space and time which can result in high computational costs that become prohibitive in higher dimensions. This high computational cost drastically affects the cost of model evaluation over many potential parameters and leads modelers to be selective about which aspects of the physical system to emulate. In particular, we see this issue arises in the context of geophysical models where structures of interest (i.e. faults, magma reservoirs) exist below the Earth's surface and cannot be directly studied. Instead we rely on surface observations that may be used to infer subsurface properties. However, the measured data may not be sufficiently informative to uniquely infer information about the subsurface state. This suggests a Bayesian approach for solving the resulting inverse problem which further increases computational cost and can make simulations intractable even in high-performance computing. We are motivated by these challenges to develop novel approaches to solving inverse problems that may be more computationally feasible while still providing useful insight into the subsurface state of the Earth. This dissertation presents two approaches for handling such inverse problems. First we model a pressurized, subsurface magma reservoir surrounded by thermo-viscoelastic host rock. Simulations were carried out using a high-order finite-element method and explicit time-stepping for which numerically stable time steps were derived. System response is characterized by a transfer function connecting peak surface deformation with reservoir stress. The model reveals that the volume of the viscous region surrounding the reservoir is dependent on the frequency of reservoir forcing. Our use of analytic signals yields a non-singular transfer function that simplifies the parametric considerations of the model and decomposes the output into component signals that may be used to invert for stress along the reservoir wall. For our second problem we model an antiplane strike-slip fault governed by a nonlinear rate-and-state aging law. A solution is approximated by training a Physics-Informed Neural Network (PINN) to satisfy an Initial Boundary Value Problem (IBVP). By including an additional network within the friction law, we train the network to solve the IBVP and invert for depth-dependent friction parameters without significantly

increasing training cost. Additionally we demonstrate that exact enforcement of initial conditions results in better network approximations of the IBVP solution and the inferred friction parameter. Moreover, we verify that the PINN maintains its accuracy over various resolutions of the space-time domain. Our findings constitute important new techniques for solving inverse problems by leveraging both analytic tools as well as emergent deep learning techniques.

This dissertation includes previously published co-authored material.

CURRICULUM VITAE

NAME OF AUTHOR: Cody Rucker

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
Portland State University, Portland, OR, USA
Oregon State University, Corvallis, OR, USA

DEGREES AWARDED:

Master of Science, Mathematics, 2019, Portland State University
Bachelor of Science, Mathematics, 2017, Oregon State University

AREAS OF SPECIAL INTEREST:

Numerical PDEs
Machine Learning
High performance computing

PROFESSIONAL EXPERIENCE:

Graduate Research Assistant, University of Oregon, Advisor: Brittany A. Erickson, Fall 2019
- Spring 2025
Graduate Teaching Assistant, University of Oregon, Fluency with Info Tech, Computer Science
I, Intern Data Structures, Spring 2020, Fall 2021, Winter 2021

GRANTS, AWARDS AND HONORS:

Lokey Graduate Science Award, University of Oregon, 2019

PUBLICATIONS:

Rucker, C. and Erickson, B. A. (2024). Physics-informed deep learning of rate-and-state fault friction. Computer Methods in Applied Mechanics and Engineering, 430:117211. doi:10.1016/j.cma.2024.117211.

- Crozier, J., Karlstrom, L., Montgomery-Brown, E., Angarita, M., Cayol, V., Bato, M.-G., Wang, T. A., Grapenthin, R., Shreve, T., Anderson, K. R., Astort, A., Bodart, O., Cannavó, F., Currenti, G., Dabaghi, F., Erickson, B. A., Garg, D., Head, M., Iozzia, A., Cheol Kim, Y., Le Mével, H., Novoa Lizama, C., Rucker, C., Silverii, F., Trasatti, E., Zhan, Y. (2023). Understanding the drivers of volcano deformation through geodetic model verification and validation. *Bulletin of Volcanology*, 85(12):74. doi:10.1007/s00445-023-01687-4.
- Rucker, C., Erickson, B. A., Karlstrom, L., Lee, B., and Gopalakrishnan, J. (2022). A computational framework for time-dependent deformation in viscoelastic magmatic systems. *Journal of Geophysical Research: Solid Earth*, 127(9):e2022JB024506. doi:10.1029/2022JB024506.

ACKNOWLEDGEMENTS

Special thanks to Dr. Leif Karlstrom and Dr. Jay Gopalakrishnan who helped mentor me during our work on the magma reservoir project.

Thank you to Dr. Daniel Lowd, Dr. Allen Malony, and Dr. Boyana Norris for serving on my committee and helping me acquire necessary the skills for my multidisciplinary studies.

Thank you to Cheri Smith and Nicole Moynahan who helped guide me through the UO PhD program.

To Brittany, for showing me what kind of scientist I'd like to be and reminding me to go outside every now and then.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	14
1.1. Main Research Problems	14
1.1.1. High-performing Numerical Methods for Multiscale Problems	14
1.1.2. Limited Observational Domain of Geophysical Models	14
1.1.3. Inverse Problem Dependence on Forward Models	15
1.1.4. Thesis Topic	15
II. METHODOLOGY	17
2.1. Introduction	17
2.2. Background	18
2.2.1. Feed Forward Neural Network (FFNN)	18
2.2.2. Universal Approximation Theorems	19
2.2.3. Enforcing PDE in ML Training	20
2.3. Physics-Informed Neural Networks	21
2.3.1. Data-Driven Solutions	22
2.3.2. Data-Driven Discovery	23
2.3.3. Discrete Time Model	24
2.4. Alternate PINN Formulations	26
2.4.1. Variational Form (vPINN)	27
2.4.2. Conservative Form (cPINN)	29
2.5. Parametric PDE	30
2.5.1. DeepONet	31
2.6. Approximation Theory	32
2.6.1. Approximation Error $\mathcal{E}_A$	34
2.6.2. Generalization Error $\mathcal{E}_G$	35
2.7. Future Research Areas	36
2.7.1. Mesh-Free Integration	36
2.7.2. Initial Guesses for Iterative Methods	37
2.7.3. Persistent, General Solutions	38

Chapter	Page
2.8. Final Remarks	39
III. MAGMA RESERVOIR	40
3.1. Introduction	40
3.2. Mathematical Framework	43
3.2.1. Problem Formulation and Geometry	43
3.2.2. Governing Equations	45
3.3. Computational Framework	47
3.3.1. Solving the Static Equilibrium Equation	47
3.3.2. Temporal Discretization	49
3.3.3. Model Specifics and Non-Dimensionalization	50
3.3.4. Temperature-Dependent Material Parameters	53
3.4. Analysis of time dependent viscoelastic deformation	55
3.4.1. Insights from the 1D Maxwell Model	55
3.4.2. Transfer Function and Analytic Signals	59
3.4.3. Numerical Calculations of the Transfer Function	62
3.5. Computational Results	63
3.6. Discussion	71
3.6.1. Computational Considerations for Time-evolving Magmatic Systems	71
3.6.2. Frequency Dependent Magmatic Deformation	72
3.6.3. Implications for Transcrustal Magmatic Systems	76
IV. PINN INVERSION	79
4.1. Context and Motivation	79
4.2. Physics-Informed Deep Learning Framework	82
4.2.1. Feed-forward Deep Neural Networks	82
4.2.2. PINN Architecture	83
4.3. Learning Problems for Earthquakes on Rate-and-State Faults	84
4.3.1. 1D Illustration	84
4.3.2. 2D Application in Seismic Faulting	88
4.3.3. Forward and Inverse Problems for the 2D Application	92
4.3.3.1. Forward Problem	93

Chapter	Page
4.3.3.2. Inverse Problem	94
4.3.3.3. Soft vs. Hard Enforcement of Boundary Conditions	94
4.4. 2D Verification, Validation and Applications	95
4.4.1. Verification with the Method of Manufactured Solutions	95
4.5. Summary and Future Work	99
4.6. Data Availability Statement	101
V. CONCLUSION AND FUTURE WORK	102
5.1. Conclusion	102
5.2. Future Work	102
5.2.1. Transfer Function	102
5.2.2. PINNs	102
APPENDICES	
A. AREA EXAM APPENDIX	104
A.1. Gauss-Legendre Runge-Kutta Method	104
A.2. Special Considerations	106
A.3. Adaptive Activation Functions	107
A.3.1. Layer-Wise Locally Adaptive Activation	107
A.3.2. Neuron-Wise Locally Adaptive Activation	108
B. MAGMA RESERVOIR APPENDIX	109
B.1. Stability and Verification via Convergence Tests	109
B.2. Construction of general time-varying signals	114

LIST OF FIGURES

Figure		Page
1.	Generic PINN diagram	22
2.	Runge-Kutta bounds on temporal error	26
3.	Discrete-time model for PINNs	27
4.	DeepONet network diagram	33
5.	Integral approximation via ML training	37
6.	Computational domain for pressurized reservoir problem	44
7.	Spatial plots for material parameters	56
8.	Transfer function phase lag as a function of forcing period	65
9.	Strain/displacement evolution with respect to forcing period	68
10.	Spatial pattern of surface displacements and subsurface distribution of von Mises stress	69
11.	Transfer function between reservoir pressure and maximum vertical surface displacement as a function of sinusoidal pressure forcing period	70
12.	Analytic prediction of max displacement from composite input signal based on the transfer function	74
13.	Spatial regions associated with a local Deborah number $De = 10$ for varying forcing periods	77
14.	PINN diagram for generic IBVP	84
15.	A schematic of the PINN framework for solving a 1D IBVP with rate and state friction	88
16.	Approximate solutions given by trained displacement and state networks	89
17.	Domain schematic for antiplane strike-slip fault problem	91
18.	Network diagram for enforcing RSF while inferring depth-dependent friction along the fault	95
19.	Displacements and L^2 -errors for network solving inverse problem	100
20.	Convergence of loss components and inferred friction during training	100
B.1.	Temperature impact on temporal stability	111
B.2.	Example of a pressurization time series consisting of a linear ramp to a maximum value	114

LIST OF TABLES

Table	Page
1. Parameters for magma reservoir problem	64
2. Parameter values used in the manufactured solution tests.	96
3. Errors of loss components for forward/inverse problems with hard/soft boundary enforcement	98
B.1. Parameters used in Convergence Tests and their Symbols.	113
B.2. Spatial convergence data	113
B.3. Temporal convergence data	113

CHAPTER I

INTRODUCTION

1.1 Main Research Problems

Mathematical modeling is a multidisciplinary field that provides abstract representations for concrete systems. Models are simplified emulators which can be analyzed, probed, or expanded to gain insight about the system in question. The abstract nature of modeling allows for experimentation which may not be physically viable, as is common in geophysics where much of the system lies beneath Earth’s surface. These models are often posed as Initial Boundary Value Problems (IBVP) consisting of governing equations, constitutive laws, and constraints (initial/boundary conditions and kinematic equations) which are specified to ensure that the problem is mathematically well-posed. However, solutions to such IBVP are not guaranteed to have a closed form and thus we often turn to numerical methods. Here we provide a brief overview of the unique challenges that pertain to numerical methods (1.1.1), geophysical models (1.1.2), and inverse problems (1.1.3) that motivate the central topic of this dissertation as given in section 1.1.4.

1.1.1 High-performing Numerical Methods for Multiscale Problems. Traditional numerical methods such as Finite Differences (FDM) and Finite Element (FEM) seek a solution to IBVP by discretizing the domain and computing a solution over the resulting mesh. In the case of hyperbolic PDE, a choice must be made about how to handle temporal evolutions. *Implicit* time stepping allows for stable temporal evolutions to be computed at an increase to the computational cost of each time step while *explicit* time stepping provides a cheaper compute cost at each step but is subject to Courant–Friedrichs–Lewy (CFL) stability criteria. In the presence of multiscale spatio-temporal features, these discretization techniques can produce large linear systems that are costly to solve and are subject to the curse of dimensionality. This high computational cost can be especially restrictive when it is necessary to solve the IBVP over a field of possible parametric inputs as is often the case in geophysical modeling.

1.1.2 Limited Observational Domain of Geophysical Models. In developing geophysical models, subsurface characteristics (e.g. fault/reservoir geometry, stress history, thermal gradient) are crucial for understanding how a system might evolve in time. However, this subsurface information is not known apriori and we must include assumptions about the system in order to properly define an IBVP. This leads to a scenario in which the solutions to the IBVP are dependent

on the assumptions that modelers make about the system. A model that produces a solution given a set of assumed parameters is said to solve a *forward problem* and is a necessary component in trying to relate observed data to unobservable parameters.

1.1.3 Inverse Problem Dependence on Forward Models. The significance of the system’s subsurface state leads us to consider an inverse problem to the mathematical model. That is, we would like to use observations at Earth’s surface to infer aspects of the system’s state within the unobservable subsurface. Consider the forward model $\mathcal{F}$ to be an operator acting on a functional space of system parameters. Then $\mathcal{F}$ defines a correspondence between parameters and data so that the inverse problem amounts to the following: Given data q , find parameter p such that

$$q = \mathcal{F}[p]. \quad (1.1)$$

The issue we encounter in modeling Earth systems is that surface data may not be sufficiently informative to accurately reconstruct the subsurface state (which may not be unique to the surface deformation). Because of this, we would require the Bayesian framework for inverse problems which outputs a distribution of possible parameters for the observed data. However, this approach requires the solution of the forward problem over a large number of possible parameters in order to properly sample the probability distribution of an unknown parameter given known data [1]. The large number of forward-mode evaluations makes this solution technique computationally challenging especially if the forward model is itself compute intensive.

1.1.4 Thesis Topic. This dissertation demonstrates two approaches to tackling inverse problems that aim to overcome the prohibitive computational cost. The first is an analytic tool that is used to deconstruct surface signals and compute component-wise transfer functions that allow for stress inversions around a subsurface, pressurized, magma reservoir. The second integrates machine Learning (ML) techniques to solve the forward and inverse problem (simultaneously without much increase to computational cost) to obtain a solution and infer friction along a strike-slip fault governed by nonlinear rate-and-state friction.

The remainder of this dissertation is organized as follows: Chapter II provides an overview of the ML technique known as Physics-Informed Neural Network (PINN) and its application in solving differential equations and inverse problems, Chapter III details our framework for time-dependent evolution of subsurface, magma, reservoir in a thermo-viscoelastic medium and introduces an analytic signal approach to computing transfer functions relating stress and surface deformation,

in Chapter IV we use a PINN to infer the friction parameters along a strike-slip fault governed by rate-and-state-friction.

Co-author Acknowledgement. Specific contributions are detailed at the beginning of each chapter. Chapter II contains unpublished material, for which, I am the sole author. Chapter III contains published material co-authored by Dr. Brittany A. Erickson, Dr. Leif Karlstrom, Brian Lee, and Dr. Jay Gopalakrishnan. Chapter IV contains published material co-authored by Dr. Brittany A. Erickson.

CHAPTER II

METHODOLOGY

This chapter consists of unpublished material from the departmental Area Exam requirement. This Area Exam is a survey of the state-of-the-art in physics-informed neural networks and their use in approximating solutions to initial-boundary value problems. While I am the sole author of this Area Exam, I received help (in the form of feedback, and suggestions) from my dissertation committee members: Dr. Brittany A. Erickson, Dr. Daniel Lowd along with Dr. Boyana Norris, and Dr. Hank Childs.

2.1 Introduction

In mathematics, PDE are used to encode physical phenomena in terms of differential relationships. Solutions to PDE are functions which exhibit the desired physical property, but such solutions are not guaranteed to have an analytic form so we often turn to numerical schemes like the Finite Difference Method(FDM) or Finite Element Method(FEM) to approximate solutions. However, an emergent Deep Learning(DL) framework has recently demonstrated a surprising capacity for approximating PDE solutions. Though this DL framework lacks the robust mathematical theory of the traditional methods, it shows particular promise in solving problems for which traditional numerical methods are ill suited.

Traditional numerical methods are typically solved on a mesh which, due to the curse of dimensionality [2], leads to computationally intractable problems in higher dimensions. Additionally, traditional approaches are not amenable to real-world observation and ultimately cannot be efficiently generalized to solutions of parametric PDE. This is contrasted by the DL framework which produces a closed, analytic form for the solution. Because of this, the solution is continuous and defined at every point in the domain allowing one to evaluate the solution 'off-grid' without having to solve the PDE again. While the DL framework cannot compete with traditional methods in lower dimensions, the 2017 foundational framework, PINNs, [3] suggests that both approaches may be used in tandem to offset specific weaknesses of a method.

The rest of this paper is organized as follows: In section 2.2 we provide brief history of Artificial Neural Networks (ANN) in solving PDE. section 2.3 Introduces the PINN framework and we give details for handling forward and inverse learning problems along with a formulation for discretizing the network with respect to time. Once we have established the basic PINNs framework

section 2.4 covers some ways that the PINN framework may be altered to enhance certain model features. section 2.5 introduces operator methods which provide a means by which PINNs can be generalized to a family of PDE solutions. section 2.6 provides a look at the current state of an emerging approximation theory for physics-informed networks. Finally, I will conclude the paper with section 2.7 in which I give my thoughts on the framework and carve out future areas of research.

2.2 Background

We begin by introducing the Feed Forward Neural Network (FFNN) architecture and give its Universal Approximation Theorems (UAT). FFNN also inherit regularity from their activation functions and, along with UAT this allows them to be infinitely differentiable function approximators, a property that is shared with polynomials. This makes FFNN a valid option in solving PDE and we will show how Machine Learning (ML) can be constrained by PDE.

2.2.1 Feed Forward Neural Network (FFNN). Let $\mathbf{x} \in \mathbb{R}^n$, $W \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Then a single hidden layer of a neural network can be expressed as

$$\ell(\mathbf{x}; \theta) = \varphi.(W\mathbf{x} + b), \quad \theta = (W, b)$$

where φ is an activation function. That is, φ is bounded and

$$\begin{aligned} \lim_{x \rightarrow +\infty} \varphi(x) &= \alpha \\ \lim_{x \rightarrow -\infty} \varphi(x) &= \beta \end{aligned}$$

with $\alpha \neq \beta$. The notation $\varphi.(*)$ signifies component-wise application of the activation function. We construct a deep neural network by repeated composition of hidden layers. Let $L \in \mathbb{N}_{>0}$ be the network depth and let $\{\phi_i\}_{i=1}^L$ be a collection of activation functions along with a sequence of trainable network parameters $\{\theta_i\}_{i=1}^L$ where $\theta_k = (W_k, b_k)$ for each $0 < k \leq L$. Take $\mathbf{x} \in \mathbb{R}^n$ to be the network input and network parameters $(W_1, b_1) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$, $(W_k, b_k) \in \mathbb{R}^{m \times m} \times \mathbb{R}^m$ for $1 < k < L$, and $(W_L, b_L) \in \mathbb{R}^{d \times m} \times \mathbb{R}^d$. Then the recursive composition

$$\ell_0 = \mathbf{x}, \tag{2.1a}$$

$$\ell_k = \varphi_k.(W_k \ell_{k-1} + b_k) \quad \text{for } 0 < k < L \tag{2.1b}$$

defines a neural network $\mathcal{N} : \mathbb{R}^n \rightarrow \mathbb{R}^d$ (of depth L and width m) with network parameters θ by

$$\mathcal{N}(\mathbf{x}; \theta) = W_L \ell_{L-1} + b_L. \tag{2.2}$$

2.2.2 Universal Approximation Theorems. When considering such neural networks as function approximators there are two primary avenues to consider: arbitrary width and arbitrary depth. The arbitrary width formulation considers a continuous activation function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ and a collection of feed-forward neural networks $\mathcal{N}_n^\phi$ mapping $\mathbb{R}^n$ to $\mathbb{R}$ which have a single hidden layer with an arbitrary number of neurons. Then we get the following universal approximation theorem (UAT):

Theorem 1 (Arbitrary-Width UAT[4])

For compact subsets $K \subseteq \mathbb{R}^n$, $\mathcal{N}_n^\phi$ is dense in $C(K)$ if and only if ϕ is nonpolynomial.

Theorem 1 can be trivially extended to networks with a bounded number of hidden layers by including identity mapping layers [4]–[8].

The arbitrary-depth formulation, on the other hand, considers the approximation capabilities of networks with an arbitrary number of hidden layers each with fixed width. This formulation has been explored for Rectified Linear Units(ReLU) [9]–[11] and a minimal network size has been established as a necessary condition for deep networks to be universal approximators. However, as we will see later in the paper, the ReLU activation function cannot be utilized when constraining a network to satisfy PDE. More recently, a more general theorem has been proposed which shows the following: Let $\phi : \mathbb{R} \rightarrow \mathbb{R}$ and $n, d, m \in \mathbb{N}$. Denote by $\mathcal{N}_{n,d,m}^\phi$ the class of functions mapping $\mathbb{R}^n$ to $\mathbb{R}^d$ described by feed forward neural networks with n input neurons, d output neurons, and an arbitrary number of hidden layers, each with m neurons and activation function ϕ . Each neuron in the output layer uses the identity activation function. Then we have the following theorem:

Theorem 2 (Arbitrary-Depth UAT [4])

For any non-affine continuous function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ which is continuously differentiable at at least one point (with nonzero derivative at that point), and compact subset $K \subseteq \mathbb{R}^n$, the set $\mathcal{N}_{n,d,n+d+2}^\phi$ is uniformly dense in $C(K; \mathbb{R}^d)$.

Note that this theorem is quite general as far as the allowable activation functions. It includes polynomial activation functions which distinguishes it significantly from Theorem 1. Moreover, [4] proves that Theorem 2 can be extended to networks with activation functions which are bounded and continuous, but differentiable nowhere.

Theorem 3 ([4])

Let $w : \mathbb{R} \rightarrow \mathbb{R}$ be any bounded continuous nowhere differentiable function. Let $\phi(x) = \sin(x) + w(x)e^{-x}$, which is also nowhere differentiable. Let $K \subseteq \mathbb{R}^n$ be compact. Then $\mathcal{N}_{n,d,n+d+1}^\phi$ is dense in $C(K; \mathbb{R}^d)$ with respect to the uniform norm.

The oddity of this result lies in the fact that the set of differentiable functions is contained within the set of continuous functions. This means that we can uniformly approximate differentiable functions using activation functions which are differentiable nowhere. Thus deep neural networks maintain their universal approximation property for a pretty non-restrictive class of activation functions. Considering that gradient descent will require almost-everywhere differentiable activation functions, the choice of activation functions is limited by network training. The universal approximation theorems (in addition to the Neural Net’s closed, analytic form) establishes ANN as potential PDE solvers. To train a Network to satisfy a PDE we first need to augment the ML optimization problem to incorporate information from IBVP. Inclusion of the IBVP is what makes a network *Physics Informed*.

2.2.3 Enforcing PDE in ML Training. One of the earlier appearances of a physics-informed network was proposed by [12] as a numerical solution for PDE on orthogonal box domains and was later expanded to domains of complex geometry by [13]. They use the compact and differentiable closed form of an Artificial Neural Network (ANN) solution to construct a trial function consisting of two terms: One term has no trainable parameters and satisfies a desired Boundary Condition (B.C) while the second term is a trainable feed-forward network which does not contribute to B.C but can be trained to solve a minimization problem. While this approach is fairly simple on orthogonal box domains, it becomes more involved on complex domains. Additionally, the manual computation of the loss function derivative terms is prone to error and becomes quite burdensome on deep neural networks.

More recently, [14] presents a unified framework for approximating PDE solutions using artificial neural networks and gradient-based optimization. This approach captures the idea proposed in [12] and provides a good basis for understanding physics-informed networks. We start with the general form of a PDE

$$\mathcal{O}[u](\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad (2.3a)$$

$$\mathcal{B}[u](\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma \subset \partial\Omega, \quad (2.3b)$$

where $\mathcal{O}$ is a differential operator, f a forcing function, $\mathcal{B}$ a boundary operator and g the boundary data. Here we take $\Omega = \mathbb{R}^n$ and Γ is the region on the boundary $\partial\Omega$ where boundary conditions are imposed. We approximate the solution by $u(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \theta)$ and consider $N_\Omega, N_\Gamma \in \mathbb{N}_{>0}$ collocation points taken from Ω and Γ respectively. Then we may define the constrained optimization problem

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^{N_\Omega} \frac{1}{2} \frac{1}{N_\Omega} \|\mathcal{O}[\mathcal{N}](\mathbf{x}_i; \theta) - f(\mathbf{x}_i)\|^2, \quad (2.4)$$

subject to constraints at the boundary defined by

$$\mathcal{B}[\mathcal{N}](\mathbf{x}_i) = g(\mathbf{x}_i) \quad \forall i = \{1, \dots, N_\Gamma\}. \quad (2.5)$$

This can be reformulated as an unconstrained optimization problem by training a network $G = G(\mathbf{x})$ to be a smooth extension of boundary data g and training a distance network $D = D(\mathbf{x})$ to be a smooth distance function that computes the distance from an internal point $\mathbf{x} \in \Omega$ to the nearest boundary point in Γ . Then, define a new network approximation for u by

$$\tilde{\mathcal{N}}(\mathbf{x}; \theta) = G(\mathbf{x}) + D(\mathbf{x})\mathcal{N}(\mathbf{x}; \theta) \quad (2.6)$$

and the unconstrained optimization problem becomes

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^{N_\Omega} \frac{1}{2} \frac{1}{N_\Omega} \|\mathcal{O}[\tilde{\mathcal{N}}](\mathbf{x}_i; \theta) - f(\mathbf{x}_i)\|^2. \quad (2.7)$$

The particular form of Equation 2.6 guarantees that the boundary conditions are satisfied on Γ . By building the boundary data in to the approximate solution, we are implementing a *hard* enforcement of boundary conditions. Now, we point out that the unconstrained optimization problem given by Equation 2.7 is very similar to the framework proposed in [12], [13] but the trial solution is vastly simplified by training boundary and distance networks. Additionally, manual differentiation of the loss function is no longer necessary as Automatic Differentiation (AD) can be used. Using AD, it is now possible to efficiently construct loss functions that constrain a network's derivatives without needing to massively restrict the network's depth. This allows for deeper, more expressive networks to be used when approximating PDE with ANN.

2.3 Physics-Informed Neural Networks

The 2017 framework, PINNS, establishes a class of neural networks that are trained to respect the physical laws encoded by general nonlinear PDE [3]. By constraining the network using an Initial Boundary Value Problem (IBVP), we effectively reduce the complexity of the network solution space which allows the network to be trained on a smaller set of data points. While this kind of network

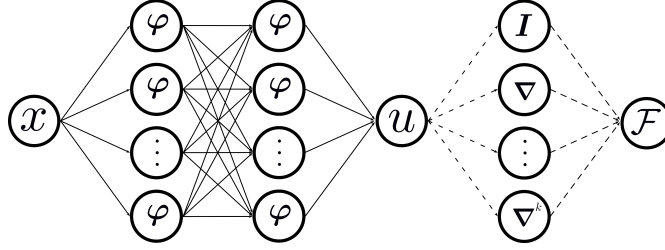


Fig. 1 Network diagram for a generic PINN with activations φ , input x , output u and PDE $\mathcal{F}$. Network connections shown with dashed lines represent non-trainable parameters.

constraint is not novel [12], the framework proposes a model set up that can be minimally modified to handle two main classes of problems: data driven solutions and data-driven discovery of PDE. Furthermore, temporal evolution can be handled continuously or discretely. This gives the framework significant flexibility and efficiently exploits the predictive power of neural networks [15]. In this section we will first detail the model set up for data-driven solutions and data-driven discovery before showing how to handle the discrete-time formulation.

Consider a spatial domain $\Omega \subseteq \mathbb{R}^n$ and time interval $\tau = [0, T]$ for some $T > 0$. Then $\hat{\Omega} = \Omega \times \tau$ is a spatio-temporal domain with boundary $\partial\hat{\Omega}$ on which we may define the general boundary value problem

$$u_t + \mathcal{O}[u; \lambda] = f, \quad (x, t) \in \hat{\Omega} \quad (2.8a)$$

$$\mathcal{B}[u] = g, \quad (x, t) \in \hat{\Gamma} \subseteq \partial\hat{\Omega} \quad (2.8b)$$

where $\mathcal{O}[\cdot; \lambda]$ is a nonlinear differential operator parametrized by λ , while f , $\mathcal{B}$, and g are similarly defined as in Equation 2.3. When the operator parameters λ are known apriori, we can solve the *forward* problem which approximates a solution u to Equation 2.8.

2.3.1 Data-Driven Solutions. We start by approximating the solution to Equation 2.8 using a feed-forward, multilayer neural network $u(x, t) \approx \mathcal{N}(x, t; \theta)$ and define a physics-informed neural network $\mathcal{F}$ by

$$\mathcal{F} := \mathcal{N}_t + \mathcal{O}[\mathcal{N}; \lambda] - f. \quad (2.9)$$

Figure 1 shows the network diagram for a general PINN. Note that $\mathcal{F}$ and $\mathcal{N}$ will have the same set of trainable network parameters which can be learned by minimizing a mean squared error loss. To define this loss, we will consider N_{∂} many initial/boundary points $\{x_{\partial}^i, t_{\partial}^i, g^i\}_{i=1}^{N_{\partial}}$ where $g^i = g(x_{\partial}^i, t_{\partial}^i)$

for each $1 \leq i \leq N_\partial$ and a set of N_Ω internal collocation points $\{x_\Omega^i, t_\Omega^i\}_{i=1}^{N_\Omega}$. Then for a set of N_{data} points given by $\{x^i, t^i, u^i\}_{i=1}^{N_{\text{data}}}$ we define the mean squared loss as

$$MSE_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} |\mathcal{N}(x^i, t^i) - u^i|^2 \quad (2.10a)$$

$$MSE_\partial = \frac{1}{N_\partial} \sum_{i=1}^{N_\partial} |\mathcal{B}[\mathcal{N}](x_\partial^i, t_\partial^i) - g^i|^2 \quad (2.10b)$$

$$MSE_\Omega = \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} |\mathcal{F}(x_\Omega^i, t_\Omega^i)|^2 \quad (2.10c)$$

$$MSE = \omega_d MSE_{\text{data}} + \omega_B MSE_\partial + \omega_F MSE_\Omega \quad (2.10d)$$

where $\omega_d, \omega_B, \omega_F$ are, respectively, penalty weights associated with enforcement of the data, boundary, and differential conditions. The penalty terms can be learned by the network or set manually. Solving Equation 2.8 is done by minimizing Equation 2.10d with respect to the network parameters θ . That is, we employ a gradient descent algorithm to approximate the set of solution parameters θ^* such that

$$\theta^* = \arg \min_{\theta} MSE. \quad (2.11)$$

This results in the network $\mathcal{N}$ becoming an approximate solution to Equation 2.8. Moreover, the PINN given by Equation 2.9 includes the velocity term $\mathcal{N}_t$, so the resulting solution is continuous in time as well as in space. The data-driven solution can actually be carried out in the absence of any labelled data. In the event that $\{u^i\}_{i=1}^{N_{\text{data}}} = \emptyset$, the data-driven solution problem reduces to an unsupervised learning task [16]. This is one of the small differences to consider when dealing with data-driven discovery which requires that we have training data.

2.3.2 Data-Driven Discovery. The setup of a data-driven discover PINN is nearly the same as in subsection 2.3.1 but with a minor change. Instead of knowing the parameters λ of the nonlinear differential operator, we establish them as trainable network weights in Equation 2.9. This means that $\mathcal{N}$ and $\mathcal{F}$ no longer share the same set of trainable parameters as $\mathcal{F}$ is composed of the parameter set $\theta' = \theta \cup \lambda$. In order for the network to learn the operator weights we must have $\{u^i\}_{i=1}^{N_{\text{data}}} \neq \emptyset$, ideally with N_{data} sufficiently large so that the network accurately learns the operator parameters. This is all that is different between the two problem set ups. The collocation and boundary points are the same in each case and so is the definition of the loss function MSE . It is worth noting that the inverse problem of inferring system parameters is just as simple to set

up as the forward problem of solution approximation. For both the solution and the discovery class of problems we have used a continuous-time formulation. While this formulation is fine for a small number of spatial dimensions, it can cause a significant bottleneck in higher dimensional problems as global enforcement of the PDE condition would require an exponential increase in collocation points [3]. The PINN framework circumvents this issue by introducing numerical quadrature along the time axis.

2.3.3 Discrete Time Model. For dynamic problems like Equation 2.8, PINNs may implement either continuous or discrete time evolution. The continuous model arises when we include a loss term approximating the velocity u_t . While this results in a solution that is continuous in time, the continuous model can be challenging to train. Alternatively, we can use a more traditional approach and discretize the time-axis. Performing a time discretization leads to a solution that will be continuous in space but discrete in time. Moreover, we show how PINNs can exploit implicit Runge-Kutta methods to perform large time steps without accumulating temporal error. Let $r > 0$ and consider the r -stage general Runge-Kutta discretization of Equation A.1.3 defined for each $i = \{1, 2, \dots, r\}$ at time t_η

$$u_i^\eta := u^{\eta+c_i} - \Delta t \sum_{j=1}^r a_{ij} \mathcal{O}[u^{\eta+c_j}], \quad (2.12a)$$

$$u_{r+1}^\eta := u^{\eta+1} - \Delta t \sum_{j=1}^r b_j \mathcal{O}[u^{\eta+c_j}]. \quad (2.12b)$$

This approach uses two sub-networks. The first of which uses input $x \in \mathbb{R}$ to approximate a vector of evaluations for u at each temporal RK-stage. Let $\mathcal{N}^\eta$ be a neural net defined by

$$\mathcal{N}^\eta(x; \theta) \approx \begin{bmatrix} u^{\eta+c_1}(x) \\ u^{\eta+c_2}(x) \\ \vdots \\ u^{\eta+c_r}(x) \\ u^{\eta+1}(x) \end{bmatrix}. \quad (2.13)$$

The second sub-net uses Equation 2.13 as its input along with Equation 2.12 to define a network encoding the PDE derivative conditions. To simplify notation, take $\mathbf{A} = (a_{ij})$, $\mathbf{b} = (b_i)$, and $\mathbf{c} = (c_i)$.

Define the network $\mathcal{H}$ and composite network $\hat{\mathcal{N}}$ by

$$\mathcal{H}[\mathcal{N}^\eta](x; \theta') = \mathcal{N}^\eta(x) - \Delta t \begin{bmatrix} \mathbf{A} \\ \mathbf{b}^\top \end{bmatrix} \mathcal{O}[\mathcal{N}^\eta](x), \quad (2.14)$$

$$\hat{\mathcal{N}} = \mathcal{H}[\mathcal{N}^\eta]. \quad (2.15)$$

Let $\{x^k, u^{\eta,k}\}_{k=1}^{N_\eta}$ be system data associated with time t_η and take enforced boundary points $\{x_\partial^k\}_{k=1}^{N_\partial}$. Then the network parameters common to $\mathcal{N}$ and $\mathcal{H}$ can be learned by minimizing the sum of square errors

$$SSE_\eta = \sum_{j=1}^{r+1} \sum_{i=1}^{N_\eta} |\hat{\mathcal{N}}_j^\eta(x^i) - u^{\eta,i}|^2, \quad (2.16a)$$

$$SSE_b = \sum_{i=j}^{r+1} \sum_{i=1}^{N_\partial} |\mathcal{B}[\hat{\mathcal{N}}_j^\eta](x_\partial^i) - g(x_\partial^i)|^2, \quad (2.16b)$$

$$SSE = SSE_\eta + SSE_b. \quad (2.16c)$$

This formulation may seem a bit odd at first. The neural net is set up so that we can use the known solution at each time step (starting with the initial condition) to train the intermediate network to approximate the solution along the Runge-Kutta quadrature points in time. A network diagram is given in Figure 3 that shows how the RK PINN is connected. Because the new network is trained only on the known solution at the current time we no longer need any collocation points and $MSE_{\mathcal{F}}$ is dropped from the loss function. It is important to note that this formulation has a distinct advantage over classical Runge-Kutta methods. Namely, the number of RK stages can be increased without significantly increasing the training cost of the network. For each additional stage we just need to add a neuron to the output layer and include its associated network parameters. So the network parameters will only increase linearly with the number of Runge-Kutta stages [3], [15].

Note that if we choose $\mathbf{A}, \mathbf{b}, \mathbf{c}$ according to the Gauss-Legendre quadrature and Equation A.1.5 then the RK method is an implicit method with truncation error $\varepsilon = \mathcal{O}(\Delta t^{2r})$ for time step Δt and an r -stage method. This gives us a standard way of determining how many stages are necessary to bound the numerical approximation by some tolerance ε when Δt is known. That is, we may choose the number of RK stages by

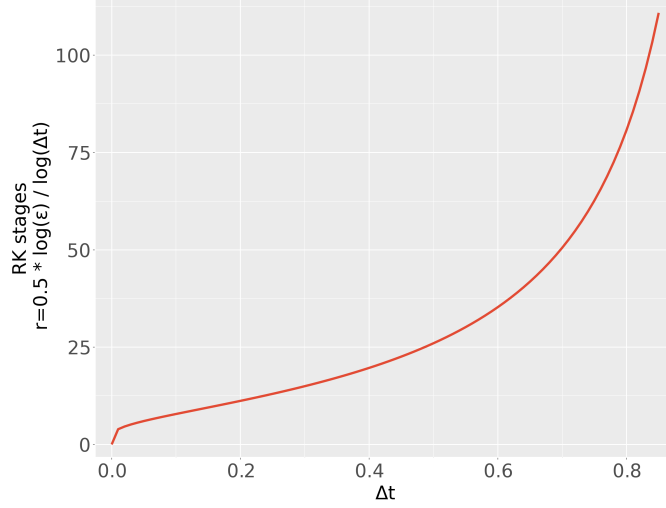


Fig. 2 Number of Runge-Kutta stages necessary to bound the temporal error accumulation (for GLRK) below machine precision $\varepsilon = 2.220e - 16$.

$$r = \frac{1}{2} \frac{\log \varepsilon}{\log (\Delta t)}. \quad (2.17)$$

and since the Gauss-Legendre RK method is implicit and therefore unconditionally stable we may use large time steps and only need concern ourselves with the method’s accuracy. Though the number of stages becomes unreasonable close to $\Delta t = 1$, we are permitted to take a time step of $\Delta t = 0.85$ and only need $r = 110$ to ensure the time-component of error is below ε . This means that the DL framework is capable of utilizing arbitrarily high order implicit RK methods for a relatively small cost. Considering the recent advancements that have been made in computing the GL quadrature weights and nodes [17], [18], along with the presentation of simple forms for the RK coefficient matrices [19], this particular feature of the DL method shows great potential as a numerical tool in scientific computing.

The use of the implicit RK method in tandem with PINNs suggests that hybridizing discrete and DL methods could yield powerful numerical tools. It is not yet clear how effective the hybridized approach is but we are compelled to investigate further. It seems natural to then wonder if the PINN framework could benefit from hybridization with other traditional numerical techniques.

2.4 Alternate PINN Formulations

Since the PINN was proposed in [3] there has been a massive uptick in publications relating to deep neural network applications in solving PDE [16], [20]. PINNs have been implemented in

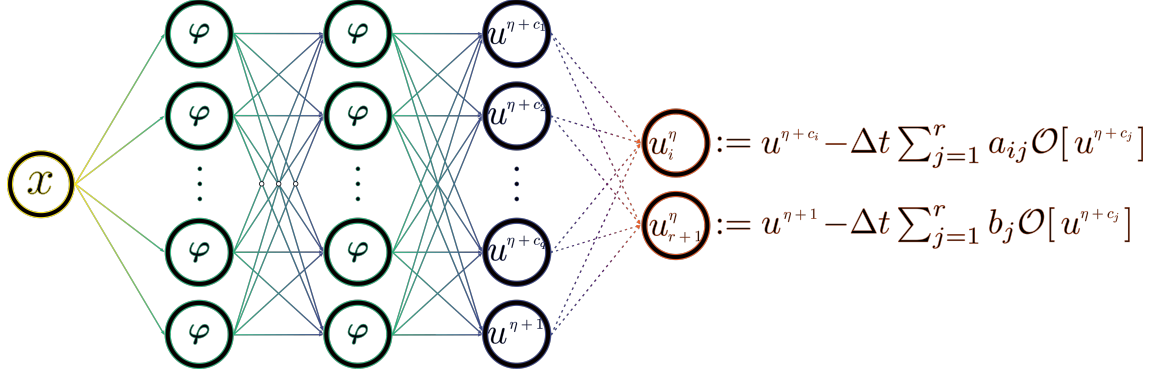


Fig. 3 A physics-informed neural net diagram for a discrete-time model using a general Runge-Kutta method. Dashed connections represent non-trainable network parameters.

solutions for a variety of physical problems like Navier-Stokes [21]–[23], convection heat transfer [24], solid mechanics, [25], [26] and the Euler equations [27]. Many of these studies apply the PINN framework on a very specific class of problems that range in complexity. We are concerned with PINNs as a framework so rather than delving too far into specific problems we consider a more general approach that utilizes alternate forms of the PINN framework.

Though the standard PINN outlined in section 2.3 is the most commonly used formulation, there are a number of modified approaches that draw inspiration from more traditional numerical methods. While these alternate formulations may provide some computational advantage[28]–[30] they also highlight the flexibility of the deep neural network model and how it can be effectively hybridized with traditional numerical methods.

2.4.1 Variational Form (vPINN). The vPINN, presented in [28], draws inspiration from the Petrov-Galerkin method for finite element approximations for solving PDE. We proceed by defining test and trial function spaces and using integration-by-parts to reduce the differential order of the governing problem [31], [32]. Consider the following initial boundary value problem,

$$\mathcal{O}[\mathcal{N}; \lambda](\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega \quad (2.18a)$$

$$\mathcal{B}[\mathcal{N}](\mathbf{x}) = h(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad (2.18b)$$

where $\mathcal{O}$ is a (potentially) non-linear differential operator with parameters λ , $\mathcal{B}$ a boundary operator, and f some external forcing function. Equation 2.18 is known as the strong-form of the mathematical problem. By considering an appropriate test function $\phi(\mathbf{x})$, we may multiply Equation 2.18a by ϕ

and integrate over Ω to get

$$(\mathcal{O}[\mathcal{N}; \lambda](\mathbf{x}), \phi(\mathbf{x}))_{\Omega} = (f(\mathbf{x}), \phi(\mathbf{x}))_{\Omega} \quad (2.19a)$$

$$\mathcal{B}[\mathcal{N}](\mathbf{x}) = h(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \quad (2.19b)$$

where we use $(\cdot, \cdot)_{\Omega}$ to denote the usual inner product of functions on Ω . Though the boundary residual term is essentially the same as it is in the standard PINN formulation, the differential residual is no longer trained on collocation points within the domain. Instead we consider a collection $\Phi = \{\phi_k, k = 1, \dots, K\}$ of admissible test functions and construct the residuals

$$MSE^{\nu} = \frac{1}{K} \sum_{k=1}^K |(\mathcal{O}[\mathcal{N}], \phi_k)_{\Omega} - (F, \phi_k)_{\Omega}|^2, \quad (2.20a)$$

$$MSE_{\partial} = \tau \frac{1}{N_{\partial}} \sum_{i=1}^{N_{\partial}} |\mathcal{B}[\mathcal{N}](\mathbf{x}_{\partial}^i) - h(\mathbf{x}_{\partial}^i)|^2, \quad (2.20b)$$

$$MSE = MSE^{\nu} + MSE_{\partial}, \quad (2.20c)$$

where $\{\mathbf{x}_{\partial}^i\}_{i=1}^{N_{\partial}}$ is a set of boundary points of Ω and τ is a penalty term. The penalty term helps enforce data at the boundary points and is typically chosen based on the problem but can also be included in the set of learned parameters[28]. It is worth noting that the integration required in this approach only has an analytic representation for very simple networks and requires traditional numerical integration for deep neural networks. Note that this formulation changes the space of training data. Here one must define a set of domain points to be fixed as quadrature points while the variational PINN is trained using test functions. The spatial integral reduces the order of the PDE by one, allowing for the network to be more easily trained.

Though I have not seen it mentioned in any literature thus far, the variational problem allows us yet another option for handling the boundary conditions. One can build the B.C into the network(hard enforcement) or you can include it as a term in the loss function (soft enforcement). Because we are solving a variational problem over a space of test functions, the original boundary conditions of the problem may be enforced *essentially* or *naturally* depending on the quantity enforced by the B.C. Since we are approximating the solution using a space of functions we can choose to use functions which inherently satisfy the boundary conditions. The *essential* b.c are those which are imposed explicitly on the solution while *natural* b.c are imposed on solution derivatives.

As an example, suppose we have the following boundary value Poisson problem:

$$-\Delta u = f \quad \text{on } \Omega, \quad (2.21a)$$

$$u = 0 \quad \text{on } \Gamma \subset \partial\Omega, \quad (2.21b)$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } \partial\Omega \setminus \Gamma, \quad (2.21c)$$

where $\frac{\partial u}{\partial n}$ is the derivative of u in the direction normal to the boundary $\partial\Omega$. Then we construct a variational space to incorporate the essential boundary condition Equation 2.21b

$$V := \{v \in H^1(\Omega) : v|_{\Gamma} = 0\} \quad (2.22)$$

where $H^1(\Omega)$ is the space of L_2 functions on Ω whose first order derivatives are also in L_2 . Additionally, $v|_{\Gamma}$ is interpreted using the trace theorem in $L_2(\partial\Omega)$, so we consider it to mean $v \cdot \chi_{\Gamma} = 0$ where χ is the usual characteristic function. With the choice of test space V , the *natural* boundary condition Equation 2.21c is automatically met for a $u \in V$ satisfying the variational form $a(u, v) = (f, v)$ for every $v \in V$ [33], [34].

Then the loss function consists solely of the term MSE^{ν} and boundary conditions are enforced exactly. Though this approach requires more manual overhead for the user, we can see how the variational form along with function space theory can reduce the cost of training the network. The drawback of this approach lies in establishing a numerical quadrature and being able to precisely choose a valid test space of functions.

2.4.2 Conservative Form (cPINN). cPINNs, presented in [29], partitions the computational domain and defines a separate PINN on each sub-domain. Each PINN is tasked with approximating a solution locally on the sub-domain while preserving global requirements through interface conditions.

Recall the general IBVP given by Equation 2.8

$$\begin{aligned} u_t + \mathcal{O}[u; \lambda] &= 0, \quad (x, t) \in \hat{\Omega} \\ \mathcal{B}[u] &= g, \quad (x, t) \in \hat{\Gamma} \subseteq \partial\hat{\Omega} \end{aligned}$$

where this formulation refers to $\mathcal{O}$ as a nonlinear flux parametrized by λ . On each sub-domain we define a neural net training problem constrained by the IBVP for the system along with a conservative flux at sub-domain interfaces which maintain global conditions.

Let on sub-domain p we let $\mathcal{N}_p(x, y, t; \theta) \approx u(x, y, t)$ and define the local loss function

$$MSE_{\partial p} = \frac{1}{N_{\partial p}} \sum_{i=1}^{N_{\partial p}} |g^i - \mathcal{B}[\mathcal{N}_p](x_{\partial p}^i, y_{\partial p}^i, t_{\partial p}^i)|^2 \quad (2.23a)$$

$$MSE_{\mathcal{F}_p} = \frac{1}{N_{\mathcal{F}_p}} \sum_{i=1}^{N_{\mathcal{F}_p}} |\mathcal{F}_p(x_{\mathcal{F}_p}^i, y_{\mathcal{F}_p}^i, t_{\mathcal{F}_p}^i)|^2 \quad (2.23b)$$

$$MSE_I = \frac{1}{N_I} \sum_{i=1}^{N_I} |\mathcal{O}_p[\mathcal{N}_p](x_I^i, y_I^i, t_I^i) \cdot \mathbf{n} - \mathcal{O}_p^+[\mathcal{N}_p](x_I^i, y_I^i, t_I^i) \cdot \mathbf{n}|^2 \quad (2.23c)$$

$$MSE_{\text{avg}} = \frac{1}{N_I} \sum_{i=1}^{N_I} |\mathcal{N}_p(x_I^i, y_I^i, t_I^i) - \{\mathcal{N}(x_I^i, y_I^i, t_I^i)\}|^2 \quad (2.23d)$$

$$MSE_p = \omega_{\partial} MSE_{\partial p} + \omega_{\mathcal{F}} MSE_{\mathcal{F}_p} + \omega_I MSE_I + \omega_a MSE_{\text{avg}} \quad (2.23e)$$

where $\mathcal{F}$ is the residual for the governing PDE, $\mathcal{O}_p \cdot \mathbf{n}$ and $\mathcal{O}_p^+ \cdot \mathbf{n}$ are interface fluxes across subdomains. $N_{\partial p}$, $N_{\mathcal{F}_p}$, and N_I are the number of boundary/initial data, number of collocation points, and the number of interface points, respectively, within sub-domain p . Additionally, the average of $\mathcal{N}$ is

$$\{\mathcal{N}\} = \mathcal{N}_{\text{avg}} := \frac{\mathcal{N}_p + \mathcal{N}_p^+}{2} \quad (2.24)$$

This formulation allows for meticulous control over the total network. Since each subnetwork only interacts with its neighbors, training the network is very amenable to parallelization and would only require a small number of data transfers between neighboring networks. Additionally, each subnetwork need not be the same. Then shallower networks could be used in areas where the solution is well-behaved and more complex networks used on less behaved regions of the domain. Individual networks on each subdomain also increase the network expressivity while the flux conditions at interfaces reduce error propagation across subdomains [29].

2.5 Parametric PDE

Up until now we have been primarily concerned with solutions for IBVP in which every parameter of the problem is fixed and the resulting numerical output can only describe the fixed-state system. Instead, we may wish to solve the PDE over a range of possible scenarios (such as varying boundary/initial conditions, operator parameters, or domain geometries) so that our trained

network represents a family of PDE solutions. This generalized problem is referred to as a parametric PDE and it cannot be solved using the traditional finite difference/element methods. Because traditional numerical methods would require a simulation to be carried out for every single scenario, the computational cost (along with the need to store each solution) quickly becomes prohibitive. While it is possible to adjust the PINNs framework for this task by introducing new input variables, this results in a high-dimensional PDE that will drive up the computational cost of higher-order derivatives in the loss function [35]. A number of techniques known as Neural Operator Methods have been proposed for this kind of problem [36]–[38], but there exists a fairly simple architecture which is established using a universal approximation theorem for operators.

2.5.1 DeepONet. The 2020 neural net architecture, DeepONet[39], employs a lesser known universal approximation theorem that established neural networks as universal approximators of functionals and operators[40], [41]. Like with Neural Operator methods, this shift in learning focuses on the approximation of infinite-dimensional function spaces rather than a finite-dimensional map between Euclidean spaces. We outline the approach in this section.

Consider the operator G with input function u . Then, for y in the domain of $G(u)$, $G(u)(y)$ is a real number. The DeepONet architecture consists of two sub-networks each trained on a separate space of inputs. The *branch* network is trained on different function inputs while the *trunk* network is trained on inputs from the PDE’s Euclidean domain. However, we cannot simply train a new network on function inputs but rather must train the network on the image of said inputs. To this end, we define a set of *sensor* points $\{\xi_i\}_{i=1}^s$ which are to remain fixed across different function inputs. Then the *branch* network $\mathcal{N}_B : \mathbb{R}^s \rightarrow \mathbb{R}^p$ is defined as

$$\mathcal{N}_B([u(\xi_1), \dots, u(\xi_s)]^T; \theta_B) = [b_1, \dots, b_p]^T, \quad (2.25)$$

and the *trunk* network $\mathcal{N}_T : \mathbb{R}^n \rightarrow \mathbb{R}^p$ is defined as

$$\mathcal{N}_T(y; \theta_T) = [t_1, \dots, t_p]^T. \quad (2.26)$$

The two networks are combined by using the output of the branch network as weights to the output of the trunk network in a straightforward linear combination the defines an operator network $\mathcal{N}_\theta$ by

$$\mathcal{N}_\theta[u](y) = \mathcal{N}_B(u(\boldsymbol{\xi}); \theta_B) \cdot \mathcal{N}_T(y; \theta_T) = \sum_{i=1}^p b_i t_i \quad (2.27)$$

where $\theta = \theta_B \cup \theta_T$. This formulation is shown to exhibit an approximation error which is dependent on the number of fixed sensors, $\{\xi_i\}_{i=1}^s$, and input function type while also demonstrating a reduction in generalization error when compared to a standard fully-connected network [39]. Moreover, [42] demonstrates how the DeepONet architecture can be used alongside PINNs to train an operator across various initial and boundary conditions. This may allow for the construction of persistent and efficient families of solutions for PDE. To develop the appropriate loss function let $\{u^i\}_{i=1}^N$ be a collection of N separate input functions and for each u^i take $\{y_{\partial j}^i\}_{j=1}^P$ to be P locations determined from data, initial, or boundary conditions and $\{y_{\Omega j}^i\}_{j=1}^Q$ to be a set of collocation points sampled from the domain of $G[\mathbf{u}^i]$. Here we use $\mathbf{u}^i = [u^i(\xi_1), \dots, u^i(\xi_s)]^T$. Then the relevant loss function is given by

$$\mathcal{L}_{\partial}(\theta) = \frac{1}{NP} \sum_{i=1}^N \sum_{j=1}^P \left| \mathcal{B}[G[\mathbf{u}^i]](y_{\partial j}^i) - u^i(y_{\partial j}^i) \right|^2, \quad (2.28a)$$

$$\mathcal{L}_{\Omega}(\theta) = \frac{1}{NQs} \sum_{i=1}^N \sum_{j=1}^Q \sum_{k=1}^s \left| \mathcal{O}[G_{\theta}[\mathbf{u}^i]](y_{\Omega j}^i) \right|^2, \quad (2.28b)$$

$$\mathcal{L}(\theta) = \mathcal{L}_{\partial}(\theta) + \mathcal{L}_{\Omega}(\theta). \quad (2.28c)$$

Figure 4 gives a network connectivity diagram for a physics-informed deep operator network formulated across multiple initial/boundary data functions.

2.6 Approximation Theory

Statistical Learning Framework for PINN Error Analysis. Consider $\{x_i\}_{i=1}^N$ collocation points of $\bar{\Omega} = \Omega \cup \partial\Omega$.

Let $\mathcal{N}_{\theta}$ be a neural net approximation (realized with parameters θ) approximating a solution to Equation 2.8. Define the residuals

$$\mathcal{R}_{\text{PDE}}[v] = v_t + \mathcal{O}[v] - f \quad (2.29)$$

$$\mathcal{R}_{\text{bdry}}[v] = \mathcal{B}[v] - g \quad (2.30)$$

The risk for a PINN can be defined as

$$\begin{aligned} E_G^2(\theta) &= \int_{\bar{\Omega}} |\mathcal{R}_{\text{PDE}}[\mathcal{N}_{\theta}](x, t)|^2 dx dt \\ &\quad + \int_{\partial\Omega} |\mathcal{R}_{\text{bdry}}[\mathcal{N}_{\theta}](x, t)|^2 dx dt. \end{aligned} \quad (2.31)$$

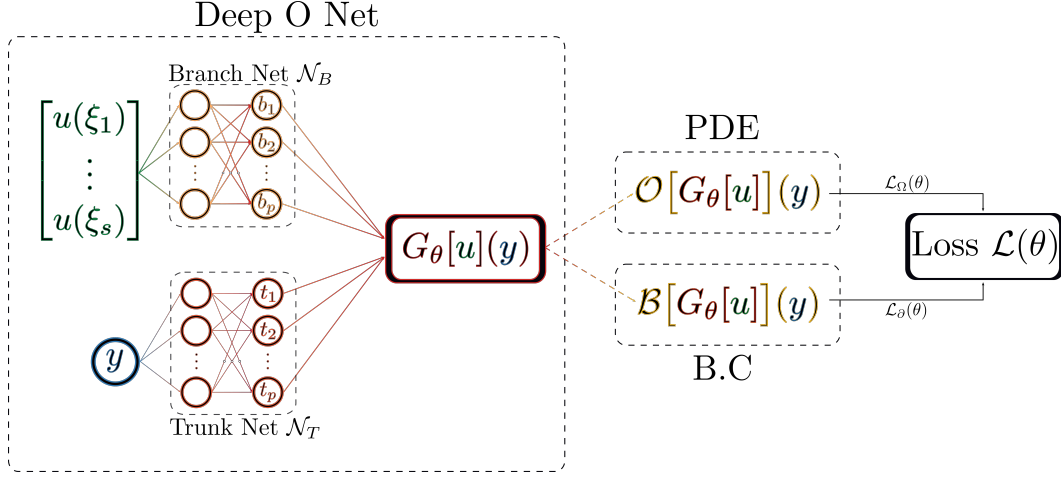


Fig. 4 Presents the network connectivity graph for a physics-informed deep operator network formulated to learn the PDE solution operator across various initial and boundary data functions. The DeepONet consists of a branch network $\mathcal{N}_B : \mathbb{R}^s \rightarrow \mathbb{R}^p$ and trunk network $\mathcal{N}_T : \mathbb{R}^n \rightarrow \mathbb{R}^p$. The branch network $\mathcal{N}_B$ is trained using input functions evaluated along a fixed set of sensors $\{\xi_i\}_{i=1}^s$ while the trunk network $\mathcal{N}_T$ is trained on the spatio-temporal domain $\mathbb{R}^n$ of the PDE. Dashed connection signify non-trainable parameters. This figure is a recreation of Fig. 1 in [42].

While we cannot compute the risk, we can approximate it using a discrete integral. The empirical risk is given by

$$E_T^2(\theta) = \frac{1}{N} \sum_{i=1}^N |\mathcal{R}_{\text{PDE}}[\mathcal{N}_\theta](x_i, t_i)|^2 + \frac{1}{N_\partial} \sum_{i=1}^{N_\partial} |\mathcal{R}_{\text{bdry}}[\mathcal{N}_\theta](x_i, t_i)|^2. \quad (2.32)$$

Here, the training error E_T approximates the generalization error E_G by a simple 'rectangular' quadrature. Let θ^* be the final parameter set achieved by training the PINN. Then the optimization error $\mathcal{E}_O$ is given by

$$\mathcal{E}_O := E_T^2(\theta^*) - \inf_{\theta \in \Theta} E_T^2(\theta) \quad (2.33)$$

which measures the empirical risk of the final approximation against the smallest attainable empirical risk. Next, measuring the separation between risk and empirical risk we define $\mathcal{E}_G$ by

$$\mathcal{E}_G := \sup_{\theta \in \Theta} |E_G^2(\theta) - E_T^2(\theta)|. \quad (2.34)$$

Some of the literature involving PDE refers to $\mathcal{E}_G$ as the generalization error but I have opted to use generalization error as it is used in ML. Namely, E_G is the generalization error and $\mathcal{E}_G$ the risk

approximation error. Lastly, the approximation error $\mathcal{E}_A$ given by

$$\mathcal{E}_A := \inf_{\theta \in \Theta} E_G^2(\theta). \quad (2.35)$$

These errors give rise to an upper bound on the global error between a trained network and the exact solution. The global error bound

$$E_G^2(\theta^*) \leq \mathcal{E}_O + 2\mathcal{E}_G + \mathcal{E}_A \quad (2.36)$$

also indicates how each error term is vital in constraining the global error and define key research areas to improve mathematical applications for deep neural networks [16], [43]. While standard DL models can take advantage of ReLU activation functions to establish bounds on approximation error $\mathcal{E}_A$ [44], solving PDE requires networks (and by extension activation functions) which are sufficiently differentiable. While [45] presents an error analysis framework for PINNs when applied to linear PDE, a more general bound on approximation error can be established on Sobolev spaces [16], [46].

2.6.1 Approximation Error $\mathcal{E}_A$. While universal approximation theorems Theorem 1 and Theorem 2 can help identify a minimal network size for universal approximation they do not provide any sufficiency conditions to guarantee accuracy of the network. Specifying bounds on approximation error requires one to choose an appropriate network architecture and error is typically quantified as an increase to network complexity as $\epsilon \rightarrow 0$ (asymptotic approximation rates). However, we instead would like to know how large a network must be to guarantee a specified accuracy will be reached. Such an error bound has recently been proposed for networks with two hidden layers which utilize hyperbolic tangent activation functions [46]. The choice to use a shallow but wide network is backed by empirical observation that shallow/wide networks perform better on scientific computing problems where training data may be scarce [47]. To give the approximation theorem we first define $R > 0$ such that $|\varphi^{(m)}|$ is decreasing on $[R, \infty)$ for every $1 \leq m \leq k$ and define

$$|P_n, d| = \binom{n+d-1}{n}. \quad (2.37)$$

Theorem 4 (Sobolev approximation for tanh networks [46])

Let $d, s \in \mathbb{N}, R > 0, \delta > 0$ and $f \in W^{s,\infty}([0,1]^d)$. There exist constants $c(d, k, s, f), N_0(d) > 0$, such that for every $N \in \mathbb{N}$ with $N > N_0(d)$ there exists a tanh neural network $\hat{f}^N$ with two hidden layers, one of width at most $3\lceil \frac{s}{2} \rceil |P_{s-1,d+1}| + d(N-1)$ and another of width at most $3\lceil \frac{d+2}{2} \rceil |P_{d+1,d+1}| N^d$,

such that

$$\|f - \hat{f}^N\|_{L^\infty([0,1]^d)} \leq (1 + \delta) \frac{c(d, 0, s, f)}{N^s}, \quad (2.38)$$

and for $k = 1, \dots, s - 1$,

$$\begin{aligned} \|f - \hat{f}^N\|_{W^{k,\infty}([0,1]^d)} &\leq 3^d (1 + \delta) (2(k+1))^{3k} \\ &\quad \times \max\{R^k, \ln^k(\beta N^{s+d+2})\} \\ &\quad \times \frac{c(d, k, s, f)}{N^{s-k}} \end{aligned} \quad (2.39)$$

where we define

$$\beta = \frac{k^3 2^d \sqrt{d} \max\{1, \|f\|_{W^{k,\infty}([0,1]^d)}^{\frac{1}{2}}\}}{\delta \min\{1, \sqrt{c(d, k, s, f)}\}}. \quad (2.40)$$

If $f \in C^s([0,1]^d)$, then it holds that $N_0(d) = \frac{3d}{2}$ and

$$c(d, k, s, f) = \max_{0 \leq \ell \leq k} \frac{1}{(s - \ell)!} \left(\frac{3s}{2}\right)^{s-\ell} |f|_{W^{s,\infty}([0,1]^d)}, \quad (2.41)$$

otherwise $N_0(d) = 5d^2$ and

$$c(d, k, s, f) = \max_{0 \leq \ell \leq k} \frac{\pi^{\frac{1}{4}} \sqrt{s}}{(s - \ell - 1)!} (5d^2)^{s-\ell} |f|_{W^{s,\infty}([0,1]^d)}, \quad (2.42)$$

The weights of $\hat{f}^N$ scale as

$$O\left(c(d, k, s, f)^{-s/2} N^{d(d+s^2+k^2)/2} (s(s+2))^{3s(s+2)}\right). \quad (2.43)$$

Equation 4 provides explicit approximation bounds characterized by d , problem dimension, k , order of the approximation Sobolev space, s , order of the target Sobolev space, f , the target function, and N , the number of training points. This result pushes back against the idea that depth is a necessary feature for highly expressive networks. Additionally, the theorem extends to a wider class of activation functions including logistic functions. However, approximation error is only part of the total error and more work is required to bound the generalization error.

2.6.2 Generalization Error $\mathcal{E}_G$. Recent work has provided bounds on PINN generalization error for linear second-order PDE [48] (later extended to all linear problems [45]) and some specific cases like Navier-Stokes[49]. Of particular note, however, is the abstract framework for PINNs in which generalization error E_G is estimated by the training error E_T and number of training points for both forward[50] and inverse[51] problems. Moreover, the abstract framework leverages stability of PDE to provide conditions under which generalization error is small whenever

training error is small. The formulation of the bounds on generalization error is quite intricate and requires substantial use of functional analysis. This falls outside of the scope of this study but we include this section as it is important in the development of PINNs.

2.7 Future Research Areas

2.7.1 Mesh-Free Integration. It may be useful to have a DL process for integrating functions over domains in higher dimensions. This issue briefly appears in subsection 2.4.1 where the variational form gives rise to a numerical quadrature, which becomes numerically intractable for high dimensional problems[52]. I have not encountered any literature directly addressing this but I have seen some mention that the numerical quadrature is necessary unless we get access to DL integral methods [28]. However, in solving differential equations we have already seen that a mesh-free integral technique may be plausible. This is because constraining network derivatives results in a somewhat implicit computation of an antiderivative. To be explicit, consider the function $f(x, y) = \cos(x^2 - y^2)$ for which we wish to compute the integral

$$\int_{-1}^1 \int_{-1}^1 f(x, y) dx dy = \int_{-1}^1 \int_{-1}^1 \cos(x^2 - y^2) dx dy. \quad (2.44)$$

Let $\mathcal{N}(x, y; \theta_1)$, $\mathcal{M}(y; \theta_2)$ each be Feed-forward deep neural networks with parameters θ_1 and θ_2 respectively. The idea here is to approximate an antiderivative of f (w.r.t x) using $\mathcal{N}$ and then use $\mathcal{M}$ to approximate the antiderivative of $\mathcal{N}$ (w.r.t y). Let $\{(x_i, y_i)\}_{i=1}^N$ be collocation points chosen from $[-1, 1] \times [-1, 1]$ and define the loss function

$$\mathcal{L}_{\partial x}(\theta) = \frac{1}{N} \sum_{i=1}^N |\mathcal{N}_x(x_i, y_i) - f(x_i, y_i)|^2, \quad (2.45a)$$

$$\mathcal{L}_{\partial y}(\theta) = \frac{1}{N} \sum_{i=1}^N |\mathcal{M}_y(y_i) - F^x(y_i)|^2, \quad (2.45b)$$

$$\mathcal{L}(\theta) = \mathcal{L}_{\partial x}(\theta) + \mathcal{L}_{\partial y}(\theta), \quad (2.45c)$$

where $\theta = \theta_1 \cup \theta_2$ and $F^x(y) = \mathcal{N}(1, y) - \mathcal{N}(-1, y)$ computes the internal integral w.r.t the x variable. The top row of Figure 5 gives the exact integrand plot along with $\mathcal{N}_x \approx f$ and the error between the two while the bottom row plots $\mathcal{M}_y$ and F^x along the y -axis to track how well the internal integral is approximated. By minimizing Equation 2.45c we train the network $\mathcal{M}$ to

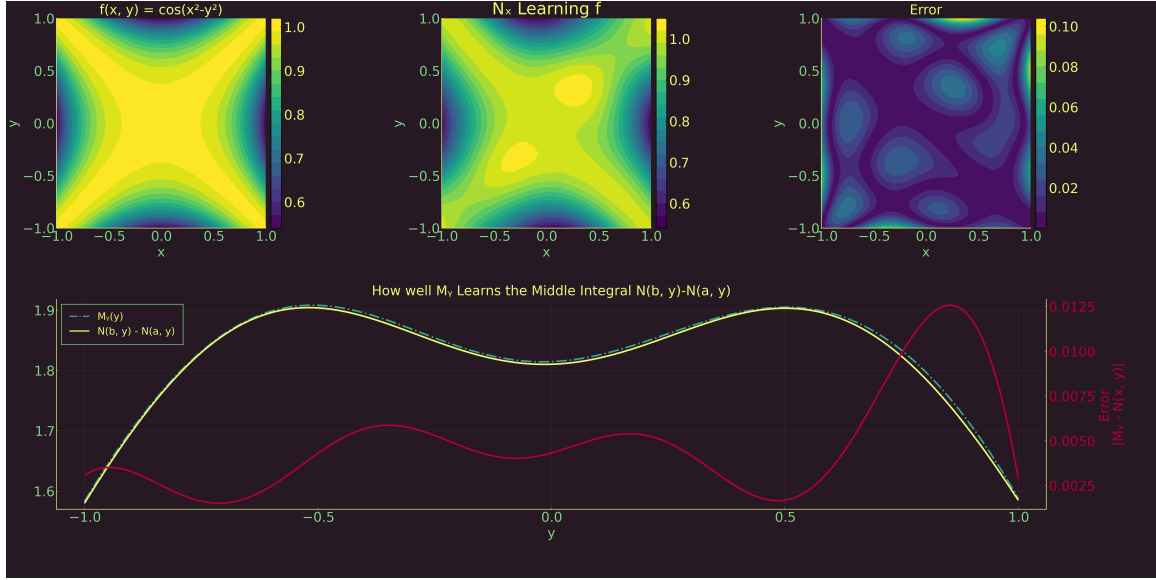


Fig. 5 For networks $\mathcal{N}$, and $\mathcal{M}$, each having 2 hidden layers with 25 neurons, approximate the integral of $\cos(x^2 - y^2)$ over the 2D domain $[-1, 1] \times [-1, 1]$ by minimizing Equation 2.45c using stochastic gradient descent. The above figure tracks intermediate training goals but method error is determined by how well the integral is approximated. This network configuration gave error $\epsilon = 2.207e - 4$.

approximate a mixed antiderivative and thus the integral

$$\int_{-1}^1 \int_{-1}^1 f(x, y) dx dy \approx \mathcal{M}(1) - \mathcal{M}(-1). \quad (2.46)$$

Comparing the neural net integral with the known exact integral yields an error $\epsilon = 2.207e - 4$.

This approach is fairly straight forward and easily generalizes to integrals over domains of higher dimension. Rather than using a network that is differentiated multiple times it seems to be more efficient to use a separate network to approximate each necessary antiderivative. Additionally, this approach only works on simple, rectangular domains and some care will be needed to extend this mesh-free integration to arbitrary domains. It may be worthwhile to consider a DeepONet architecture as well as it may be necessary to learn a more general antiderivative operator. We believe a careful formulation of a DL technique for integration on generalized domains could be a beneficial tool in computational mathematics. It would provide another avenue of exploration for DL techniques and could allow for a more general class of solutions to PDE.

2.7.2 Initial Guesses for Iterative Methods. The existence of continuous and discrete formulations for time evolution in PINNs leads to two primary approaches for implementing an iterative method accelerator. It has been shown that using an implicit RK method of extremely

high precision in a discrete time model can provide a good initial guess for Newton’s method [19]. The question then is about efficiently learning these initial guesses. For the discrete-time case, we must train the network at every time step and can iterate using Newton’s method once the training process is done. Alternatively, in the continuous-time model the network could be trained across a time interval and results in a solution which is continuous in time. If the continuous approximation happens to provide a good initial guess, then an iterative method could call the continuous PINN at each time step to ensure convergence in a small number of iterations. It is unclear which of these approaches would be more efficient or if either is capable of outperforming a stand alone iterative approach.

2.7.3 Persistent, General Solutions . When creating physical models it is common to identify systems in which various physical models are coupled together to simulate complex dynamical systems. In this case, each coupled system is usually described by its own set of differential equations and the result is a system of coupled PDE. Such problems can become computationally expensive for relatively simple physical systems. The idea behind this area of research is to assess whether the DeepONet solution for parametric PDE can be leveraged to build more efficient coupled physical systems. Because PINN learn solutions to PDE, we believe that training individual systems on a variety of boundary or initial conditions, it may be possible to replace parts of a coupled PDE system with forward passes of a PINN. To illustrate this, consider the incredibly simplified magma reservoir problem. Given a magma chamber that exerts uniform pressure P on the surrounding medium let $\mathcal{F}(P) = \mathbf{u}$ represent the resulting displacements induced by pressure P . Furthermore, suppose that the change in pressure is determined by some change in volume

$$\frac{dP}{dt} = \alpha \frac{dV}{dt} \quad (2.47)$$

This defines a coupled system where the external problem uses a pressure to compute Displacements and, by extension, change in reservoir volume. This change in volume then causes the internal system to change the exerted pressure. So the two systems are coupled along the reservoir boundary and exchange pressure and volume data.

Using the DeepONet architecture, solution for pressure can be learned that is a function of time and $\frac{dV}{dt}$. Then the system evolves from time t_n to time t_{n+1} by

1. $\mathcal{F}(P_n) = \mathbf{u}_n$

2. $\Delta V = \int_{\Gamma} \mathbf{u} \cdot \mathbf{n}$

3. $P_{n+1} = \mathcal{P}(t_{n+1}, \frac{\Delta V}{\Delta t})$

where Γ is the reservoir boundary, $\mathcal{P}$ the DeepONet pressure approximation, and $\Delta t = t_{n+1} - t_n$. This way, it might be possible to use pre-trained networks in coupled systems rather than needing to solve the coupled system all at once.

2.8 Final Remarks

DL shows incredible potential as a tool for numerical solutions to PDE and recent developments show a promising, emerging framework in the form of PINNs. In this study we see how PINNs can be bolstered using traditional numerical methods while also showing how some traditional methods can be augmented with DL to overcome computational challenges. This positions DL as a means to extend traditional methods to problems they could not previously address, and in the case of Parametric PDE we see how DL can perform where mesh-based methods cannot. While DL approaches have typically shown to perform well empirically, the recent progress in PINN approximation theory is beginning to legitimize PINNs as PDE solutions. Being able to predictably bound the error of a method is critical in developing numerical solutions. As PINNs become more reliable tools, we may apply them in unique ways to expand the reach of physical models.

CHAPTER III

MAGMA RESERVOIR

This chapter contains work published in the Journal of Geophysical Research: Solid Earth in September 2022. Co-authors include Dr. Brittany A. Erickson, Dr. Leif Karlstrom, Brian Lee, and Dr. Jay Gopalakrishnan.

Dr. Leif Karlstrom conceptualized the problem and contextualized our findings within the broader volcanology literature. Dr. Jay Gopalakrishnan developed the finite element and time-stepping solvers. Dr. Brittany A. Erickson and I worked together to develop the analytic signal approach to computing the transfer function. Brian Lee worked on defining temperature-dependent viscosity and ran simulations for computing von Mises stress. I was first author on this work and was responsible for non-dimensionalization of the problem, derivation of stable time stepping parameters, model validation via convergence test, and I wrote code to carry out simulations and collect data.

Plain Language Summary

Ground motions associated with subsurface magma reservoirs are the result both of magma movement and time-dependent deformation of crustal rocks. We have developed a new computational framework to help interpret surface deformations associated with magmatic systems embedded within viscoelastic rocks as expected in volcanic regions. This framework is general in the sense that a broad range of scientific studies can be explored by specifying particular conditions at domain boundaries or magma reservoir geometries, and we perform rigorous numerical tests to ensure credible solutions. We then apply the model to study a simple but highly generalizable type of transient behavior - the cyclic pressurization and depressurization of a spherical reservoir. We develop a theoretical approach to simply analyze the time-dependent output, and find that temporal lag and amplification of surface deformation with respect to the reservoir pressure is explained by an aureole of material surrounding the chamber with a dominantly viscous response, whose size is frequency-dependent. Our results can be extended to many transient deformation scenarios because a sinusoidal response forms the basic element of general pressure time-series.

3.1 Introduction

Magma reservoirs represent a fundamental link between mantle melting and volcanic activity seen at the surface. Eruptions that drain these reservoirs are the most dramatic example of magma chamber mechanics, and a wide spectrum of time-varying surface deformation and other unrest

seen in volcanic regions likely has an origin within crustal storage zones [53]–[56]. As a result, understanding controls on time-dependent magma chamber deformation and stress is a long-standing research topic in volcanology [57]–[59]. However, modeling magma reservoir evolution is a challenging problem because time-dependence may arise from a variety of physical processes occurring both internal and external to the magma transport system, many of which leave non-unique signatures in ground deformation patterns.

On sufficiently short time scales, it is appropriate to assume an elastic/brittle rheology of host rocks. Elastic models have been widely used to interpret geodetic data gathered at volcanoes [60]–[62]. Such models predict that time-dependent behavior comes only from reservoir magma mass balance/state variable changes [54] or boundary forcing, although poroelastic effects can also lead to time-dependence [63]. Time dependent deformation and stressing of the reservoir at timescales longer than the onset of viscous creep likely involves ductile response of host rocks e.g., [64]–[66], suggesting an overall viscoelastic rheology.

Viscoelastic effects have been identified as defining a notion of magma chamber stability, providing a mechanism for modulating stresses and deformation associated with pressurization of the chamber [67]–[70]. Viscoelastic effects may play a role in the development of large silicic reservoirs [71] as well as eruption sequences from long-lived magma reservoirs [72] and time-dependent ground deformation at active volcanoes in diverse settings [73]–[77]. On tectonic timescales, transitions in the efficacy of viscous deformation within the transcrustal system may manifest as state shifts in magma transport, such as increasing intrusive-extrusive ratios and spatial organization of volcanism around long-lived centers [78].

Deformation style is strongly tied to the thermal state of the magmatic system, because both rock and magma rheology are temperature dependent. Thus it is to be expected that a viscoelastic response varies spatially, and evolves in time with the transcrustal magma transport system. Such unsteady effects, both spatial and temporal, are poorly constrained. Instead it is typically assumed that magma reservoirs reside in a steady state geotherm [79]–[81], or that the mechanical response is well-approximated by a pre-specified shell of viscous material in an elastic host [68], [72], [82]–[84]. Time evolution is often either imposed kinematically through stress boundary conditions (e.g., to model an eruptive event, e.g., [67]) or arises dynamically through mass and energy balance e.g., [68]. Viscous creep independent of time-variable forcing has also been invoked to explain deformation

signals [83], [85]. Time variation is often idealized due a lack of unique constraints on the source-time function of deformation in magmatic systems, especially on timescales longer than individual unrest or eruptive episodes. General time dependent deformation that does not make inherent assumptions about deep source characteristics has not been widely studied.

Viscoelastic deformation of volcanoes has been studied numerically by numerous authors e.g., [59], [69], [79], [86], [87]. However, we are unaware of a systematic analysis of the numerical and computational issues associated with this problem. As volcanic deformation datasets become better resolved in space and time, and as magma reservoir models are generalized to include more physical processes over an increasing range of timescales, neglecting these numerical and computational considerations is likely to be a major factor limiting scientific progress. In this work, we address two aspects of viscoelastic deformation in magmatic systems. First, we derive and implement a high order numerical modeling framework for simulating transient thermo-mechanical behavior of a subsurface magma reservoir in an isotropic, heterogeneous, viscoelastic domain. Second, we study stress and crustal deformation associated with periodic pressure variation at the chamber wall. This represents a different sort of idealization than previous studies: we consider a spatially resolved mechanical response, but treat time evolution as harmonic. In this way we isolate the frequency dependence of the viscoelastic rheology, and develop a transfer function approach using analytic functions to predict material response. This idealization might approximate some magmatic forcing scenarios, such as cyclic stress from seismic waves, periodic magma injection, or glacial cycles, and we note that quasi-periodic deformation at multiple frequencies has been observed in long-term geodetic timeseries [88]. Our approach also implies a superposition framework for studying much more general time evolution.

Our model is developed to handle general axisymmetric geometries in the subsurface and surface, including lateral loads and topographically complex material interfaces. However, we focus on the relatively simple and well-studied case of a sphere in a half-space without remote loading to explore transient effects, deriving material properties from a steady state temperature distribution within the medium. After detailing the numerical framework we verify convergence using the method of manufactured solutions [89]. Finally we use the verified framework to characterize the system’s response to spatially variable viscoelastic material properties. We develop a transfer function between chamber pressure and maximum vertical surface deformation to

demonstrate that two parameters – the phase lag between pressurization and surface deformation, and their relative amplitude – imply a frequency-dependent viscoelastic response that depends on chamber temperature and geothermal gradient magnitude. We demonstrate that this transfer function permits the reconstruction of complex deformation histories, and show that the spatial thermo-rheologic structure beneath the chamber influences frequency domain expression of surface deformation.

The paper is organized with mathematical and computational details provided first, followed by the spectral (and transfer function) analysis and example calculations. In Section 3.2 we introduce the governing equations and generic physical problem of interest. In Section 3.3 we discuss the computational framework, and develop the specific non-dimensional time-dependent problem of interest. Readers wishing to skip such technical details can go directly to section 3.4, which introduces the transfer function approach that represents our primary analysis tool. Section 3.5 then discusses results of computations and Section 3.6 discusses implications for magmatic systems.

3.2 Mathematical Framework

3.2.1 Problem Formulation and Geometry. We consider a subsurface magma reservoir in an isotropic, viscoelastic space, see Figure 6. In general the system evolves in time in response to mass, momentum, and energy balance associated with magma transport in and out of the reservoir. We focus here on the host response to one particular state variable, a uniform but time-evolving pressure on the reservoir wall.

We employ a cylindrical coordinate system (r, z, θ) with the origin at the reservoir center. The assumption of axisymmetry means the problem shows no variation along the θ -coordinate enabling solutions in the one-sided (r, z) -plane. Figure 6 illustrates the geometry which defines the computational region surrounding a reservoir. The magma cavity has horizontal axis $a > 0$ and vertical axis $b > 0$, with center at the origin, and Earth’s free surface at $z = D + b$ (z positive upwards). Maximum depth of the computational domain is denoted by L_z and the maximum lateral distance from the center of radial symmetry is denoted by L_r .

We construct the region outside of the cavity by intersecting a closed, rectangular region $\mathcal{D} = \{(r, z) \in \mathbb{R}^2 \mid 0 < r < L_r, -L_z < z < D + b\}$ and a punctured domain $\mathcal{B} = \{(r, z) \in \mathbb{R}^2 \mid \frac{r^2}{a^2} + \frac{z^2}{b^2} > 1\}$. The region Ω outside of the cavity, defined by $\Omega = \mathcal{D} \cap \mathcal{B}$ forms our two-dimensional

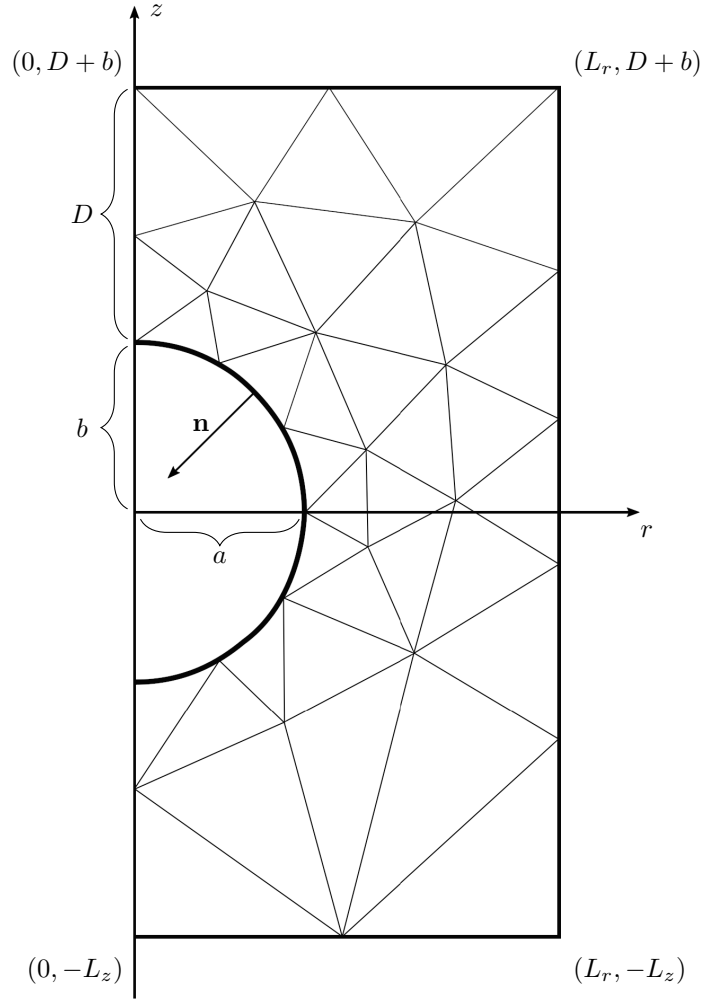


Fig. 6 The region Ω outside a subsurface, spheroidal magma reservoir centered at the origin is discretized with a high-order FEM. The reservoir has a horizontal axis $a > 0$ and vertical axis $b > 0$. The distance from the top of the reservoir to the surface is $D > b$. The region is bounded by a maximal depth L_z and maximal distance from the radial center L_r . Note that our simulations are performed on a finer triangulation of points than is shown here.

computational domain. The physical three-dimensional problem is posed on the revolution of Ω , the three-dimensional domain we denote by $\check{\Omega}$.

3.2.2 Governing Equations. We assume sufficiently slow deformation so that quasi-static viscoelasticity is a valid description of the momentum balance. We assume the medium deforms according to the Maxwell constitutive law [90]. This material model is chosen for its simplicity and flexibility. A variety of linear and nonlinear viscoelastic models have been proposed for crustal rocks at high temperature; the Maxwell model is a useful and easily generalizable reference case for understanding the phenomenology of viscoelastic deformation [81], [91], [92].

Let $\mathbf{u}, \underline{\varepsilon}, \underline{\gamma}, \underline{\sigma}$ be, respectively, the displacement vector, the total strain tensor, the viscous strain tensor, and the stress tensor. The time derivative of $\underline{\gamma}$ is denoted by $\dot{\underline{\gamma}}$. The relevant governing equations are:

$$\operatorname{div} \underline{\sigma} = \mathbf{f} \quad \text{in } \check{\Omega}, \quad (3.2.1a)$$

$$\dot{\underline{\gamma}} = \mathbf{A} \underline{\sigma} \quad \text{in } \check{\Omega}, \quad (3.2.1b)$$

$$\underline{\sigma} = \mathbf{E}(\underline{\varepsilon}(\mathbf{u}) - \underline{\gamma}) \quad \text{in } \check{\Omega}, \quad (3.2.1c)$$

where $\underline{\varepsilon}(\mathbf{u}) = (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/2$, $\mathbf{E}$ is the fourth-order, isotropic elastic stiffness tensor whose (i, j, k, l) -component in Cartesian coordinates is given by

$$E_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}). \quad (3.2.2)$$

Here, μ denotes the shear modulus, λ denotes Lamé's first parameter, and δ denotes the components of the identity tensor. The fourth-order tensor $\mathbf{A}$ relates viscous strain to stress, and is derived from the Maxwell constitutive law [90] to produce the form

$$\mathbf{A} \underline{\sigma} = \frac{1}{2\eta} \left(\sigma_{ij} - \frac{1}{3} \sigma_{kk} \delta_{ij} \right), \quad (3.2.3)$$

where η denotes the viscosity and repeated indices indicate summation over that index.

Equation (3.2.1a) is the static equilibrium equation where $\mathbf{f}$ represents body forces. Equation (3.2.1b) is the aging law for a Maxwell material and Equation (3.2.1c) is Hooke's Law. When supplemented by initial and boundary conditions, the system (3.2.1a) can be solved in any coordinate system.

We use the cylindrical coordinate system (r, z, θ) , writing the displacement vector field as $\mathbf{u} = u_r \mathbf{e}_r + u_z \mathbf{e}_z + u_\theta \mathbf{e}_\theta$ where $\mathbf{e}_r, \mathbf{e}_\theta$, and $\mathbf{e}_z$ denote the unit vectors of the cylindrical coordinate system. The source $\mathbf{f}$ can also be similarly expressed. We assume that u_θ and f_θ are zero. Furthermore, by the assumption of axial symmetry, u_r and u_z are independent of θ . Hence, employing the cylindrical components of the strain tensor, displacements in the Earth are related to strains by

$$\underline{\epsilon}(\mathbf{u}) = \frac{u_r}{r} \mathbf{e}_\theta \otimes \mathbf{e}_\theta + \sum_{i,j \in \{r,z\}} \frac{1}{2} (\partial_i u_j + \partial_j u_i) \mathbf{e}_i \otimes \mathbf{e}_j. \quad (3.2.4)$$

The stress tensor can be expressed, omitting its zero components, as

$$\underline{\sigma} = \sigma_{\theta\theta} \mathbf{e}_\theta \otimes \mathbf{e}_\theta + \sum_{i,j \in \{r,z\}} \sigma_{ij} \mathbf{e}_i \otimes \mathbf{e}_j. \quad (3.2.5)$$

The equilibrium equation (3.2.1a) then takes the form

$$\left(\partial_r \sigma_{rr} + \partial_z \sigma_{rz} + \frac{1}{r} (\sigma_{rr} - \sigma_{\theta\theta}) \right) \mathbf{e}_r + \left(\partial_r \sigma_{rz} + \partial_z \sigma_{zz} + \frac{1}{r} \sigma_{rz} \right) \mathbf{e}_z = \mathbf{f}. \quad (3.2.6)$$

Using (3.2.4) and (3.2.1c) to obtain expressions for the cylindrical components of the stress tensor, the equilibrium equation (3.2.6) can be solved for the components of the displacement in the two-dimensional meridian (rz) plane.

To reduce the problem to the meridian half-plane where $r > 0$, we need to impose the following boundary conditions on the axial boundary $\Gamma_0 = \{(r, z) \in \partial\Omega : r = 0\}$, namely

$$u_r = 0, \quad \text{on } \Gamma_0 \quad (3.2.7a)$$

$$\sigma_{rz} = 0, \quad \text{on } \Gamma_0. \quad (3.2.7b)$$

The first follows from a “no-opening” condition at $r = 0$. The second comes from requiring continuity of stresses in the $\mathbf{e}_z$ direction at $r = 0$. Other boundary conditions are imposed by partitioning the remaining boundary $\partial\Omega \setminus \Gamma_0$. We let $\Gamma_{\text{disp}} \subseteq \partial\Omega$ and $\Gamma_{\text{trac}} = \partial\Omega \setminus \Gamma_{\text{disp}}$ denote a general partitioning of $\partial\Omega$ into subdomains where either displacement or traction boundary conditions are imposed, respectively. Explicitly, these conditions are

$$\mathbf{u} = \mathbf{g}_{\text{disp}}(t) \quad \text{on } \Gamma_{\text{disp}}, \quad (3.2.7c)$$

$$\underline{\sigma} \cdot \mathbf{n} = \mathbf{g}_{\text{trac}}(t) \quad \text{on } \Gamma_{\text{trac}}, \quad (3.2.7d)$$

where $\mathbf{n}$ is the outward unit normal to the domain Ω , and $\mathbf{g}_{\text{disp}}$, $\mathbf{g}_{\text{trac}}(t)$ are given, time-varying boundary data. This general model enables the study of reservoir pressure, lateral loads and topography, among other studies in axisymmetric geometries.

In addition to boundary conditions, we must also supplement the aging law, Equation (3.2.1b), with an initial condition on viscous strain, namely

$$\underline{\gamma}(r, z, t = 0) = \underline{\gamma}_0(r, z), \quad (r, z) \in \Omega. \quad (3.2.8)$$

3.3 Computational Framework

We solve initial-boundary-value problem (Equations (3.2.1a),(3.2.4)-(3.2.8)) numerically by pairing a finite difference discretization in time with a high-order finite element method (FEM) in space. As described in this section, at each time step the spatial problem is governed by static equilibrium, with viscous effects manifested as a time-dependent source term. Simulations are done using Python code developed on top of the free and open source multi-physics library NGSolve [93] and the accompanying mesh generator [94]. The Python code is available in a public repository [95]. We use a two-dimensional mesh of triangles. To capture the magma chamber boundary accurately, we use nonlinear mappings for those elements with edges on the curved boundary to improve geometrical conformity [96]. The following subsections outline the static problem, the temporal discretization, and the details of the specific problem considered in this work.

3.3.1 Solving the Static Equilibrium Equation. We solve the equilibrium Equation (3.2.1a) subject to boundary conditions (3.2.7) using a FEM, which requires the weak form of the problem. To construct the weak form, we perform the following steps: (i) multiply equation (3.2.6) by r and take the dot product of both sides with a test function $\mathbf{v} = v_r \mathbf{e}_r + v_z \mathbf{e}_z$, (ii) integrate by parts on Ω , (iii) replace σ_{ij} by functions of u_i using (3.2.4) and (3.2.1c), and (iv) incorporate the boundary conditions of (3.2.7), letting $\mathbf{v}$ take on homogeneous displacement boundary conditions on Γ_{disp} . The result is the equation

$$\int_{\Omega} \mathbf{E}(\underline{\varepsilon}(\mathbf{u}) - \underline{\gamma}) : \underline{\varepsilon}(\mathbf{v}) r dr dz - \int_{\Gamma_{\text{trac}}} \mathbf{g}_{\text{trac}} \cdot \mathbf{v} r ds = - \int_{\Omega} \mathbf{f} \cdot \mathbf{v} r dr dz. \quad (3.3.1)$$

Here the colon denotes the Frobenius inner product. To simplify notation, we let $(\cdot, \cdot)_r$ and $\langle \cdot, \cdot \rangle_r$ respectively denote the integrals over Ω and Γ_{trac} of r multiplied by the appropriate (dot or Frobenius) inner product of the arguments. Then the above equation may be rewritten as

$$(\mathbf{E}\underline{\varepsilon}(\mathbf{u}), \underline{\varepsilon}(\mathbf{v}))_r = -(\mathbf{f}, \mathbf{v})_r + \langle \mathbf{g}_{\text{trac}}, \mathbf{v} \rangle_r + (\mathbf{E}\boldsymbol{\gamma}, \mathbf{v})_r. \quad (3.3.2)$$

The Lagrange FEM is derived by imposing the above equation on a space of piecewise polynomials. Given a triangulation of Ω , denoted by Ω_h , the Lagrange finite element space of order p , denoted by V_h consists of all functions which are continuous on Ω whose restriction to each element K of Ω_h is a polynomial of degree at most p in r and z . The method is high-order, meaning that polynomials of high degree can be used within each mesh element to approximate the solution. When degree p is used within an element of diameter h , a smooth solution on that element can be approximated at the rate $O(h^{p+1})$ (in the L^2 -norm). Thus, the difference between $p = 1$ and $p = 2$ case, for example, is that as $h \rightarrow 0$, under a uniform mesh refinement halving element diameters, while we expect the error to be halved in the first case per $O(h)$, in the second case, it reduces by a fourth due to $O(h^2)$ convergence. In the FEM, the data $\mathbf{f}$ and $\mathbf{g}_{\text{trac}}$ are integrated while the data $\mathbf{g}_{\text{disp}}$ is interpolated. Assuming the latter interpolation is done, let

$$\mathbf{V}_h^{\mathbf{g}_{\text{disp}}} = \{\mathbf{v} = v_r \mathbf{e}_r + v_z \mathbf{e}_z : v_r \in V_h, v_z \in V_h, \text{ and } \mathbf{v}|_{\Gamma_{\text{disp}}} = \mathbf{g}_{\text{disp}}\}.$$

Also let

$$\mathbf{V}_h^0 = \{\mathbf{v} = v_r \mathbf{e}_r + v_z \mathbf{e}_z : v_r \in V_h, v_z \in V_h, \text{ and } \mathbf{v}|_{\Gamma_{\text{disp}}} = \mathbf{0}\}.$$

Then, the FEM computes $\mathbf{u}_h \in \mathbf{V}_h^{\mathbf{g}_{\text{disp}}}$ satisfying

$$(\mathbf{E}\underline{\varepsilon}(\mathbf{u}_h), \underline{\varepsilon}(\mathbf{v}))_r = -(\mathbf{f}, \mathbf{v})_r + \langle \mathbf{g}_{\text{trac}}, \mathbf{v} \rangle_r + (\mathbf{E}\boldsymbol{\gamma}, \mathbf{v})_r, \quad \text{for all } \mathbf{v} \in \mathbf{V}_h^0, \quad (3.3.3)$$

provided $\mathbf{f}$, $\mathbf{g}_{\text{disp}}$, $\mathbf{g}_{\text{trac}}$, and $\boldsymbol{\gamma}$ are given. Equation (3.3.3) leads to a linear system of equations once a finite element basis of shape functions (which are basis functions determining one degree of freedom in the finite element system) is used.

3.3.2 Temporal Discretization. Our time-stepping method is inspired by that of [97] where viscous strains appear as a time-dependent source term on the equilibrium equation, namely, once $\boldsymbol{\gamma}$ is computed at a specific time, it appears as a known term in (3.3.3) which can then be solved for a displacement approximation. However, to compute $\boldsymbol{\gamma}$, we need to apply a time integrator to the aging law, Equation (3.2.1b). We now illustrate this time-stepping method in general terms so that it could be implemented in a variety of computational settings. We then specifically detail it for our FEM framework. First, assume all fields are known at time t^n . The procedure to integrate to t^{n+1} over step size $\Delta t = t^{n+1} - t^n$ is as follows:

1. Use the current solution at time t^n to integrate viscous strain one time step (e.g. with the aging law (3.2.1b)) and obtain $\boldsymbol{\gamma}^{n+1}$.
2. Compute all the known data (e.g. source terms, boundary data) at time t^{n+1} and use these (together with $\boldsymbol{\gamma}^{n+1}$) to solve the equilibrium equation to obtain the remaining fields at time t^{n+1} .

For our specific FEM framework the above time-stepping method is implemented as follows: For computational purposes only it is convenient to let $\underline{\mathbf{C}} = \mathbf{E}\boldsymbol{\gamma}$, since the use of $\underline{\mathbf{C}}$ allows us to skip the assembly and inversion of a mass matrix made of inhomogeneous material coefficients. Since $\mathbf{E}$ is time independent, simplifying $\mathbf{E}\mathbf{A}\boldsymbol{\sigma} = (\mu/\eta)\text{dev}(\boldsymbol{\sigma})$, Equation (3.2.1b) implies

$$\dot{\underline{\mathbf{C}}} = \frac{\mu}{\eta} \text{dev } \boldsymbol{\sigma}. \quad (3.3.4)$$

Here $\text{dev}(\boldsymbol{\sigma})$ denotes deviatoric tensor $\boldsymbol{\sigma} - \text{tr}(\boldsymbol{\sigma})$. Time integration of Equation (3.3.4) is carried out using the first-order accurate forward Euler method (chosen for its simplicity as we lay the computational groundwork; higher order methods will be incorporated in future developments). At each time step, we solve the weak form of equilibrium equation (Equation (3.3.3)) and use the computed displacement to obtain $\underline{\mathbf{C}}$ at the next time step. Again assuming all fields are known at time t^n , we specifically implement steps 1 and 2 above as:

1. Use $\mathbf{u}_h^n$ to update $\underline{\mathbf{C}}$ via forward Euler

$$\underline{\mathbf{C}}^{n+1} = \underline{\mathbf{C}}^n + \Delta t \frac{\mu}{\eta} \text{dev}(\mathbf{E}\boldsymbol{\varepsilon}(\mathbf{u}_h^n) - \underline{\mathbf{C}}^n). \quad (3.3.5)$$

2. Compute data $\mathbf{f}^{n+1}$, $\mathbf{g}_{\text{disp}}^{n+1}$, $\mathbf{g}_{\text{trac}}^{n+1}$ at time t^{n+1} and use them, together with the output of the previous step, to solve the static equation: compute $\mathbf{u}_h^{n+1} \in \mathbf{V}_h^{\mathbf{g}_{\text{disp}}^{n+1}}$ satisfying

$$(\mathbf{E}\underline{\boldsymbol{\varepsilon}}(\mathbf{u}_h^{n+1}), \underline{\boldsymbol{\varepsilon}}(\mathbf{v}))_r = -(\mathbf{f}^{n+1}, \mathbf{v})_r + \langle \mathbf{g}_{\text{trac}}^{n+1}, \mathbf{v} \rangle_r + (\underline{\mathbf{C}}^{n+1}, \mathbf{v})_r \quad (3.3.6)$$

for all $\mathbf{v} \in \mathbf{V}_h^0$.

Stable time steps for our problem are derived in B.1. B.1 also contains details of rigorous convergence tests in both space and time (to verify correctness) via the method of manufactured solutions (MMS) [89]. Code verification could also be done via benchmarking against simple analytic models [86] or community verification exercises e.g., [98].

3.3.3 Model Specifics and Non-Dimensionalization. The majority of analysis in this work will examine how a spatial distribution of viscoelastic properties impacts deformation around magma reservoirs subject to cyclic loading. We proceed by idealizing the boundary pressure as a sinusoid, which approximates a canonical problem in viscoelasticity [99], and provides a framework for studying arbitrary time dependent signals through superposition. For example, consider a forcing function $S(t)$ (which might be applied to any domain boundary) of duration ℓ , consisting of a linear-ramp until $t = \ell_c \leq \ell$, after which it remains constant at S_0 . $S(t)$ can be expressed as a superposition of sinusoids via its Fourier-sine series expansion

$$S(t) = \sum_{n=1}^{\infty} b_n \sin(\omega_n t), \quad (3.3.7)$$

where $b_n = 2S_0(-1)^{n+1}/(n\pi) + 2S_0\ell \sin(n\pi\ell_c/\ell)/(\ell_c n^2 \pi^2)$ and $\omega_n = n\pi/\ell$. We discuss the representation of such general time-varying functions further in B.2, but restrict our analysis to particular components of (3.3.7) in what follows.

We assume a specific boundary partition where Γ_{trac} encompasses the reservoir wall, Earth's free surface, and the computational boundary at depth ($z = -L_z$). Γ_{disp} is the lateral boundary $r = L_r$. We then set specific boundary data

$$\mathbf{g}_{\text{disp}}(t) = 0, \quad (3.3.8)$$

so that displacements vanish at $r = L_r$. At Earth's free surface and at depth we take

$$\mathbf{g}_{\text{trac}}(t) = 0. \quad (3.3.9)$$

At the reservoir wall we set

$$-\mathbf{n} \cdot \mathbf{g}_{\text{trac}}(t) = P(t), \quad (3.3.10a)$$

$$\mathbf{m} \cdot \mathbf{g}_{\text{trac}}(t) = 0, \quad (3.3.10b)$$

where

$$P(t) = P_0 \sin(\omega t). \quad (3.3.11)$$

Equation (3.3.10a) sets the normal component of the traction vector (the pressure) equal to a sinusoidal time-varying condition with amplitude P_0 and frequency ω . In what follows we will often refer to forcing period

$$\tau = 2\pi/\omega \quad (3.3.12)$$

rather than frequency. Equation (3.3.10b) imposes that the shear component of traction be equal to 0, where vector $\mathbf{m} = \mathbf{n} \times \mathbf{e}_z$ is tangent to the reservoir wall.

Non-dimensionalization of the governing equations reveals important physical parameters and re-scales the problem to help reduce round-off errors. We begin by handling the scaling of the spatial domain before addressing governing equations. Tildes in what follows indicate non-dimensional variables. Let $r = a\tilde{r}$, $z = a\tilde{z}$, $\tilde{\mathcal{D}} = \{(\tilde{r}, \tilde{z}) \in \mathbb{R}^2 \mid 0 \leq \tilde{r} \leq \frac{L_r}{a}, -\frac{L_z}{a} \leq \tilde{z} \leq \frac{D+b}{a}\}$ and $\tilde{\mathcal{B}} = \{(\tilde{r}, \tilde{z}) \in \mathbb{R}^2 \mid \tilde{r}^2 + \frac{a^2}{b^2}\tilde{z} \geq 1\}$. Then our resulting scaled domain is given by

$$\tilde{\Omega} = \tilde{\mathcal{D}} \cap \tilde{\mathcal{B}}, \quad (3.3.13)$$

with scaled boundaries $\tilde{\Gamma}_{\text{disp}}$ still representing the (scaled) lateral boundary and $\tilde{\Gamma}_{\text{trac}}$ the (scaled) reservoir wall, Earth's free surface, and computational boundary at depth. We also scale displacements by a , namely $a\tilde{\mathbf{u}} = \mathbf{u}$, which effectively means that total strain $\underline{\epsilon}$ is not scaled. We scale stress and time by the amplitude and frequency of the sinusoidal pressure, $\mathbf{E}$ by characteristic

shear modulus μ and body force by its magnitude F_0 (for example magnitude of gravitational force), giving

$$\underline{\sigma} = P_0 \tilde{\sigma}, \quad (3.3.14)$$

$$\underline{E} = \mu \tilde{E}, \quad (3.3.15)$$

$$\underline{\mathbf{f}} = F_0 \tilde{\mathbf{f}}, \quad (3.3.16)$$

$$t\omega = \tilde{t}, \quad (3.3.17)$$

which implies a scaling of $\underline{C} = P_0 \tilde{C}$. The scaled form of the equilibrium Equation (3.2.1a) is thus

$$\text{div } \underline{\sigma} = \frac{aF_0}{P_0} \tilde{\mathbf{f}}, \quad (3.3.18)$$

and Hooke's law Equation (3.2.1c) becomes

$$\tilde{\sigma} = \frac{\mu}{P_0} \tilde{E}(\underline{\varepsilon} - \underline{\gamma}). \quad (3.3.19)$$

The two dimensionless parameters in Equations 3.3.18-3.3.19 physically represent the ratio of body force to reservoir boundary tractions, and a scaled reservoir pressure, respectively.

The modified aging law (Equation (3.3.4)) becomes

$$\partial_{\tilde{t}} \tilde{C} = \frac{1}{De} \text{dev } \tilde{\sigma}, \quad (3.3.20)$$

where

$$De = \frac{\eta\omega}{\mu} = \frac{2\pi\eta}{\tau\mu} \quad (3.3.21)$$

is the non-dimensional Deborah number, a ratio of elastic pressurization timescale $\tau/2\pi$ to Maxwell viscous relaxation timescale η/μ . Viscosity η , shear modulus μ and pressurization time τ are understood to be characteristic scales if spatially or time variable. De commonly appears as a control parameter in models for magma chamber mechanics [71], [100], cycles of eruptions [72], [101], and the spatial structure of transcrustal magma systems [78], [102]. It will play an important

role in our results.

Computationally, all problems considered in this work are solved in this non-dimensional form. The specific non-dimensional boundary conditions we thus take are

$$\underline{\tilde{\mathbf{u}}} = 0 \quad \text{on } \tilde{\Gamma}_{\text{disp}}, \quad (3.3.22a)$$

$$\underline{\tilde{\boldsymbol{\sigma}}}\mathbf{n} = \tilde{\mathbf{g}}_{\text{trac}}(\tilde{t}) \quad \text{on } \tilde{\Gamma}_{\text{trac}}, \quad (3.3.22b)$$

and at the reservoir wall,

$$-\mathbf{n} \cdot \tilde{\mathbf{g}}_{\text{disp}}(\tilde{t}) = \tilde{P}(\tilde{t}) \quad (3.3.23)$$

$$\mathbf{m} \cdot \tilde{\mathbf{g}}_{\text{trac}}(\tilde{t}) = 0. \quad (3.3.24)$$

where $\tilde{P}(\tilde{t}) = \sin(\tilde{t})$. For all our applications we assume negligible body forces, so $aF_0/P_0 \ll 1$.

3.3.4 Temperature-Dependent Material Parameters. We assume that viscosity of crustal rocks is described by a temperature-dependent Arrhenius relation, an assumption common to many thermomechanical models of magmatic systems e.g., [79]. This neglects grain-size and stress-dependent effects [103], but parameterizes our assumption that temperature is the dominant factor controlling crustal rheology during crustal magma transport. In general, temperature evolves in time in response to magmatism e.g., [104], but we assume a steady state geotherm here as our goal is simply to explore the role of realistic spatial structure of material parameters.

Accordingly, we solve the stationary heat equation

$$\nabla^2 T = 0 \quad \text{in } \check{\Omega}, \quad (3.3.25)$$

where $T(r, z)$ is the temperature field, which we assume to be axisymmetric. At the top, bottom and lateral parts of the boundary, we enforce a steady-state geothermal profile given by

$$T(z) = T_s - \alpha(z - (D + b)), \quad (3.3.26)$$

where T_s is the surface temperature constant and α is a parameter specifying the temperature gradient. At the chamber wall we set $T = T_c$, a constant temperature. We use a finite element space of order p to solve the heat equation. Here, p is the same order as is used in the finite element

solution of the equilibrium equation. The formulation uses radial weighting to reduce the problem to the two-dimensional domain Ω and as usual—see e.g., [105]—set zero temperature flux $\nabla T = 0$ at Γ_0 , the $r = 0$ boundary, to maintain our consideration of a one-sided problem. The solution of this BVP for the heat equation informs the temperature field throughout the domain, from which the viscosity is deduced according to the Arrhenius formula

$$\eta = A_D \exp\left(\frac{E_a}{RT}\right) \quad (3.3.27)$$

where A_D is the Dorn parameter, E_a is the activation energy, and R is the Boltzmann constant. For numerical computation, we prefer to use the equivalent formula

$$\eta = \eta_0 \exp\left(\frac{E_a}{R} \left[\frac{1}{T} - \frac{1}{T_s}\right]\right), \quad (3.3.28)$$

where $\eta_0 = A_D \exp\left(\frac{E_a}{RT_s}\right)$, to avoid numerical issues associated with very large viscosities predicted by low temperatures in the near surface. In Equation (3.3.28) we use absolute temperature, so both T and T_s should be converted from degrees Celsius to Kelvin.

As shown in B.1, numerically stable time steps depend on Deborah number, thus the exponential dependence of viscosity leads to prohibitively small time steps at high temperatures. This limits the degree to which we can exactly explore high magma temperatures without artificially thresholding model temperature.

Elastic parameters are also considered to be temperature dependent. [106] provide smooth and continuous forms for temperature-dependent Young's modulus $E(T)$ and Poisson's ratio $\nu(T)$ as

$$E(T) = c_1 \left[1 - \operatorname{erf}\left(\frac{T - \bar{T}}{s}\right)\right] + c_2 T + c_3, \quad (3.3.29)$$

$$\nu(T) = \left[1 - \frac{E}{E_{\max}}\right] \cdot [\nu_{\max} - \nu_{\min}] + \nu_{\min} \quad (3.3.30)$$

where $\nu_{\min} = 0.25$, $\nu_{\max} = 0.49$ define the range of possible Poisson's ratios and $E_{\max}$ is the max value Young's modulus achieves for a given temperature profile. $\bar{T}$ is a temperature threshold for which Young's modulus decreases by an order of magnitude and c_1, c_2, c_3, s are empirical parameters. To convert E and ν to λ, μ (elastic moduli used in our framework), we use $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$, $\mu = \frac{E}{2(1+\nu)}$. Figure 7 demonstrates the spatial pattern exhibited by the material parameters for

a temperature profile characterized by 800°C reservoir temperature, 0°C surface temperature and a geothermal gradient of 20°C/km.

3.4 Analysis of time dependent viscoelastic deformation

We now develop tools to analyze the time evolution of viscoelastic deformation predicted from our numerical calculations. Towards our goal of examining how a realistic distribution of viscoelastic properties impacts deformation around magma reservoirs subject to cyclic loading, we begin with a 1D analysis of the Maxwell model to illustrate inherent properties of the system which may be generalized in the 2D problem. This analysis is generalizable to other viscoelastic models, and leads to concrete implications for inferring viscoelastic behavior in magmatic systems from ground deformation.

3.4.1 Insights from the 1D Maxwell Model. Given the spatial domain $x \in [0, L]$, the 1D strain-displacement relation is given by

$$\varepsilon = u_x \tag{3.4.1}$$

and the 1D governing equations (equilibrium, viscous strain evolution and Hooke's law, respectively) are

$$\frac{\partial \sigma}{\partial x} = 0, \tag{3.4.2a}$$

$$\dot{\gamma} = \frac{1}{\eta} \sigma, \tag{3.4.2b}$$

$$\sigma = \mu(\varepsilon - \gamma), \tag{3.4.2c}$$

where σ , ε , γ , and u are, respectively, the 1D stress, total strain, viscous strain, and displacement. Boundary conditions are chosen to reflect the conditions for the 2D problem. The origin experiences the sinusoidal pressure condition (representing the reservoir) and displacements vanish at the far boundary, namely

$$\sigma(x = 0, t) = \sin(\omega t), \tag{3.4.3a}$$

$$u(x = L, t) = 0. \tag{3.4.3b}$$

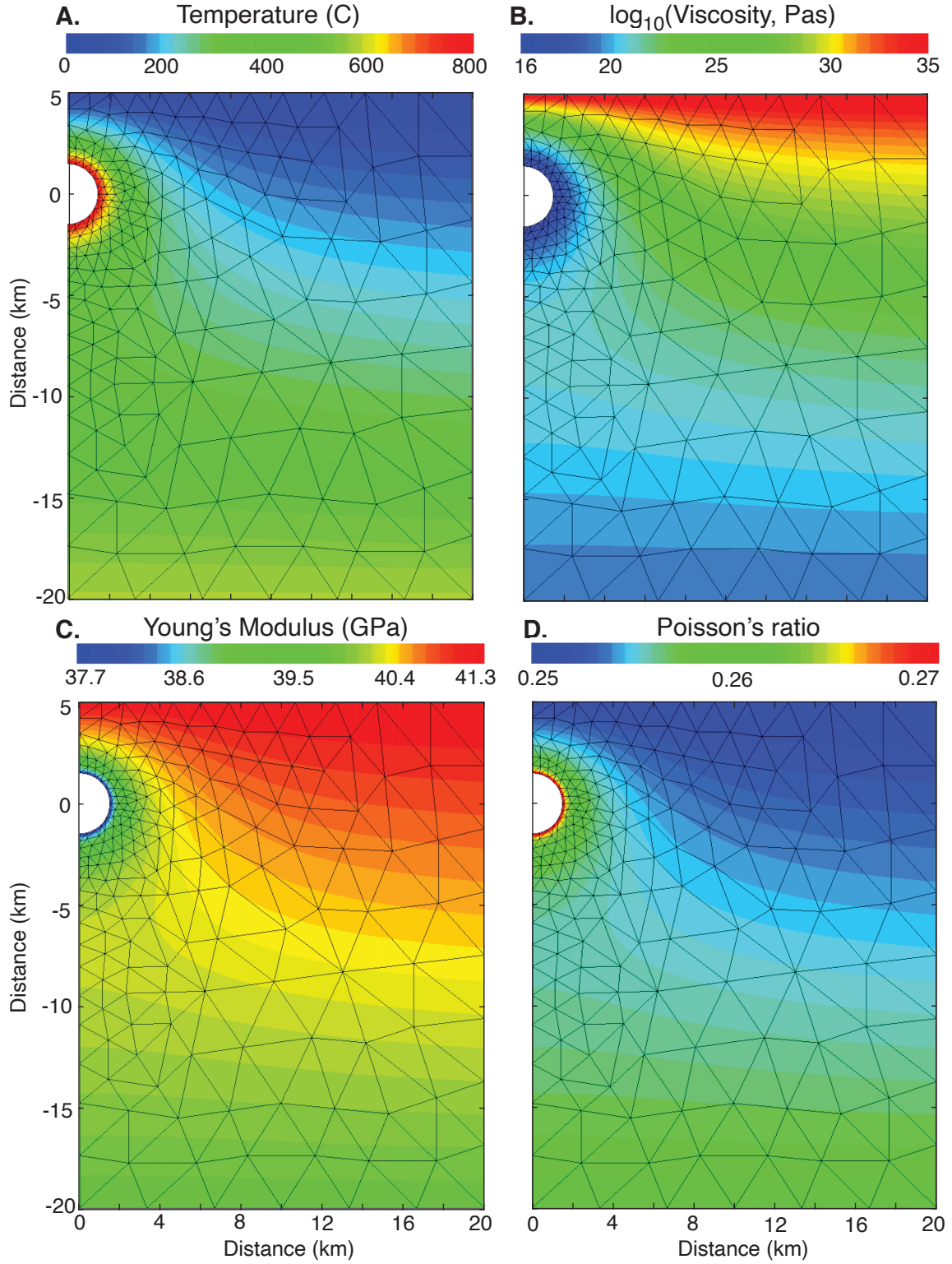


Fig. 7 Material parameters used in our reference variable coefficients parameter study, with finite element mesh overlaid. **A.** Temperature, obtained by solving Equation (3.3.25) with $T_c = 800^\circ\text{C}$, surface temperature $T_s = 0^\circ\text{C}$, and geothermal gradient $\alpha = 20^\circ\text{C/km}$. **B.** Viscosity from Equation (3.3.28). **C.** Young's Modulus from Equation (3.3.29). **D.** Poisson's ratio from Equation (3.3.30).

We consider $t > 0$; the aging law Equation (3.4.2b) thus requires an initial viscous strain to be specified, which we express in general terms

$$\gamma(x, t = 0) = \gamma_0(x), \quad (3.4.4)$$

where γ_0 as a given function. The Maxwell model thus gives rise to an initial-boundary value problem defined by Equations 3.4.1-3.4.4.

We are interested in the response between stress and strain at the reservoir boundary, with the expectation that viscous relaxation will lead to a phase difference. To do this analysis it is useful to work with Hooke's law in rate form, namely,

$$\dot{\epsilon} = \frac{1}{\mu} \dot{\sigma} + \frac{1}{\eta} \sigma. \quad (3.4.5)$$

Following [99], application of the Fourier transform to Equation (3.4.5) yields the constitutive law in frequency space

$$\hat{\sigma}(\omega) = \hat{\mu}(\omega) \hat{\epsilon}(\omega), \quad (3.4.6)$$

which gives the usual relationship where stress is expressed as a function of strain through a complex shear modulus $\hat{\mu}$ defined by

$$\hat{\mu}(\omega) = \left(\frac{1}{\mu} - i \frac{1}{\eta\omega} \right)^{-1}. \quad (3.4.7)$$

The decomposition $\hat{\mu}(\omega) = \hat{\mu}_1(\omega) + i\hat{\mu}_2(\omega)$ into storage and loss moduli allows us to express $\hat{\mu}$ as

$$\hat{\mu}(\omega) = |\hat{\mu}(\omega)| e^{-i\delta} \quad (3.4.8)$$

where $\delta = -\tan^{-1}(\frac{\hat{\mu}_2}{\hat{\mu}_1})$.

In our applications, however, we are interested in the strain response to an applied (sinusoidal) stress, thus we must consider the constitutive relation Equation (3.4.6) in the form

$$\hat{\varepsilon}(\omega) = \hat{d}(\omega)\hat{\sigma}(\omega), \quad (3.4.9)$$

where $\hat{d}(\omega) = 1/\hat{\mu}(\omega)$ is the complex creep modulus given by

$$\hat{d}(\omega) = \frac{1}{\mu} - i\frac{1}{\eta\omega}, \quad (3.4.10)$$

which can be decomposed into $\hat{d}(\omega) = \hat{d}_1(\omega) + i\hat{d}_2(\omega)$ as before, and gives rise to the similar form

$$\hat{d}(\omega) = |\hat{d}(\omega)|e^{-i\beta}, \quad (3.4.11)$$

for $\beta = -\tan^{-1}\left(\frac{\hat{d}_2(\omega)}{\hat{d}_1(\omega)}\right)$. Applying the inverse Fourier transform to Equation (3.4.9) and using 3.4.3a yields

$$\begin{aligned} \varepsilon(t) &= [d * \sigma](t), \\ &= \hat{d}_1(\omega) \sin \omega t + \hat{d}_2(\omega) \cos \omega t, \\ &= \sin(\omega t - \beta), \end{aligned} \quad (3.4.12)$$

which gives strain as an explicit function of stress, delayed by phase lag β . Since $\hat{d}$ is chosen as the multiplicative inverse of $\hat{\mu}$ note that

$$|\hat{d}(\omega)| = \frac{1}{|\hat{\mu}(\omega)|}, \quad (3.4.13a)$$

$$\beta = -\delta, \quad (3.4.13b)$$

therefore the phase lag that strain experiences in response to an applied stress will be equal and opposite when reversing roles and considering stress in response to an applied strain. Note that we have used the sign convention for the phase lag such that positive values of β correspond to strain lagging behind stress.

To summarize, the strain response to a sinusoidal stress is also sinusoidal with a phase lag β , which can be simplified in terms of the Deborah number De by substituting in the real and imaginary parts of $\hat{d}(\omega)$, resulting in

$$\beta = \tan^{-1} \left(\frac{1}{De} \right). \quad (3.4.14)$$

This analytic result provides insight into the physics of the viscoelastic model, as two limiting cases of the Deborah number (namely $De \rightarrow \infty$ and $De \rightarrow 0$) yield phase lags of 0 and $\pi/2$ (respectively) corresponding to the elastic and viscous limits (respectively). In addition, these analytic results can be generalized to higher dimensions which we do in the next section, providing useful code verification metrics as well as providing insight into the frequency response of more physically realistic scenarios.

3.4.2 Transfer Function and Analytic Signals. The phase lag analysis for the 1D problem of the previous section can be generalized using the theory of Linear Time-Invariant (LTI) systems such as the viscoelastic problem we consider here. For general LTI systems, one can characterize some output signal $y(t)$ as the linear transformation of a system input $x(t)$, where we consider one-sided signals (i.e. they are 0 for $t < 0$) [107]. The response y can be determined as a convolution of the input x with the system impulse response h , namely

$$\begin{aligned} y(t) &= (x * h)(t) \\ &= \int_0^t x(t')h(t - t') dt'. \end{aligned} \quad (3.4.15)$$

The transfer function connecting the output signal $y(t)$ given the input signal $x(t)$ we denote $H\{y(t) | x(t)\}(i\omega)$, however we drop the argument within curly braces or functional dependence within parenthesis when these are implied via context. The transfer function is defined as

$$\begin{aligned} H(i\omega) &= \mathcal{L}\{h\}(i\omega) \\ &= \frac{\mathcal{L}\{y\}}{\mathcal{L}\{x\}}(i\omega), \end{aligned} \quad (3.4.16)$$

where $\mathcal{L}$ denotes the Laplace transform (a function of the complex variable s) and we have evaluated at $s = i\omega$. The transfer function thus provides the amplitude of the system output as a function of frequency of the input signal. As an example, Equation (3.4.9) illustrates how $\hat{d} = H\{\varepsilon(t) | \sigma(t)\}$, i.e the transfer function when stress is the input signal and strain is the output.

If we consider specific input and output signals $x(t) = A_{in} \sin(\omega t)$ and $y(t) = A_{out} \sin(\omega t - \phi)$, then we can use the Laplace transform to calculate the transfer function, namely,

$$\begin{aligned}
H(i\omega) &= \left. \frac{A_{\text{out}}}{A_{\text{in}}} \frac{(-s \sin(\phi) + \omega \cos(\phi))/(s^2 + \omega^2)}{\omega/(s^2 + \omega^2)} \right|_{s=i\omega} \\
&= \frac{A_{\text{out}}}{A_{\text{in}}} e^{-i\phi},
\end{aligned} \tag{3.4.17}$$

i.e. a constant, independent of ω . Performing an inverse Laplace transform indicates that the corresponding system impulse response is a delta function, namely, $h(t) = (A_{\text{out}}/A_{\text{in}})\delta(t - \phi/\omega)$.

Equation (3.4.17) illustrates the important point that evaluation at $s = i\omega$ must take place after the ratio is computed, so that the poles in the Laplace transforms of the sinusoids x and y are removed. In numerical studies making use of the discrete Fourier transform, this evaluation cannot be done after the ratio is computed, which can lead to division by zero. An alternative means for defining the transfer function therefore is via the concept of analytic signals, which have straight-forward numerical approximations and avoid potential division by zero.

Analytic signals are defined in the following manner. Consider the real valued signal $z(t)$ and denote its Fourier transform by $\hat{z}(\xi)$. Define the function

$$\hat{z}_a(\xi) = 2\mathcal{H}(\xi) \hat{z}(\xi) \tag{3.4.18}$$

(where $\mathcal{H}$ is the Heaviside step function), which contains only the non-negative frequency components of $\hat{z}(\xi)$. The analytic signal corresponding to z , denoted $z_a(t)$, is a complex-valued function obtained by transforming $\hat{z}_a$ back to the time domain using the inverse Fourier transform, yielding

$$z_a(t) = z(t) + i\mathbb{H}\{z\}(t), \tag{3.4.19}$$

where $\mathbb{H}$ is the Hilbert transform. Properties of Hilbert transforms mean that for input signal $x(t)$ and response signal $y(t)$ of an LTI system, we have that

$$y_a(t) = (h * x_a)(t). \tag{3.4.20}$$

Considering the analytic signals $x_a(t) = -iA_{\text{in}}e^{i\omega t}$ and $y_a(t) = -iA_{\text{out}}e^{i(\omega t - \phi)}$ associated with the input and output signals under consideration, plugging these into (3.4.20) yields

$$A_{\text{out}}e^{i(\omega t - \phi)} = A_{\text{in}}e^{i\omega t}H(i\omega). \quad (3.4.21)$$

Equation (3.4.21) illustrates the fact that for an input signals of form $e^{i\omega t}$ (called a characteristic function), the response signal is given by $e^{i\omega t}H(i\omega)$, indicating that the output signal is simply a scaling of the input by $H(i\omega)$.

We can solve (3.4.21) for the transfer function, namely,

$$H(i\omega) = \frac{A_{\text{out}}}{A_{\text{in}}}e^{-i\phi}, \quad (3.4.22)$$

previously obtained using Laplace transforms. The amplitude $|H| = \left| \frac{A_{\text{out}}}{A_{\text{in}}} \right|$ is often referred to as the gain because it describes how the frequency content in the output signal is amplified in response to the input. Finally $\phi = -\arg(H)$ is the phase lag, which agrees with that of the 1D Maxwell model considered in the previous section.

As a corollary, if the transfer function is known, we may directly relate the input and output signals. For example, let $x(t) = A \sin(\omega t - \psi)$, with phase ψ , be an input signal and let $H(i\omega) = |H(i\omega)|e^{-i\phi}$ be the transfer function. The analytic input signal is then $x_a(t) = -iAe^{i(\omega t - \psi)}$ and (3.4.20) implies that the analytic output signal is $y_a(t) = H(i\omega)x_a(t)$. The desired output signal $y(t)$ can be recovered by taking the real part of its analytic signal, namely

$$y(t) = |H(i\omega)|A \sin(\omega t - \psi - \phi). \quad (3.4.23)$$

In other words, a sinusoidal input function implies a sinusoidal output function, modulated by a phase lag ϕ and amplitude gain $|H|$.

If $\{A_k\}_{k=1}^n, \{\omega_k\}_{k=1}^n, \{\psi_k\}_{k=1}^n$ are sequences of amplitudes, frequencies, and phases, respectively, then a composite input signal can be expressed

$$x(t) = \sum_{k=1}^n A_k \sin(\omega_k t - \psi_k). \quad (3.4.24)$$

Note that each component is associated with a period $\tau_k = 2\pi/\omega_k$. By superposition, if $\{H(i\omega_k)\}_{k=1}^n$ are (known) associated transfer functions with phase lags $\{\phi_k\}_{k=1}^n$, then the corresponding output signal is given by

$$y(t) = \sum_{k=1}^n |H(i\omega_k)| A_k \sin(\omega_k t - \psi_k - \phi_k). \quad (3.4.25)$$

In discussion section 3.6, we illustrate this result for a specific composite input function defining magma reservoir pressure through time and numerically calculated transfer function for resulting surface displacements.

In the sections that follow, we explore numerically how the transfer function links reservoir pressure to surface displacements and strains. Following the notation for the transfer function, we let $\phi\{y(t) | x(t)\}$ denote the phase lag between the output signal $y(t)$ given the input signal $x(t)$, but drop the argument in curly braces when it is implied via context.

3.4.3 Numerical Calculations of the Transfer Function. The analytic signal corresponding to a real, discrete time-series is implemented in the Python SciPy library via the `scipy.signal.hilbert()` function. The transfer function connecting an input signal $x(t)$ to output signal $y(t)$ is computed via the ratio of corresponding analytic signals, from which we can compute phase and amplitude. All scripts are available in the code repository. In practice, there exists an initial spin-up period (~ 4 cycles) before solutions settle into a sinusoidal response and it is necessary to compute the transfer function once out of this phase.

In addition to the spin-up phase, the output signal can be shifted to oscillate around a non-zero value, which can complicate the calculation of the phase lag using our numerical techniques. The 1D analysis of the previous section illustrates why this occurs. Specifying the initial condition Equation (3.4.4) impacts the evolution of the displacement and stress fields in the following way: suppose $\gamma_0(x) = 0$ for each $x \in [0, L]$. We can simplify the boundary condition Equation (3.4.3) by taking $P_0 = \omega = 1$. The sinusoidal pressure imposed at the left boundary along with Equation (3.4.2a) imply a uniform stress field

$$\sigma(t, x) = \sin t. \quad (3.4.26)$$

Integrating Equation (3.4.2b) yields the viscous strain

$$\gamma(t) = -\frac{1}{\eta} \cos t + \frac{1}{\eta}, \quad (3.4.27)$$

and solving Equation (3.4.2c) for total strain gives the solution

$$\varepsilon(t) = \frac{1}{\mu} \sin t - \frac{1}{\eta} \cos t + \frac{1}{\eta}, \quad (3.4.28)$$

which illustrates how the strain response is sinusoidal with a shift of $1/\eta$. Although strain starts initially at 0, it fluctuates around the non-zero value $1/\eta$, corresponding to a volume change (length change in 1D). To avoid this situation, one could specify a different initial viscous strain, i.e. $\gamma_0(x) = -1/\eta$ which would yield a strain response fluctuating around zero. In the 2D problems considered in this work, it is difficult to know a priori the initial viscous strain that would preclude a volume change. Thus to compare the phase-lag response, fields that do not fluctuate around zero must first be shifted to do so. The spin-up phase contributes an exponentially decaying component in the output signal, therefore we calculate approximate phase and amplitude after 4 pressurization cycles.

The sinusoidal pressure forcing we impose at the reservoir wall given by Equation (3.3.10a) is considered the input signal $P(t)$ for all of our studies. To verify correctness of our numerical methods, we first consider as the output signal the normal component of strain at a single spatial point on the wall, namely $\varepsilon_{rr}(r = a, z = 0, t)$. Because at the reservoir wall the stress-strain relation effectively reduces to a 1D problem at a point, our numeric calculations are verified by comparing our numerical calculations of transfer function amplitude and phase lag against the theoretical stress-strain relationship for a Maxwell material for different forcing periods τ (see Equation (3.3.12)), as evidenced in Figure 8. In addition we compute the phase lag observed in the vertical component of displacement at Earth's surface $u_z(r = 0, z = D + b, t)$ as well as the transfer function amplitude (gain).

3.5 Computational Results

Viscoelastic behavior of magma reservoirs is often characterized in terms of deformation of a flat free surface induced by pressurization of a spheroidal reservoir e.g., [83], [85], [108]. Even in this relatively simple case, the problem is complex because a large number of control parameters matter and trade off in non-unique ways to generate surface deformation patterns. An additional challenge is that the problem is generally not amenable to analytic analysis such as has been conducted in simplified limits [67], [68], [82].

Having established our computational framework, we will now focus on a specific and relatively unexplored part of this problem here, the frequency dependence of surface deformation. All fixed

Table 1. Parameters used in Applications (unless otherwise noted).

Symbol	Explanation	Value
a	Ellipse semi-major axis	1500 m
b	Ellipse semi-minor axis	1500 m
D	Reservoir depth beneath Earth's surface	3500 m
L_r	Domain length in radial direction	20000 m
L_z	Domain length in vertical direction	20000 m
p	Degree of FE basis polynomials	4
P_0	Reservoir pressure amplitude	10 MPa
A_D	Dorn parameter	10^9 Pa s
A	Material-dependent constant for viscosity	4.25×10^7 Pa s
E_a	Activation energy	141 kJ/(mol)
R	Boltzmann's molar gas constant	8.314 J/(mol K)
T_c	Reservoir temperature	800°C
T_s	Surface temperature	0°C
α	Geothermal gradient	20°C/km
$\nu_{\min}$	Min Poisson's ratio	0.25
$\nu_{\max}$	Max Poisson's ratio	0.49
$E_{\max}$	Max Young's modulus	4.0×10^{10} Pa
c_1	Parameter in model for E	1.8×10^{10} Pa
c_2	Parameter in model for E	-3.5×10^6 Pa/°C
c_3	Parameter in model for E	4.3×10^9 Pa
s	Parameter in model for E	120 °C
$\bar{T}$	Temperature threshold	924°C

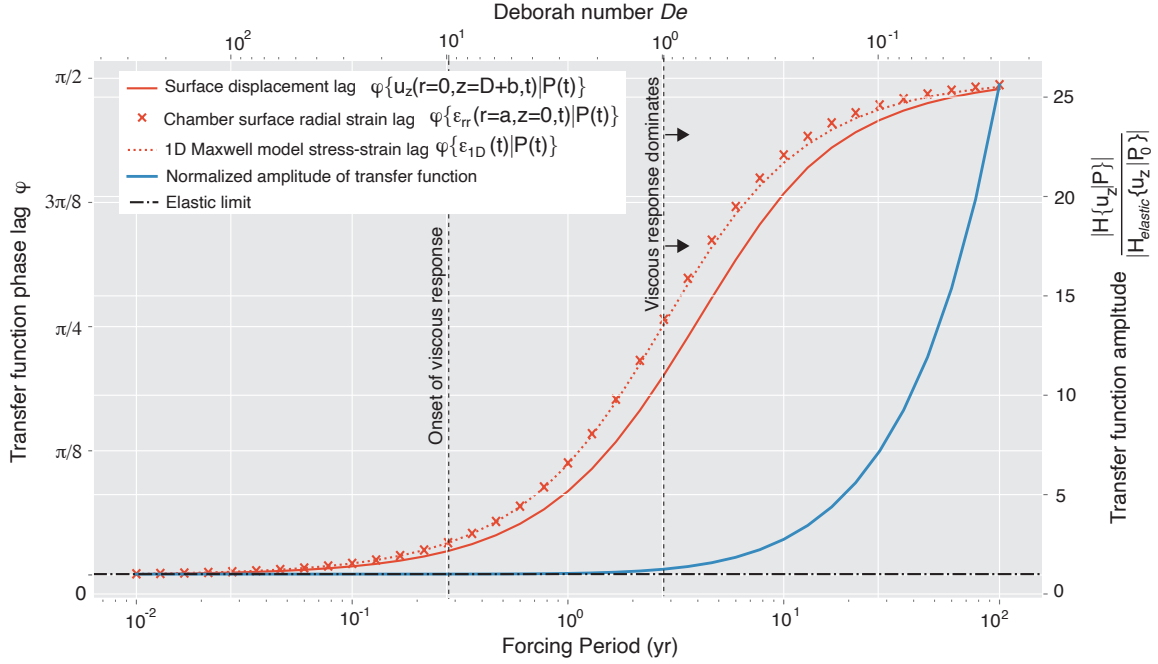


Fig. 8 Phase lag ϕ of the transfer function between reservoir pressure and radial strain at the reservoir wall ($\phi\{\epsilon_{rr}(r = a, z = 0, t|P(t))\}$, red dashed curve) and vertical displacement at the surface overlying the reservoir ($\phi\{u_z(r = 0, z = D + b, t|P(t))\}$, solid red curve). Crosses come from the 1D analytic prediction (Equation (3.4.14)). Right axis and blue curve plot the amplitude of the transfer function $|H\{u_z(r = 0, z = D + b, t|P(t))\}|$ normalized by the transfer function amplitude in a purely elastic limit (which uses the same averaged elastic coefficients but with $\eta = 1 \times 10^{34}$ making viscous effects negligible). Upper x axis is the Deborah number, lower x-axis dimensionalizes into period of sinusoidal pressure forcing using $\eta = 2.20 \times 10^{17}$ Pas, $\lambda = 16.7$ GPa and $\mu = 16.0$ GPa. Vertical dashed lines correspond to threshold Deborah numbers associated with onset of viscous response in host rocks.

parameters used in this study are listed in Table 1, unless otherwise noted. In the constant coefficient case studied in Figure 8 (a spherical reservoir in a uniform viscoelastic halfspace), sinusoidal forcing at the reservoir wall results in surface deformation patterns that are simply parameterized in terms of the Deborah number (Equation (3.4.14)). $De \approx 10$ signifies the onset of viscous response in host rocks, while for $De < 1$ the host rock response is dominantly viscous in the sense that phase lag ϕ between surface deformation is more than halfway to the viscous limit.

We construct constant coefficient models by choosing constant values of elastic parameters μ and λ through spatially averaging the non-constant coefficient calculations (Figure 8, bottom axis). For viscosity we suppose that a forcing period of 1 year yields a surface phase lag of 0.3 rad. From this phase lag we compute the associated Deborah number and solve Equation (3.3.21) for viscosity. The resulting constant material parameters are: $\mu = 16.0$ GPa, $\lambda = 16.7$ GPa, $\eta = 2.20 \times 10^{17}$ Pa.s. We can then associate a Deborah number De with a forcing period τ via Equation (3.3.21) and examine the transition to a viscous response as a function of forcing period. In this example $\tau = 1$ yr corresponds to maximum surface displacement that lags behind maximum chamber pressure by ~ 16 days at similar amplitude to the elastic limit, while $\tau = 10$ yr corresponds to a phase lag of ~ 1.9 years with $\sim 3\times$ amplitude to the elastic limit.

However, uniform viscosity is a poor approximation to crustal rheology in magmatic regions. To understand what changes with more realistic temperature-dependent viscosity and elastic constants, we also study how pressure forcing period affects ground deformation in the variable coefficient problem outlined in Section 3.3.3.

Figure 9 left axes show time series of maximum vertical surface displacement and radial strain at the reservoir wall (plotted versus dimensionless time) for several representative forcing periods τ associated with forcing by cyclic pressurization of the chamber (right axes). All quantities are normalized to facilitate comparison of phase lag as a function of forcing period, with amplitudes given in the legend. We see that phase lag differs in magnitude between surface and chamber wall.

Figure 10 plots the spatial variation in vertical and horizontal components of surface displacements u_z, u_r as well as the scalar von Mises stress $\sigma_v = \sqrt{3J_2}$ with J_2 the second deviatoric stress invariant for four positions in the pressure cycle ($\omega = 0, \pi/2, \pi, 3\pi/2$ radians) and three forcing periods. Black and white contours represent level curves of the spatially dependent Deborah number.

Finally, Figure 11 shows the transfer function phase $\phi\{u_z(r = 0, z = D + b, t) | P(t)\}$ and normalized amplitude $|H\{u_z(r = 0, z = D + b, t) | P(t)\}|/|H_{elastic}\{u_z(r = 0, z = D + b) | P_0\}|$ for a sweep through pressure forcing period τ . The elastic normalization $H_{elastic}$ is computed for each temperature separately, due to temperature dependence of elastic parameters E and ν (non-constant coefficient corrections to the known spherical cavity in half space elastic solution [109]). Transfer function results are computed for three choices of reservoir temperature $T_c = 800, 900, 1000^\circ\text{C}$ in Figure 11. The simulations are carried out at 37 logarithmically-spaced forcing periods between 0.01yr and 100yr. For each forcing period and reservoir temperature, we compute the transfer function phase and amplitude over 10 complete pressurization cycles. Because of computational burden associated with the highest reservoir temperature of 1000°C (see B.1) that lead to very small Deborah numbers, we set a maximal effective temperature of 900°C for computing material parameters in this case. We also perform an additional mesh refinement in space to mitigate poor resolution at longer forcing periods for the 1000°C reservoir.

In contrast to the constant coefficient case, Figures 9-11 demonstrate that temperature dependent material parameters strongly impact the frequency dependence of system viscoelastic response. Most pronounced is a saturation of phase lag at ~ 0.3 radians and muted amplification of displacements relative to the constant coefficient case. As evidenced by the large σ_v (which measures deviatoric shear stress magnitude), viscous effects are confined near the reservoir wall. This results in more pronounced mechanical lag at the reservoir wall than at the surface (Figure 9) and concentration of shear stress σ_v through the cycle in a narrow aureole around the chamber (Figure 10).

The strong spatial variability in material parameters now implies a spectrum of Maxwell relaxation times as has been noted in other studies, e.g., [81], and hence spatially variable Deborah number. Nonetheless, we see that a local value of De still characterizes the region experiencing significant viscous strain for each forcing period. Figure 10 shows that $De \approx 10$ effectively bounds the region experiencing significant von Mises stress, and hence viscous strain, in excess of chamber overpressure P_0 , with $De = 1$ once again a measure of the viscous region centroid. For small forcing periods the viscous region is significantly reduced ($De = 1$ does not appear for $\tau = 0.1$ year forcing period). Both contours are asymmetric with depth due to the geothermal gradient. To isolate viscous effects, the transfer amplitudes for Figure 11 are normalized using the variable

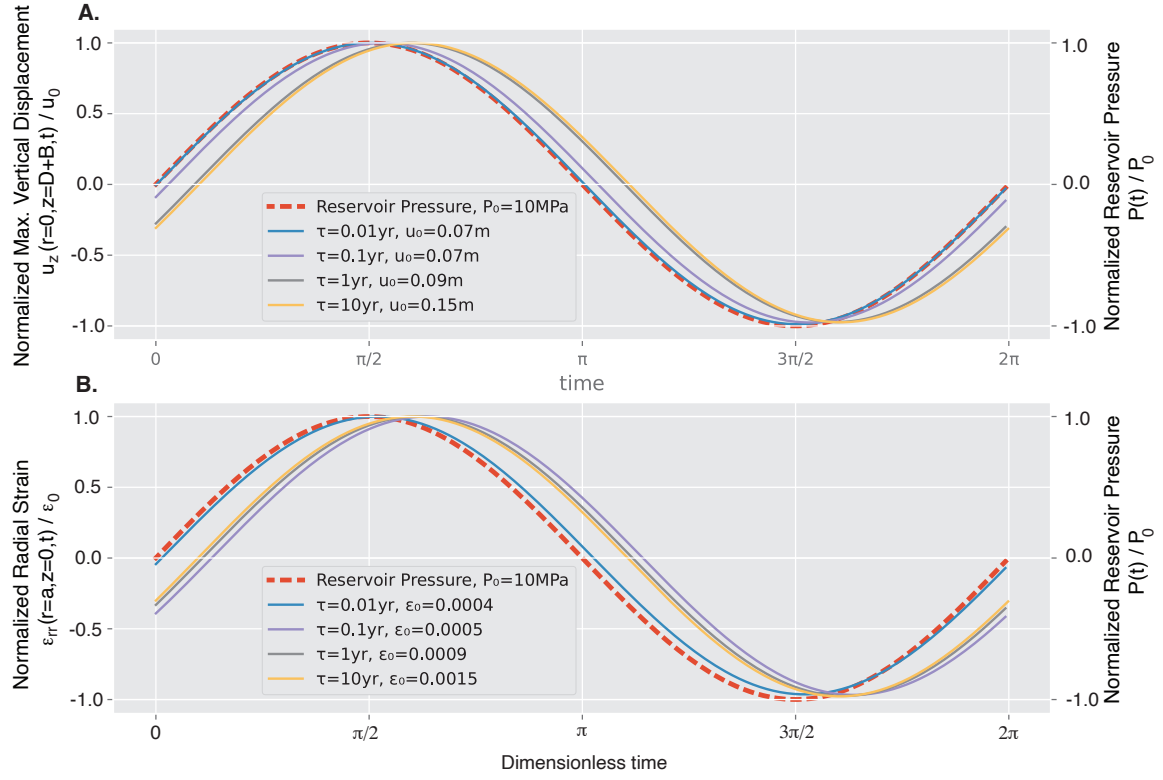


Fig. 9 Temporal evolution (time non-dimensionalized by τ) associated with non-constant coefficient simulations at select forcing periods. Colored curves correspond to different forcing periods and normalization amplitudes u_0, ϵ_0 , dashed curves show pressure normalized by P_0 . **A.** Normalized maximum vertical surface displacement. In dimensional time, peak vertical surface displacement for $\tau = 0.01, 0.1, 1, 10$ years occurs 10.0 min, 12.7 hr, 17.6 days, and 6.3 months after peak reservoir pressure, respectively, associated with phase lags $\phi\{u_z(r = 0, z = D + b, t|P(t))\} = 0.012, 0.091, 0.303$ and 0.331 radians. **B.** Normalized radial strain at the cavity wall, illustrating that phase offset of deformation from pressure forcing varies spatially through the domain.

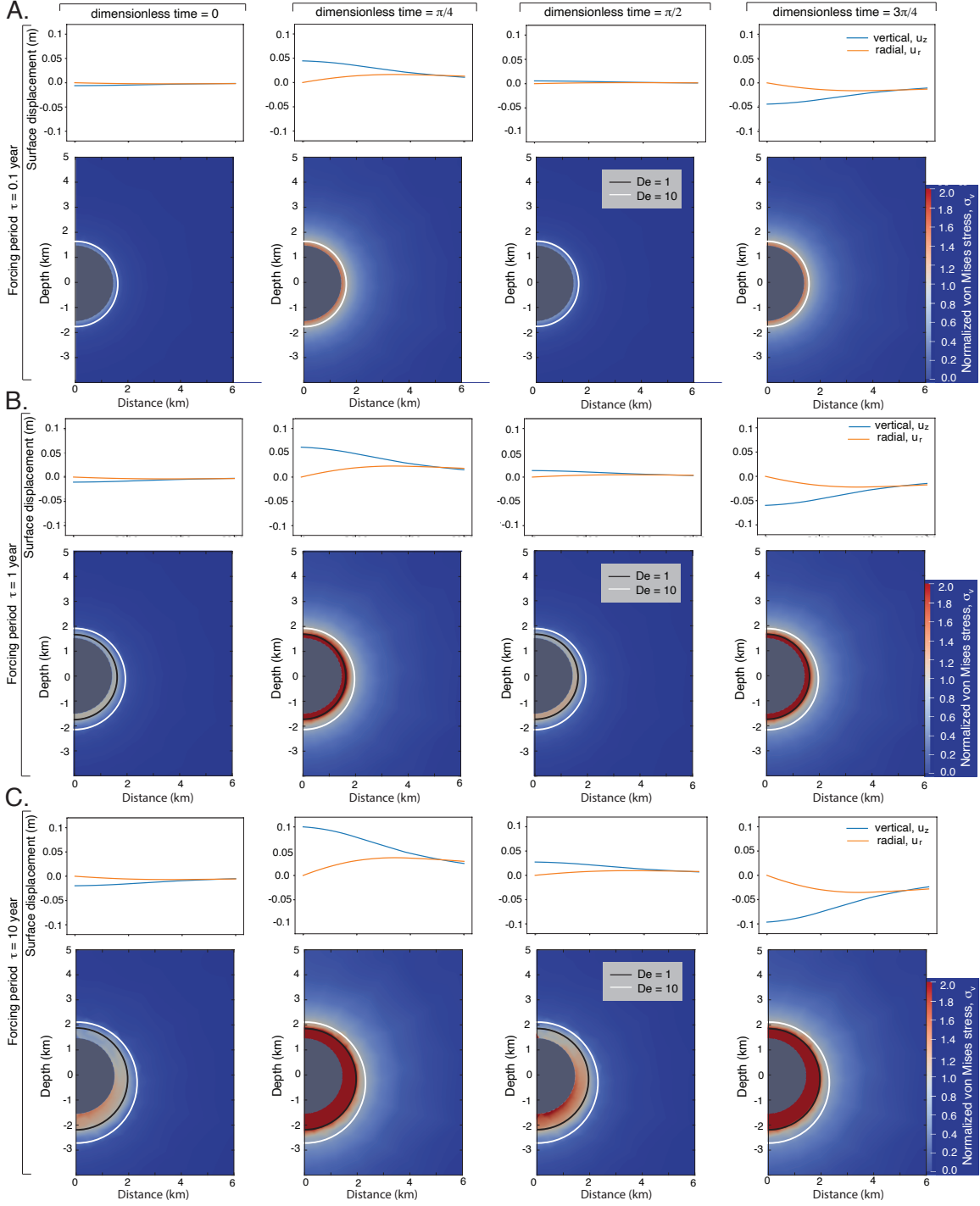


Fig. 10 Spatial pattern of surface displacements u_z, u_r (top lines) and subsurface distribution of von Mises stress σ_v (bottom colors, normalized by $P_0 = 10$ MPa) for dimensionless times $0, \pi/4, \pi/2, 3\pi/4$ during a pressure cycle. Black contour is $De = 1$, white contour is $De = 10$, illustrating that a local Deborah number contour approximates the spatial region of elevated deviatoric stress and viscous strain around the chamber. **A.** Forcing period $\tau = 0.1$ yr, max $\sigma_v = 20.9$ MPa. **B.** Forcing period $\tau = 1$ yr, max $\sigma_v = 42.2$ MPa. **C.** Forcing period $\tau = 10$ yr, max $\sigma_v = 100.7$ MPa. Supplemental movies S1-S3 show time evolution of these simulations in more detail.

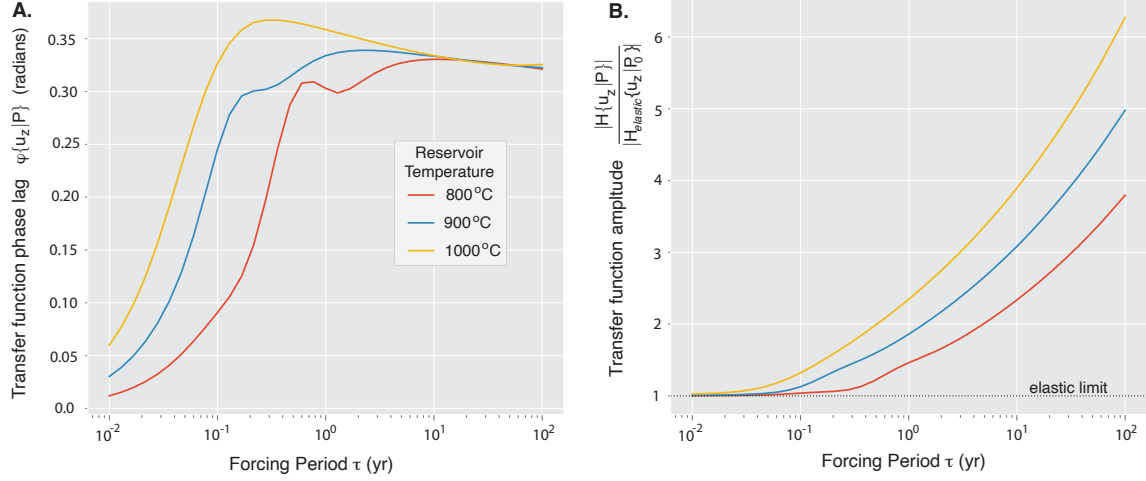


Fig. 11 Transfer function between reservoir pressure and maximum vertical surface displacement $H\{u_z(r=0, z=D+b, t)|P(t)\}$ as a function of sinusoidal pressure forcing period τ . Colored curves correspond to different reservoir temperatures, each case assumes surface temperature $T_s = 0^\circ\text{C}$ and background geothermal gradient $\alpha = 20\text{ C/km}$. **A.** Phase lag $\phi\{u_z(r=0, z=D+b, t)|P(t)\}$. **B.** Amplitude $|H\{u_z(r=0, z=D+b, t)|P(t)\}|$ normalized by the corresponding variable coefficient elastic case at each temperature. For the three reservoir temperatures explored here, $|H_{elastic}\{u_z(r=0, z=D+b)|P_0\}| = 6.509 \times 10^{-9}, 6.822 \times 10^{-9}, 7.163 \times 10^{-9}\text{ m/Pa}$ for $T_c = 800, 900, 1000^\circ\text{C}$ respectively.

coefficient elastic limit. That is, elastic parameters are computed using a thermal profile but viscosity $\eta = 1 \times 10^{34}\text{Pa}\cdot\text{s}$. Then this variable coefficient elastic problem is simulated and a transfer function $H_{elastic}$ is computed from the output.

The transfer function curves in Figure 11 have more complex structure than their constant coefficient counterpart in Figure 8. First, the phase lag $\phi\{u_z(r=0, z=D+b, t)|P(t)\}$ is non-monotonic, with two local maxima superimposed on a sigmoidal increase from 0 to ~ 0.3 radians over three orders of magnitude in forcing period. The larger of these is a global maximum for the range of forcing periods we explored (100 years maximum), likely reflecting the spatially variable temperature field and resulting finite region around the chamber in which viscous strains occur. Increasing the reservoir temperature from 800°C to 1000°C shifts this global maximum as well as the sigmoidal uptick in phase lag to shorter periods, which suggests that the local maxima are due in part to an expanded viscous shell around the reservoir (i.e., larger region where $De < 10$). We expect that the shape of this phase lag curve as metric of viscoelastic response likely depends on spatial rheologic structure, boundary conditions, and chamber geometry, although a parameter exploration is out of the scope of this study.

The apparent global maximum seen in the phase lag in Figure 11 is not mirrored by the amplitude of displacements. Relative to the elastic limit transfer function amplitude show a continuous increase in maximum displacements at increasing τ , mirrored by the spatial pattern of u_z and u_r in Figure 10. There is an inflection point that corresponds to the local minimum in ϕ for the lower reservoir temperatures, but viscous amplification is otherwise a monotonically increasing function of τ , with amplification factors at 100 yr forcing period $\sim 3.8\times$, $\sim 5\times$ and $\sim 6.3\times$ for 800°C, 900°C, and 1000°C chamber temperatures. At small τ the amplification factor is asymptotic to the variable coefficient elastic limit (dashed line) in all cases.

3.6 Discussion

This work makes two primary contributions. First, we develop a rigorous numerical framework based on a high-order finite element method for the computation of viscoelastic deformation and stress around axisymmetric magma reservoirs. Second, we study a particular problem - sinusoidal pressurization/depressurization of a spherical reservoir in a half-space - and demonstrate how surface deformation patterns are frequency dependent. This section is organized into a discussion associated with each contribution as they relate to the phenomenology of viscoelastic deformation around volcanoes.

3.6.1 Computational Considerations for Time-evolving Magmatic Systems.

Numerical modeling of viscoelastic deformation over long timescales requires careful consideration of computational issues. We derived conditions on the time step, which guarantees stability of the aging law when using the Maxwell model and showed that the numerical solution converges to the exact solution at the theoretical rates of convergence in both space and time. However, in practice, even these 2D simulations are computationally expensive because a system of equations (the discretized equilibrium equation) must be solved at each time step, and this constitutes the bulk of the computational load. We perform a direct solve of the system while it is still possible to hold the matrix factorization in system memory. For larger problems (e.g. in 3D or with larger domains sizes or if a finer spatial resolution is required), matrix-free iterative methods on parallel machines would be necessary [110]. Furthermore, if the relevant time scale of interest is the forcing period τ , which can be much longer than the minimum viscous relaxation time η/μ (so that $De \ll 1$), the problem can become arbitrarily numerically stiff: very small time steps are required for numerical stability, much smaller than that required to accurately resolve the sinusoidal pressure forcing.

To address this corresponding computational burden, an implicit time stepping scheme (such as backward Euler) would need to be applied, or alternative schemes such as splitting algorithms [111]. For problems in which total strains are large (e.g., dominated by viscous flow) it may also be advantageous to reformulate the governing equations in terms of split viscous and elastic strain rates (rather than strains), as is commonly done in mantle dynamics models e.g., [112]. A disadvantage of this approach is that elastic stresses are less explicitly resolved, which is not acceptable for the magmatic application. Still, one drawback of our method is that it is not robust in the incompressible limit ($\nu = 0.5$). More sophisticated locking-free mixed finite element techniques (e.g., [113]) could be employed to solve the equilibrium equations stably in the incompressible limit, a potential necessity in fully coupled fluid-solid magmatic models. Codes developed for large-scale geodynamic applications commonly include compressible fluid but incompressible solid mechanics e.g., [114]. This difference in approach implies that extensions of our computational framework to a broader range of problems might require further numerical developments.

The inclusion of boundary tractions (to represent background tectonic stress, for example) can be explored here directly by setting specific values of the boundary data. Axisymmetric topography at the surface or at depth can be included by modifying the axisymmetric domain geometry. Complex time-evolving forcing can be included so long as the highest frequency is resolved by the timestep, as we demonstrate in the next section. But highly multiscale time evolution, such as might be expected for pressure at the reservoir wall over eruption cycles [54], may require adaptive time-stepping techniques to integrate efficiently through regions of both slow and fast evolution. Similar challenges arise in the modeling of long-term earthquake cycles e.g., [115], and similar timestepping approaches could be leveraged for simulating volcanic activity.

3.6.2 Frequency Dependent Magmatic Deformation. We have studied here a magma chamber problem that, while simplified in some respects, has a strong basis in past observations and represents a template for future advances. In the elastic limit, corrections for less idealized geometry and material heterogeneity are known e.g., [116], and elastic parameter trade-offs have been explored to some extent e.g., [117], [118]. But viscoelastic behavior is far less well understood. Case studies have demonstrated important trade-offs in geometry, constitutive law, and thermal state, as well as complications associated with time-dependent rheology e.g., [59], [81], [85], [119]. But general time-dependence introduces significant complexities.

The cyclic forcing studied here represents a powerful framework to explore phenomenology of transient magma chamber deformation. While magma pressure histories are not generally sinusoidal, linear viscoelasticity (in any form, not just the Maxwell model) implies that arbitrary forcing histories may be constructed through appropriate superposition. Our initial example (Equation (3.3.7)) and analysis in section 3.4.2 details how the transfer function can be used to model general signals. We illustrate this approach with three examples below, with additional comments in B.2.

First, consider a reservoir pressure history (the input signal) defined by the ramp function of Section 3.3.3 whose Fourier series is given by Equation (3.3.7). Then if the transfer function for each component is $H(i\omega_n) = |H(i\omega_n)|e^{-i\phi_n}$ then the output signal $y(t)$ can be expressed in terms of its Fourier series

$$y(t) = \sum_{n=1}^{\infty} d_n \sin(\omega_n t - \phi_n) \quad (3.6.1)$$

with coefficients

$$d_n = |H(i\omega_n)|b_n, \quad (3.6.2)$$

i.e. the coefficients of the input signal, scaled by the transfer function amplitude $|H|$. This example demonstrates that sequences of impulsive pressure changes (such as eruptions or magma injections) that are non-harmonic in time can still be characterized with the framework developed here.

As a second example, if the pressure history is given by a unit impulse at $t = t_0$, namely

$$P(t) = P_0 \delta(t - t_0), \quad (3.6.3)$$

then (3.4.15) implies that the output signal is simply

$$y(t) = h(t - t_0), \quad (3.6.4)$$

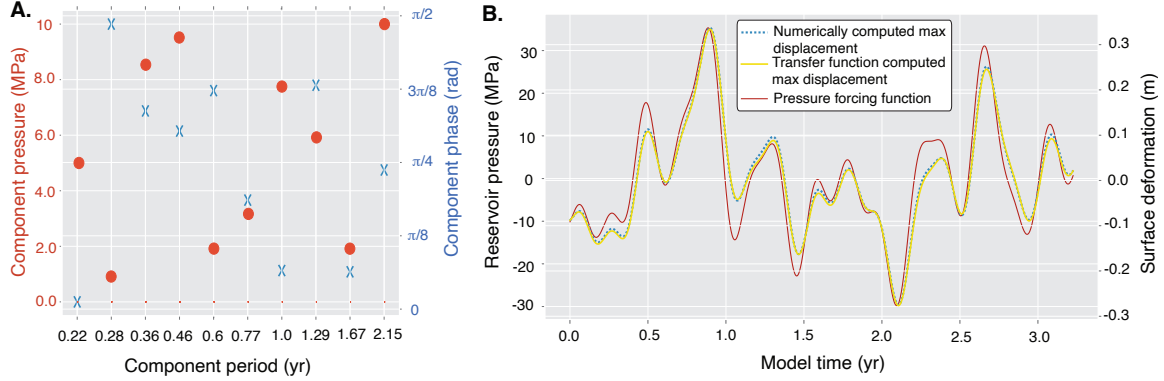


Fig. 12 **A.** Amplitudes and phases of reservoir pressure forcing function, Equation (3.4.24). **B.** Reservoir pressure timeseries (red curve, left y axis) along with numerically computed maximum surface displacement (dashed blue curve, right y axis) and analytic prediction based on the transfer function (yellow curve), Equation (3.4.25).

i.e. the system impulse response. This pressure history represents a simple model for sudden pressure perturbation e.g., [83]. The implied ground deformation in this case is the impulse response function of the magma chamber/host rock system.

These examples demonstrate the transfer function approach in a forward modeling framework. Inversion of magmatic pressure histories from ground motions, a common scenario since reservoir pressure is generally unknown, by extension involves seeking weights in the frequency-domain for forcing periods represented in Figure 11 to match general time-dependent deformation data. To demonstrate this explicitly, we present a third example in which we construct a non-harmonic input pressure signal by summing sinusoids at a subset of forcing frequencies explored in Figure 11 with random phase and amplitude (assuming an 800°C chamber representing a lower bound to the viscoelastic response) corresponding to Equation (3.4.24). Weights and phases are displayed in Figure 12.A. We compute the output signal from Equation (3.4.25) and show that the predicted surface deformation matches the numerically computed output (Figure 12.B). Numerical displacements shown here are after a spin-up to make sure the output is in steady state with the input.

Outputs of interest are thus found given knowledge of the transfer function. Of course, in reality this transfer function is unknown and would need to be computed as part of an inversion. Further studies will be needed to quantify the variability of the transfer function as control parameters are

varied. This will determine the sensitivity of phase lag and amplitude spectrum to rheologic model, chamber geometry, and temperature structure.

Figure 12.B also demonstrates the non-trivial impact of frequency-dependent phase lag and amplitude on ground deformation. Even though a relatively narrow range of frequencies is present in the forcing function ($2\pi/\omega_k = \tau_k \sim 0.2 - 2$ yr in Equation (3.4.24)), we see that shorter period forcing generates in-phase ground displacements, while longer period ground motions are out of phase with chamber pressure. These effects would be amplified for warmer (more viscous) host rocks and longer forcing periods, and should be observable in geodetic timeseries with several day resolution (phase lag associated with 1 year forcing period from Figure 11 is ~ 18 days). We also see that the ground displacement amplitude is a function of frequency as predicted from the transfer function. It is not simply proportional to the pressure as expected from linear elasticity [60], and reflects the amplitudes of each component period shown in fig 12.A scaled by the transfer function.

An interesting challenge implied by our analysis with respect to observations however is how to find initial conditions. Our time-dependent steady-state (purely oscillatory) implicitly starts from a unstressed state, but as illustrated through 1D analysis (Section 3.4) the initial strain determines the equilibrium position around which steady viscoelastic oscillations occur. In the 2D variable coefficients case the choice of initial strain that will result in a particular chamber size (or geometry) is less trivially found; equilibrium magma chamber volume is not a parameter but rather a model outcome. From a geophysical perspective, this implies that absolute stress histories are needed to interpret general surface displacement timeseries at volcanoes, and could play an important role in eruption cycles as it does for earthquake cycles e.g., [120].

Another important implication of this model is that the volume of crustal rock around the chamber experiencing viscous strain over a chamber pressure cycle depends on the frequency of forcing. As demonstrated by Figure 8, $De = 10$ effectively marks the onset of viscous host response to cycling pressure forcing. Figure 10 extends this to variable coefficients, suggesting that $De \approx 10$ effectively bounds the region in which significant deviatoric shear stresses (as measured by σ_v in excess of P_0) occur.

We suggest that the frequency-dependent $De \approx 10$ contour represents an effective outer edge to the viscoelastic “shell” at a given frequency of forcing. This shell has been largely considered fixed in size by previous models for viscoelastic magma chamber mechanics e.g., [67], [68], [70]–[72],

[83]. Our model demonstrates that viscoelastic aureole size for a steady temperature distribution depends on the time history of reservoir stress - like equilibrium reservoir size, it is a transient model output.

3.6.3 Implications for Transcrustal Magmatic Systems. Magma reservoirs that feed volcanic eruptions likely sit near the top of transcrustal magma transport networks characterized by high temperatures and partial melt [58]. Some of this magma accumulates episodically into high melt fraction reservoirs such as we model here. But it is to be expected that, as transcrustal magma transport networks mature, a significant fraction of the crust is heated and remains hot for extended periods of time. What are the implications of this rheological structure for ground deformation?

We can begin to answer this question by noting that the bulk crustal rheology of magma storage zones as expressed by surface deformation depends on frequency of forcing, as it does on the spatial structure of melt and temperature [121]. This has been long recognized for crustal rheology in other settings [91], [122]. But volcanoes offer a particularly interesting case for exploring crustal rheology, because different histories of heating – all else equal – will have distinct deformation frequency response curves (transfer functions).

Figure 13 plots the $De = 10$ contour representing onset of viscous mechanical response for different pressurization periods, from 0.1 to 1000 years. We then consider end member steady state thermal regimes: chamber boundary temperature of $T_c = 800^\circ\text{C}$ and 1200°C , and geothermal gradient of $\alpha = 20^\circ\text{C}/\text{km}$ and $35^\circ\text{C}/\text{km}$. In the cold extreme (Figure 13A), we see that viscoelastic behavior is confined to a shell around the chamber in all but 1000 year forcing. This is consistent with commonly used models of isolated magma chambers. At long forcing periods however the mid/lower crust is activated and starts to creep, defining a mid-crustal brittle-ductile transition that depends on background geothermal gradient. In the hot extreme (Figure 13D), we see that viscoelastic response of the near-chamber region extends continuously into the mid-crust for forcing periods as low as 10 years. This defines a spatially coherent viscous domain induced by magmatic heating [78], activated by long-period forcing.

While we leave further exploration of this to future work, we note that some of the structure seen in phase lag variations in Figure 11 likely reflect changes to the shape as well as volume of the viscous near-chamber region. It is notable that significant sensitivity of viscoelastic response to forcing period and variations in thermal structure in the 0.1 – 10 year range, where geodetic

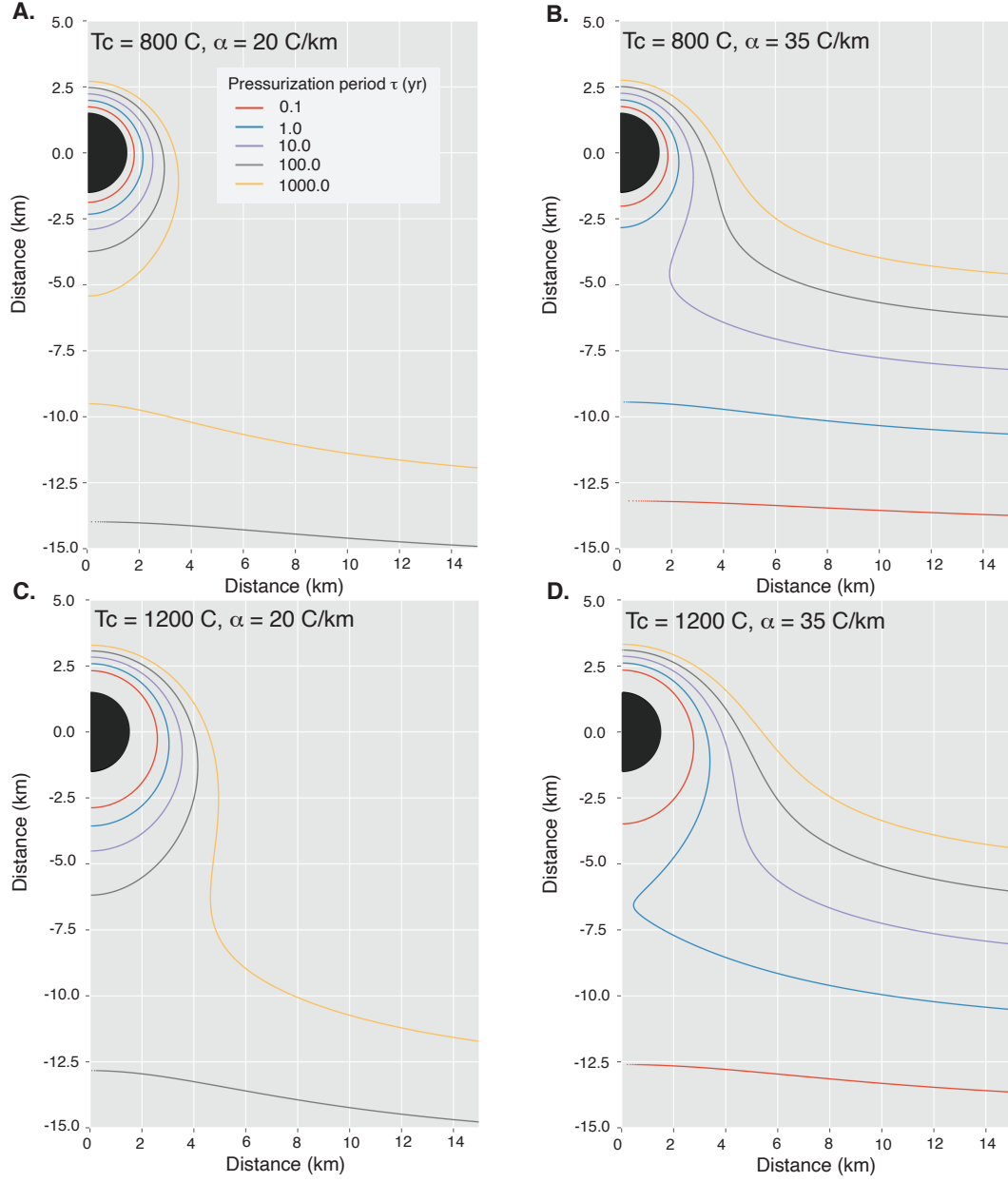


Fig. 13 Spatial regions associated with a local Deborah number $De = 10$ for varying periods τ of the chamber pressure forcing function (colored curves), illustrating end member thermal regimes. Magma reservoir is black semi-circle in all panels. **A.** Reservoir temperature $T_c = 800^\circ\text{C}$ with geothermal gradient $\alpha = 20^\circ\text{C/km}$. **B.** Reservoir temperature $T_c = 800^\circ\text{C}$ with geothermal gradient $\alpha = 35^\circ\text{C/km}$. **C.** Reservoir temperature $T_c = 1200^\circ\text{C}$ with geothermal gradient $\alpha = 20^\circ\text{C/km}$. **D.** Reservoir temperature $T_c = 1200^\circ\text{C}$ with geothermal gradient $\alpha = 35^\circ\text{C/km}$.

observations are increasingly common. Because magma transport is unsteady at many scales, ground deformation in volcanic regions will likewise include contributions from viscoelastic deformation defining the crustal thermo-rheologic footprint of magmatism on a range of timescales.

Open Research

Software consists of Python code developed on top of the free and open source multi-physics library NGSolve [93] and the accompanying mesh generator [94]. All source code is freely available in the public repository [95].

CR, BAE and LK were supported by NSF grant EAR- 2036980. LK also acknowledges NSF grant 1848554. BL and JG were supported by NSF grant DMS-1912779. This work benefited from access to the University of Oregon high performance computer Talapas and the COEUS cluster at the Portland Institute for Computational Science. We thank James Hickey, an anonymous reviewer, and the editor for comments and suggestions that improved the paper considerably. The authors acknowledge useful discussions with Yang Liao and Ben Holtzman.

CHAPTER IV

PINN INVERSION

This chapter contains work published in the journal of Computer Methods in Applied Mechanics and Engineering in July 2024. The published work was co-authored by Dr. Brittany A. Erickson.

Dr. Brittany A. Erickson and I worked together on the project’s conceptualization, methodology, and formal analysis. I was first author having developed and validated the model and code for the project. I conducted the investigation, data collection.

4.1 Context and Motivation

Faults are home to a vast spectrum of event types, from slow aseismic creep, to slow-slip to megathrust earthquakes followed by postseismic afterslip. The Cascadia subduction zone in the Pacific Northwest, for example, hosts several types of slow earthquake processes including low (and very low) frequency earthquakes, non-volcanic tremor (NVT) and slow-slip events (SSE) [123], but also large, fast earthquakes, the last of which was a magnitude ~ 9 in the year 1700 [124]. Understanding the physical mechanisms for such diversity of slip styles is crucial for mitigating the associated hazards but major uncertainties remain in the depth-dependency of frictional properties at fault zones, which affect fault locking and therefore rupture potential [125], [126]. Direct observations of earthquake nucleation and propagation are few and yet the next decade will likely see an unprecedented increase in indirect, surface observations that could be integrated into modeling efforts [127].

Traditional numerical approaches for solving the partial differential equations (PDE) governing earthquake processes (e.g. finite difference methods) have seen incredible growth in the past century, in particular in terms of convergence theory and high-performance computing. Traditional methods employ a mesh (either a finite number of grid points/nodes or elements) and a range of time-integration schemes in order to obtain an approximate solution whose accuracy depends directly on the mesh size (with error decreasing with decreasing node spacing or element size). This mesh dependency introduces limitations when high resolution is needed, and while traditional methods are well-suited for solving forward problems, solving inverse problems require additional machinery and can be prohibitively expensive [128], [129]. In addition, traditional methods require models to

specify initial and boundary conditions in order to establish a well-posed problem, even when such conditions are unknown.

Machine learning (ML), on the other hand, excels in the presence of large data and is an actively growing field in seismology, with applications ranging from earthquake early warning (EEW) to ground-motion prediction, see for example Kong, Trugman, Ross *et al.* [130], Lin, Melgar, Thomas *et al.* [131] and Kubo, Naoi and Kano [132] and references therein. However, not all ML methods incorporate rigorous physics, and purely data-driven models can predict physically unrealistic outcomes due to observational bias or extrapolation [133]. A new Deep Learning technique has recently emerged called the Physics-Informed Neural Network (PINN), which seamlessly integrates sparse and/or noisy data while ensuring that model outcomes satisfy rigorous physical constraints. PINNs do not outperform traditional numerical methods for forward problems (except in high-dimensional settings) [20], but they offer advantages over traditional numerical methods in that both forward and inverse problems can be solved in the same computational framework and solutions may be learned even on ill-posed problems [134]. However, the majority of PINN applications are currently limited to simple mechanical models, forward problems and/or do not incorporate real-world observations [135]–[137]. Here we introduce a new, physically-rigorous modeling framework for both forward and inverse problems that can be integrated with observational data in order to better understand earthquake fault processes. This Deep Learning approach will vastly expand our computational abilities to explore and infer relevant parameter spaces responsible for slip complexity, and the conditions that enable the world’s largest earthquakes.

Though the PINN framework lacks the robust error analysis that comes with traditional methods, a large number of publications have emerged since ~ 2017 which aim to customize PINNs through the use of different activation functions, gradient optimization techniques, neural network architecture, and loss function structure [16]. Careful formulation of the loss function using the weak form of the PDE have been proposed for constructing deep learning analogues of Ritz [138] and Galerkin [28], [29], [139] methods which use numerical quadrature to reduce the order of the PDE resulting in a simpler learning problem [31], [32]. In tandem, statistical learning theory has been used to deduce global error bounds for PINNs in terms of optimization error, generalization error, and approximation error [43]. For wide but shallow networks utilizing hyperbolic tangent activation functions, the approximation error has been shown to be bounded over Sobolev spaces

[46]. Bounds on PINN generalization error have been derived for linear second-order PDE [48] (later extended to all linear problems [45]) and some specific cases like Navier-Stokes [49]. Moreover, the abstract framework for PINNs can leverage stability of a PDE to provide conditions under which generalization error is small whenever training error is small for both forward and inverse problems [50], [51]. More recently, a PINN-specific optimization algorithm has achieved markedly improved accuracy over other optimization algorithms by incorporating a PDE energy into the backpropagation step [140]. In addition to this rapid framework development, PINNs have been shown to perform well on a variety of physical problems like Navier-Stokes [21]–[23], convection heat transfer [24], solid mechanics, [25], [26] and the Euler equations [27]. By constraining the network to obey physical laws, PINNs are able to operate in a small data regime and continuously approximate solutions to PDE[3] which allows for model verification across multiple domain resolutions without needing to generate a new approximation each time.

In this work we focus specifically on rate-and-state friction [e.g. 141], an experimentally-motivated, nonlinear friction law capable of reproducing a wide range of observed earthquake behaviors [142] and whose increasing use in dynamic rupture and earthquake cycle simulations has motivated the recent community benchmark exercises described in Harris, Barall, Archuleta *et al.* [143] and Erickson, Jiang, Barall *et al.* [98]. A better understanding of the depth-dependency of rate-and-state parameters - which have a direct correlation to fault locking and seismic rupture potential - is a fundamental task [125], [126]. To address this task we develop a multi-network PINN for modeling a vertical, strike-slip frictional fault embedded in an elastic half-space, and consider deformation in both 1D and 2D. The paper is organized as follows: In section 4.2 we first provide an overview of the physics-informed deep learning framework and PINN architecture for general initial-boundary-value problems, in order to best describe the implementation to our application problem. In section 4.3 we provide specific details of the PINN framework applied problems, first illustrated in 1D with an example forward problem, then further developed to include inverse problems in 2D. In section 4.4 we report details of our optimal network architecture and training methods, verifying our methods with a manufactured solution to ensure accuracy of our inversions. We conclude with a summary and discussion of future work in section 4.5.

4.2 Physics-Informed Deep Learning Framework

The physics governing motion in many applications in science and engineering give rise to partial differential equations (PDE) where analytic solutions can be difficult to obtain (due, e.g. to complex material properties, boundary conditions, geometry) and we commonly turn to numerical methods. The Physics-Informed Neural Network (PINN) is a deep learning (DL) framework for approximating solutions to PDEs. Though this DL framework lacks the robust mathematical theory of the traditional methods, it shows particular promise in solving problems for which traditional numerical methods are ill-suited[3]. The DL framework produces a closed, analytic form for the solution, which is continuous and defined at every point in the domain allowing one to evaluate the solution “off-grid” without having to resolve the PDE (i.e. it is mesh-free) [3]. In addition, both forward and inverse problems can be solved in the same computational setting, as will be shown in subsequent sections.

4.2.1 Feed-forward Deep Neural Networks. PINNs are extensions of a general feed-forward neural network. We let $\mathbf{x} \in \mathbb{R}^n$ and define a weighting matrix $W \in \mathbb{R}^{m \times n}$ and bias vector $b \in \mathbb{R}^m$. A single hidden layer of a neural network can be expressed as

$$\ell(\mathbf{x}; \theta) = \varphi(W\mathbf{x} + b), \quad \text{where } \theta = (W, b), \quad (4.2.1)$$

and φ is a known (nonlinear) activation function. Deep neural networks are obtained by repeated composition of hidden layers [15].

We let positive integer L be the deep neural network depth (i.e. number of hidden layers) and let $\{\varphi_i\}_{i=1}^L$ be a collection of activation functions along with a sequence of trainable network parameters $\{\theta_i\}_{i=0}^L$ where $\theta_k = (W_k, b_k)$ for each $0 \leq k \leq L$. Here we assume each layer consists of m neurons but this simplifying assumption may be omitted so long as one ensures that each weight and bias are of the correct dimension for matrix multiplication and addition, respectively. If $\mathbf{x} \in \mathbb{R}^n$ is the network input and network parameters $(W_0, b_0) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$, $(W_k, b_k) \in \mathbb{R}^{m \times m} \times \mathbb{R}^m$ for $1 \leq k < L$, and $(W_L, b_L) \in \mathbb{R}^{d \times m} \times \mathbb{R}^d$, then the recursive composition

$$\ell_0 = \mathbf{x}, \quad (4.2.2a)$$

$$\ell_k = \varphi_k(W_k \ell_{k-1} + b_k) \quad \text{for } 0 < k < L, \quad (4.2.2b)$$

defines a feed-forward, deep neural network $\mathcal{N} : \mathbb{R}^n \rightarrow \mathbb{R}^d$ (of depth L and width m) with network parameters θ by $\mathcal{N}(\mathbf{x}; \theta) = W_L \ell_{L-1} + b_L$.

4.2.2 PINN Architecture. The PINN architecture is defined by extending the feed-forward deep neural network to enforce physical conditions set by an initial-boundary value problem (IBVP). For simplicity in the notation (as well as generality) we define this extension for a general IBVP, with specific details for our application given in the subsequent section. Consider the general operator form of an IBVP given by

$$\mathcal{L}[\mathbf{u}; \lambda] = \mathbf{k}, \quad \text{in } \widehat{\Omega}, \quad (4.2.3a)$$

$$\mathcal{B}[\mathbf{u}; \lambda] = \mathbf{g}, \quad \text{on } \partial\widehat{\Omega}, \quad (4.2.3b)$$

where $\widehat{\Omega} \subseteq \mathbb{R}^n$, with boundary (including internal interfaces, like faults) $\partial\widehat{\Omega}$. Vector $\mathbf{u}$ is the unknown solution and $\mathbf{g}$ is boundary data. The source term $\mathbf{k}$ encompasses external and internal body forces, and $\mathcal{L}$, $\mathcal{B}$ are differential and boundary operators parameterized by λ .

The PINN first approximates the solution to the IBVP (4.2.3) using a feed-forward, deep neural network, i.e. we assume $\mathbf{u}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \theta)$ and consider N_∂ many boundary points $\{\mathbf{x}_\partial^i, \mathbf{g}^i\}_{i=1}^{N_\partial}$ where $\mathbf{g}^i = \mathbf{g}(\mathbf{x}_\partial^i)$ for each $1 \leq i \leq N_\partial$, and a set of $N_{\widehat{\Omega}}$ internal collocation points $\{\mathbf{x}_{\widehat{\Omega}}^i, \mathbf{k}^i\}_{i=1}^{N_{\widehat{\Omega}}}$ with $\mathbf{k}^i = \mathbf{k}(\mathbf{x}_{\widehat{\Omega}}^i)$ for $1 \leq i \leq N_{\widehat{\Omega}}$. Throughout this work, collocation points are generated from uniform random sampling. From (4.2.3) we construct an objective function MSE (based on the mean-square error) given by

$$MSE(\theta) = MSE_{\widehat{\Omega}}(\theta) + MSE_\partial(\theta), \quad (4.2.4)$$

where

$$MSE_{\widehat{\Omega}}(\theta) = \frac{1}{N_{\widehat{\Omega}}} \sum_{i=1}^{N_{\widehat{\Omega}}} |\mathcal{L}[\mathcal{N}; \lambda](\mathbf{x}_{\widehat{\Omega}}^i; \theta) - \mathbf{k}^i|^2, \quad (4.2.5a)$$

$$MSE_\partial(\theta) = \frac{1}{N_\partial} \sum_{i=1}^{N_\partial} |\mathcal{B}[\mathcal{N}; \lambda](\mathbf{x}_\partial^i; \theta) - \mathbf{g}^i|^2, \quad (4.2.5b)$$

are the contributions to the total loss from the PDE (4.2.3a) and boundary conditions (4.2.3b), respectively, corresponding to misfit of the network with data. Because our objective function (4.2.4) consists of multiple competing loss functions it is referred to as a multi-objective loss. Solving (4.2.3) is done by minimizing (4.2.4) with respect to the network parameters θ , typically done via an optimization algorithm that approximates the set of solution parameters θ^* such that $\theta^* = \arg \min_\theta MSE(\theta)$. Figure 14 gives a schematic representation of the objective function (4.2.4). Note that for the rest of this work with omit the notation specifying MSE dependency on network parameters θ and assume this is understood.

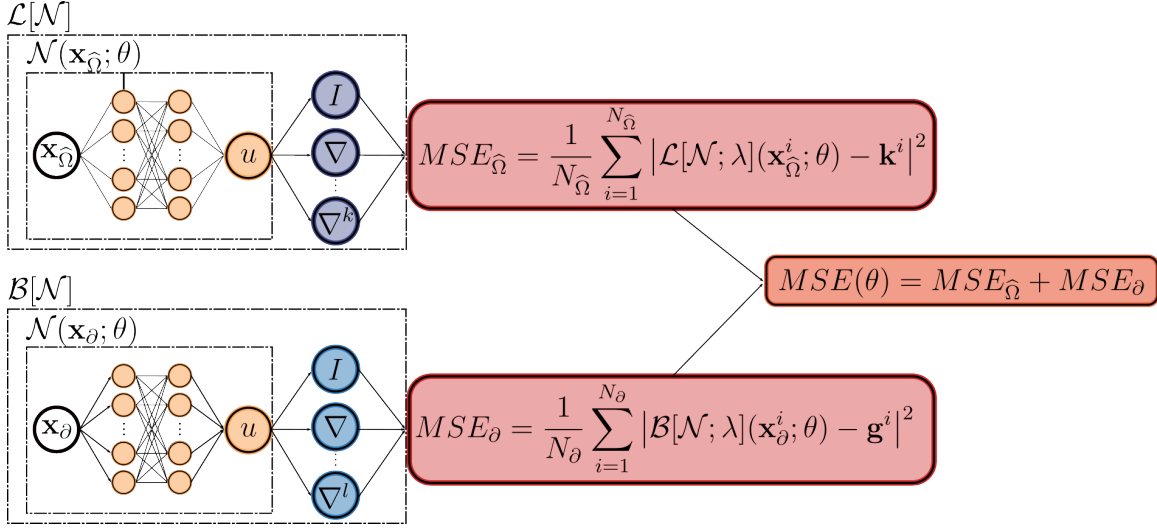


Fig. 14 A schematic of the PINN framework for solving the general boundary value problem from equation (4.2.3). Displacement approximation network $\mathcal{N}$ is trained on interior and boundary subdomains which are governed by operators $\mathcal{L}$ and $\mathcal{B}$, respectively.

4.3 Learning Problems for Earthquakes on Rate-and-State Faults

In order to best illustrate the PINN computational set-up, we begin by considering a relevant problem in 1D, in order to introduce our methodology in a simplified framework. Here we focus on the forward problem; details of the inverse problem are included in the subsequent section on the 2D application problem, which also includes a schematic diagram motivating the governing equations.

4.3.1 1D Illustration. The solid Earth is governed by momentum balance and a constitutive relation defining the material rheology; in this work we assume elastic material properties. These assumptions give rise to the elastic wave equation, namely

$$u_{tt} = c^2 u_{xx} + s(x, t), \quad x \in [0, 1], t \geq 0, \quad (4.3.1)$$

where $u(x, t)$ is the Earth's material displacement, $c = \sqrt{\mu/\rho}$ is the wavespeed where ρ and μ are the the density and shear modulus, and $s(x, t)$ accounts for external and/or body forces. Here we use subscripts to denote derivatives with respect to the subscript variable. Assumed initial and

boundary conditions are given by

$$u(x, 0) = u_0(x), \quad (4.3.2a)$$

$$u_t(x, 0) = v_0(x) \quad (4.3.2b)$$

$$-\mu u_x = F + g_0(t), \quad x = 0, \quad (4.3.2c)$$

$$Zu_t + \mu u_x = g_1(t), \quad x = 1, \quad (4.3.2d)$$

where $Z = \sqrt{\mu\rho}$ is the shear impedance. If boundary data $g_0(t) = g_1(t) = 0$, equation (4.3.2c) corresponds to the requirement that fault shear stress be equal to frictional strength F , and (4.3.2d) allows waves to freely exit the domain.

In this work we consider rate-and-state dependent friction (RSF), an experimentally-motivated, nonlinear friction law, often used in earthquake simulations for its ability to reproduce a wide range of observed seismic and aseismic behaviors [142], [144], [145]. In this context, the frictional strength is given by

$$F = \bar{\sigma}_n f(V, \psi), \quad (4.3.3)$$

where V is the slip rate (jump in velocity across a fault interface), $\bar{\sigma}_n$ is the effective normal stress (normal stress minus pore fluid pressure), and f is a friction coefficient given by

$$f(V, \psi) = a \ln \left(\frac{V}{V_0} \right) + \psi, \quad (4.3.4)$$

where V_0 is a reference slip rate and material parameter a is the “direct effect” [146]. State variable ψ evolves from some initial value ψ_0 according to its own evolution law

$$\frac{d\psi}{dt} = G(V, \psi) + h(t), \quad (4.3.5a)$$

$$\psi(0) = \psi_0 \quad (4.3.5b)$$

where we assume the aging law, i.e. $G(V, \psi) = (bV_0/D_c) \exp(\frac{f_0 - \psi}{b} - \frac{|V|}{V_0})$, which allows state to evolve even in the absence of slip [142], and $h(t)$ allows for the incorporation of additional source data. Here D_c is the characteristic slip distance, f_0 is a reference friction coefficient for sliding at speed V_0 and material parameter b captures time-dependent “evolution” effects [146].

The initial boundary value problem (IBVP) formed by the PDE (4.3.1), specified initial/boundary conditions (4.3.2) and state-evolution equation (4.3.5a) correspond to specific differential and boundary operators $\mathcal{L}$ and $\mathcal{B}$ defined in (4.2.3), for example, here $\mathcal{L} = \partial^2/\partial t^2 - c^2\partial^2/\partial x^2$. Assuming all parameters of the IBVP are known, one can solve the forward problem

for the unknown material displacement $u(x, t)$ and state variable $\psi(t)$. We define neural networks $\mathcal{N}(x, t; \theta) \approx u(x, t)$ and $\mathcal{N}^\psi(t; \theta_\psi) \approx \psi(t)$ to approximate the exact solutions. The PDE, boundary and initials conditions all give rise to a term in the objective function. We refer to these terms as the component loss functions and define them as

$$MSE_\Omega = \frac{1}{N_\Omega} \sum_{i=1}^{N_\Omega} |\mathcal{N}_{tt} - c^2 \mathcal{N}_{xx} - s|^2, \quad (4.3.6a)$$

$$MSE_0 = \frac{1}{N_0} \sum_{i=1}^{N_0} |-\mu \mathcal{N}_x - F - g_0|^2, \quad (4.3.6b)$$

$$MSE_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} |Z \mathcal{N}_t + \mu \mathcal{N}_x - g_1|^2, \quad (4.3.6c)$$

$$MSE_{u_0} = \frac{1}{N_{u_0}} \sum_{i=1}^{N_{u_0}} |\mathcal{N} - u_0|^2, \quad (4.3.6d)$$

$$MSE_{v_0} = \frac{1}{N_{v_0}} \sum_{i=1}^{N_{v_0}} |\mathcal{N}_t - v_0|^2, \quad (4.3.6e)$$

where N_Ω , N_0 , N_1 , N_{u_0} , N_{v_0} correspond to the number of randomly distributed collocation points in the relevant subdomains, i.e. the domain interior, left boundary ($x = 0$), right boundary ($x = 1$), and at $t = 0$ (for initial displacements and velocities), respectively. In addition to (4.3.6) we have the contributions to the total loss from the state evolution equation, namely

$$MSE_\psi = \frac{1}{N_\psi} \sum_{i=1}^{N_\psi} |\mathcal{N}_t^\psi - G(V, \psi) - h|^2, \quad (4.3.7a)$$

$$MSE_{\psi_0} = \frac{1}{N_{\psi_0}} \sum_{i=1}^{N_{\psi_0}} |\mathcal{N}^\psi - \psi_0|^2, \quad (4.3.7b)$$

so that the objective function we seek to minimize is given by $MSE = \sum_{\xi \in \chi} MSE_\xi$, where subscript $\chi = \{\Omega, 0, 1, u_0, v_0, \psi, \psi_0\}$ so that the sum involves contributions from all subdomains. Minimizing the MSE over the network parameters of $\mathcal{N}$ and $\mathcal{N}^\psi$ define proxy models which produce approximations to u and ψ .

An important detail about the network training involved in minimizing the objective function MSE defined in the previous paragraph is that the inclusion of the state evolution means we are now solving a coupled system of PDE. The coupling occurs at the fault $x = 0$, where two conditions (fault friction (4.3.2d) and state evolution (4.3.5a)) are now enforced. Because of this coupling, the training networks $\mathcal{N}$ and $\mathcal{N}^\psi$ both appear in the fault loss (4.3.6b) as well as in the state evolution loss (4.3.7a). This interconnectivity may cause training to favor accuracy in the

displacement network over accuracy in the state network. In particular, high temporal variations in state evolution, variations in scale, and/or network architecture may drive the training step to favor the displacement approximation. We found it helpful to isolate the state and displacement networks during the backpropagation step. To do this, we define two objective functions: The state objective function consists of loss components in equation (4.3.7) while the displacement objective function uses loss components from equation (4.3.6). Each training iteration then requires two optimization steps to update both state and displacement networks. This setup ensures that each network is only updated by one associated objective function. We found this approach to improve our network's training speed and resulted in better approximations of the state variable. The particulars of this training approach are illustrated in Algorithm 1 which provides the pseudocode for solving the 1D IBVP given by equations (4.3.1) and (4.3.2) where state evolves according to (4.3.5a).

In order to verify our computational framework for the 1D problem, we generate a known, manufactured solution, which enables direct comparison with the approximate solution produced by the neural network. This approach, known as the method of manufactured solutions [147], assumes a particular analytic solution u^e and derives consistent source terms and boundary data for the IBVP, without changing the underlying PDE and boundary operators. We manufacture u^e and ψ^e to be a solution to the IBVP and aging law, respectively, such that

$$u^e(x, t) = \tanh(0.5(x - ct + 1)), \quad (4.3.8)$$

$$\psi^e(t) = -\frac{\mu u_x^e(0, t)}{\bar{\sigma}_n} - a \ln\left(\frac{2u_t^e(0, t)}{V_0}\right) \quad (4.3.9)$$

This choice of manufactured solution defines the initial data u_0 and v_0 , the source term s , and all boundary data, namely

$$s = u_{tt}^e - c^2 u_{xx}^e, \quad \text{on } \Omega \times [0, T], \quad (4.3.10a)$$

$$g_0 = 0, \quad \text{at } x = 0, \quad (4.3.10b)$$

$$g_1 = Zu_t^e + \mu u_x^e, \quad \text{at } x = 1, \quad (4.3.10c)$$

$$h = \psi_t^e - G(2u_t^e, \psi^e), \quad \text{at } x = 0, \quad (4.3.10d)$$

where, for the 1D problem, $V(t) = 2u_t(0, t)$, and corresponds to a commonly chosen exact solution [115], [148].

Parameter values used for all studies in this work are given in Table 2. We provide Figure 15 to illustrate how the independent networks $\mathcal{N}$ and $\mathcal{N}^\psi$ are used to build each component loss function

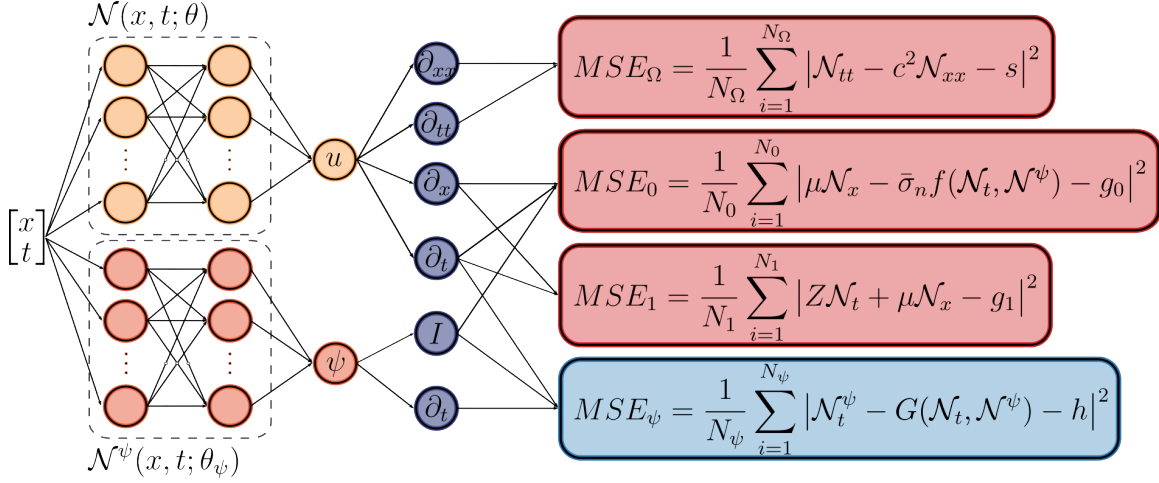


Fig. 15 A schematic of the PINN framework for solving the 1D IBVP (4.3.1) (4.3.2) with rate and state friction defined by (4.3.4) and (4.3.5a). Displacement network $\mathcal{N}$ and state network $\mathcal{N}^\psi$ are trained separately by defining two objective functions which are differentiated by red and blue nodes, respectively, in the final layer. At each training iteration, $\mathcal{N}$ is updated using component losses MSE_Ω, MSE_0 , and MSE_1 while $\mathcal{N}^\psi$ is updated using just MSE_ψ as an objective function.

and Algorithm 1 provides an outline of the implemented code. Here we take $\mathcal{N} : \mathbb{R}^2 \rightarrow \mathbb{R}$ and state variable network $\mathcal{N}^\psi : \mathbb{R} \rightarrow \mathbb{R}$ to both be fully-connected, feed-forward networks with three hidden layers, 64 neurons per layer and use hyperbolic tangent activation functions. The loss function is optimized using L-BFGS[149], a quasi-Newton optimization algorithm. Minimization is done over 10 training iterations where each iteration is trained on a random sampling of $N = 100$ interior points and 50 boundary points ($N_b = 25$ points per boundary). Network weights are initialized using uniform Xavier initialization [150]. In Figure 16 we show learned approximations for displacement and state compared to their respective exact solutions. These results suggest that our neural network parameters can solve the 1D forward problem to reasonable accuracy. To further explore performance however, and since we are primarily interested in higher dimensional settings that enable the inversion of depth-dependent parameters, we now move to the 2D formulation.

4.3.2 2D Application in Seismic Faulting. The 2D problem is obtained by first considering a bounded spatial domain in $\mathbb{R}^3$ defined by $(x, y, z) \in [-L_x, L_x] \times [-L_y, L_y] \times [0, L_z]$ where z is taken to be positive downward. We assume antiplane shear deformation, taking displacements u_x, u_z to be zero, and assume the out-of-plane displacement $u = u_y$ is independent of y . Momentum balance for an elastic solid then gives rise to the elastodynamic wave equation in 2D, namely,

$$u_{tt} = c^2 \Delta u + S(x, z, t), \quad (x, z) \in [-L_x, L_x] \times [0, L_z], \quad t \geq 0, \quad (4.3.11)$$

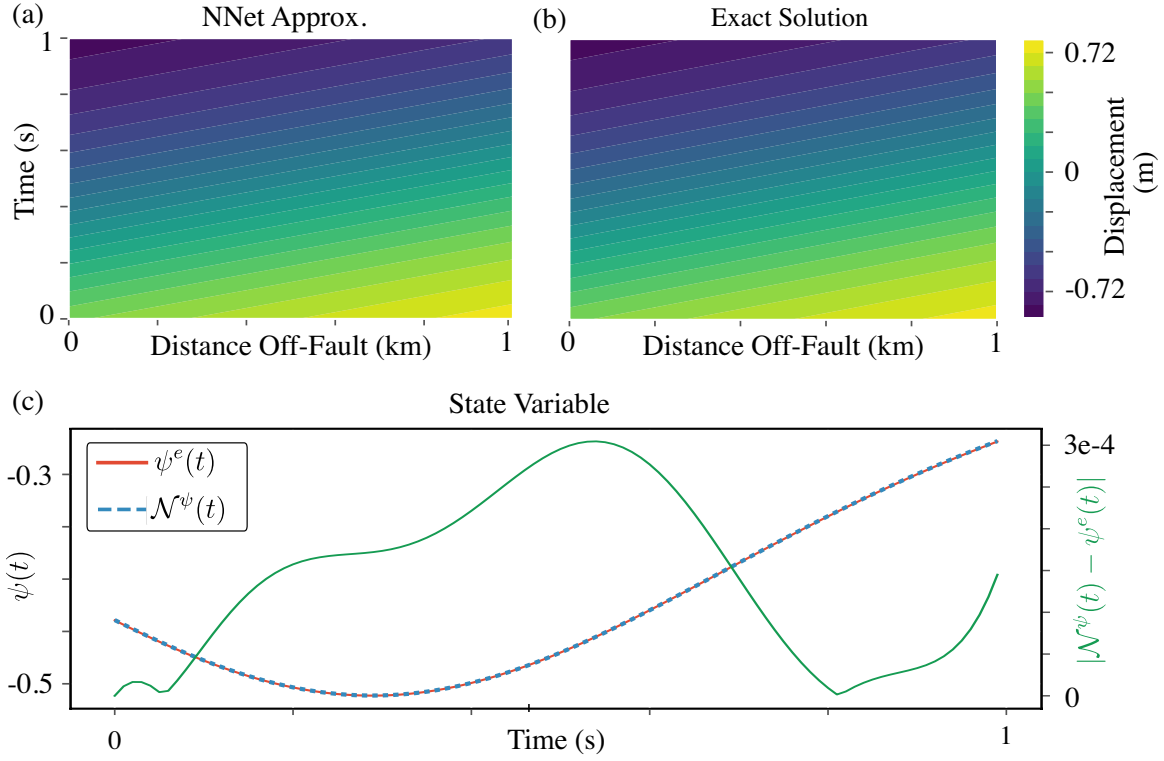


Fig. 16 Comparison of results from 1D illustration showing the (a) displacement network approximation $\mathcal{N}$ with (b) manufactured solution u^e . Additionally, the (c) state approximation network $\mathcal{N}^\psi$ is plotted against the manufactured state ψ^e along with their absolute error $|\mathcal{N}^\psi(t) - \psi^e(t)|$. Absolute displacement error was averaged over 1000 randomly sampled points and measured to be $|\mathcal{N} - u^e|_{\text{avg}} = 1.57e - 5$.

Algorithm 1 Multi-network training for 1D forward problem with RSF.

```

1:  $\mathcal{N} \leftarrow \text{NEURALNETWORK}(x, t; \theta)$  ▷ initialize network parameters  $\theta$  for  $\mathcal{N}$ 
2:  $\mathcal{N}^\psi \leftarrow \text{NEURALNETWORK}(t; \theta_\psi)$  ▷ initialize network parameters  $\theta_\psi$  for  $\mathcal{N}^\psi$ 
3:  $\mathcal{N}^{\text{opt}} \leftarrow \text{SETOPTIMIZER}(\theta)$  ▷ Instantiate an optimizer which only tracks parameters of  $\mathcal{N}$ 
4:  $\mathcal{N}^{\psi \text{opt}} \leftarrow \text{SETOPTIMIZER}(\theta_\psi)$  ▷ Instantiate an optimizer which only tracks parameters of  $\mathcal{N}^\psi$ 
5:
6:
7:  $\tilde{\mathcal{N}} \leftarrow u_0(x) + tv_0(x) + t^2\mathcal{N}(x, t)$  ▷ Employ hard enforcement of initial conditions
8:  $\tilde{\mathcal{N}}^\psi \leftarrow \psi_0 + t\mathcal{N}^\psi(t)$ 
9:
10: for  $\xi$  in  $\chi$  do ▷ Loop over each subdomain and define the relevant component loss
11:   function  $\text{MSE}_\xi(x, t)$ 
12:      $\text{output} \leftarrow \text{CONDITION}[\xi](\tilde{\mathcal{N}}(x, t), \tilde{\mathcal{N}}^\psi(t))$  ▷ Conditions specified by equations (4.3.1), (4.3.2), (4.3.3), and (4.3.5a)
13:      $\text{data} \leftarrow \text{SOURCEDATA}[\xi](x, t)$  ▷ Source terms are determined in equation (4.3.10)
14:     return  $\text{MEANSQUAREERROR}(\text{output}, \text{data})$ 
15:
16: for  $i=1$  to  $\text{training\_iterations}$  do
17:    $\text{Loss}_{\mathcal{N}} \leftarrow 0$ 
18:    $\text{Loss}_{\mathcal{N}^\psi} \leftarrow 0$ 
19:
20:   for  $\xi$  in  $\chi \setminus \{\Psi\}$  do
21:      $x_\xi, t_\xi \leftarrow \text{RAND}(N_\xi, \xi)$  ▷ Generate  $N_\xi$  randomly sampled points from subdomain  $\xi$ 
22:      $\text{Loss}_{\mathcal{N}} \leftarrow \text{Loss}_{\mathcal{N}} + \text{MSE}_\xi(x_\xi, t_\xi)$ 
23:
24:    $x_\Psi, t_\Psi \leftarrow \text{RAND}(N_\Psi, \Psi)$ 
25:    $\text{Loss}_{\mathcal{N}^\psi} \leftarrow \text{MSE}_{f_\psi}(x_\Psi, t_\Psi)$ 
26:
27:    $\nabla_\psi \leftarrow \text{grad}(\text{Loss}_{\mathcal{N}^\psi}, [\theta_\psi])$  ▷ Compute network gradient
28:    $\theta_\psi \leftarrow \mathcal{N}^{\psi \text{opt}}.\text{STEP}(\nabla_\psi)$  ▷ Update network weights using the optimization algorithm
29:
30:    $\nabla_{\mathcal{N}} \leftarrow \text{grad}(\text{Loss}_{\mathcal{N}}, [\theta])$ 
31:    $\theta \leftarrow \mathcal{N}^{\text{opt}}.\text{STEP}(\nabla_{\mathcal{N}})$ 

```

where $\Delta = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2} \right)$ is the 2D Laplacian. Here the shear wavespeed $c = \sqrt{\mu/\rho}$ as in the 1D case, and S comprises body forces. Assumed initial conditions are given by

$$u(x, z, 0) = u_0(x, z), \quad (4.3.12a)$$

$$u_t(x, z, 0) = v_0(x, z). \quad (4.3.12b)$$

We assume $z = 0$ corresponds to Earth's surface which we take to be traction free. An RSF vertical fault is embedded at the interface $x = 0$, corresponding to interface conditions

$$\tau^+ = -\tau^- \quad (4.3.13a)$$

$$\tau^+ = F(V, \psi). \quad (4.3.13b)$$

Here the traction τ at a boundary or interface is defined with respect to the outward pointing unit normal $\mathbf{n}$ by $\tau = \mathbf{n} \cdot \mu \nabla u$, and $\tau^\pm = \tau(0^\pm, z, t)$. Equation (4.3.13a) corresponds to the requirement that the traction be “equal and opposite” across the fault and (4.3.13b) is a requirement that fault shear stress be equal to frictional strength F as in the 1D case. As in the 1D illustration, F is a nonlinear function of the slip rate $V(z, t) = \dot{u}(0^+, z, t) - \dot{u}(0^-, z, t)$ and an empirical state variable $\psi(z, t)$ the particular form of which is given by (4.3.3). At the remote boundaries $(x, z) =$

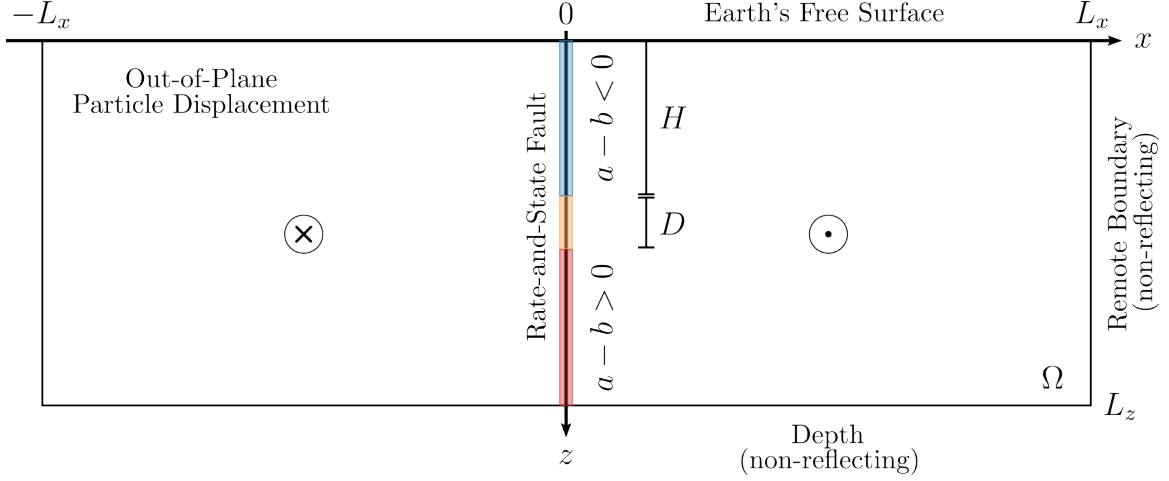


Fig. 17 A schematic of the 2D domain where $u(x, z, t)$ is the particle displacement in the y -direction. Out-of-plane motion is denoted with circles. The assumption that u is antisymmetric about $x = 0$ allows us to consider the one-sided domain $\Omega = [0, L_x] \times [0, L_z]$ to alleviate computational cost. Displacements within Ω are governed by the 2D wave equation (4.3.11). The fault, Earth's free surface, and the remaining boundaries (the remote and depth boundaries) are labeled and subject to conditions (4.3.14). At the rate-and-state fault ($x = 0$), the fault is velocity weakening ($a - b < 0$) down to a seismogenic depth H , where it transitions over distance D to velocity strengthening $a - b > 0$.

($\pm L_x, L_z$) we assume the boundaries are non-reflecting. Next we make the assumption that the displacement field $u(x, z, t)$ is antisymmetric about the fault interface $x = 0$, which automatically satisfies (4.3.13a), so that we may restrict our focus to one side of the fault as in Erickson and Dunham [115] and reduce the computational redundancy. Our final spatial domain is then $\Omega = [0, L_x] \times [0, L_z]$ where the fault interface now becomes a boundary, see Figure 17. For generality (and to aid in later verification steps) we state the final boundary conditions on $\partial\Omega$ using generic data g_f, g_s, g_r, g_d for the fault, surface, remote, and depth boundaries, respectively, namely

$$\tau = F(V) + g_f(z, t), \quad x = 0, \quad (4.3.14a)$$

$$\tau = g_s(x, t), \quad z = 0, \quad (4.3.14b)$$

$$Zu_t + \tau = g_r(z, t), \quad x = L_x, \quad (4.3.14c)$$

$$Zu_t + \tau = g_d(z, t), \quad z = L_z, \quad (4.3.14d)$$

although the Earth's free surface and non-reflecting conditions are the far field boundaries correspond to $g_s = g_r = g_d = 0$. Here $\tau = \tau^+$, $V = 2u_t^+$ and $Z = \sqrt{\mu\rho}$ is the shear impedance as in the 1D case.

For this 2D application problem we are primarily interested in the RSF parameter $a - b$, which describes the velocity-dependence of friction at steady state. Positive values (i.e. $a > b$)

correspond to stable sliding, while negative values correspond to frictional instabilities. Knowledge of the depth-distribution of $a - b$ is therefore of central importance to understanding slip behavior and deformation [125]. In other words, heterogeneities along fault interfaces can be characterized at least in part by velocity-weakening frictional behavior ($a - b < 0$) indicating that seismic rupture may nucleate and easily propagate, while stable regions are characterized by velocity-strengthening frictional behavior ($a - b > 0$) that inhibits the sliding at seismogenic speeds [141]. In order to focus our study on the inference of the depth-dependency of $a - b$, we assume that the state variable is at steady state, namely $\psi = f_0 + b \ln \left(\frac{V_0}{V} \right)$, so that the friction coefficient reduces to the rate-dependent form

$$f(V) = f_0 + \alpha \ln \left(\frac{V}{V_0} \right), \quad (4.3.15)$$

where we have introduced $\alpha = a - b$. This choice of rate-dependent friction is made to simplify the optimization problem by limiting the number of trainable networks from three to two for computational ease. We assume that α is piecewise linear with depth, defining a shallow seismogenic zone, namely,

$$\alpha(z) = \begin{cases} \alpha_{\min} & 0 < z < H \\ (z - H) * ((\alpha_{\max} - \alpha_{\min})/D) + \alpha_{\min} & H \leq z \leq H + D \\ \alpha_{\max} & H + D < z, \end{cases} \quad (4.3.16)$$

where H defines the seismogenic depth, $\alpha_{\min}$ and $\alpha_{\max}$ are constants defining the minimum and maximum values assumed by α , and D the transition distance. In this work we consider PINN solutions to both the forward and inverse problems. For forward problem we solve for the unknown displacements u in the IBVP (4.2.3) assuming $\alpha(z)$ has been specified. For the inverse problems, in addition to solving for u we also infer the friction parameter α . In this latter case, (4.3.16) sets the data for the boundary conditions, but α is not assumed to be known a priori when enforcing the frictional interface condition, as will be described in the next section.

4.3.3 Forward and Inverse Problems for the 2D Application. Governing equation (4.3.11) along with initial and boundary conditions (4.3.12), (4.3.14) provide specifics of the loss terms (4.2.5) that define the PINN. However, there is more than one way to formulate the associated learning problem. First, one can consider either a forward or inverse problem [3]. As this work is concerned with generating approximations to both forward and inverse problems, we will refer to the

primal solution as the network approximation $\mathcal{N}$ to the displacement u specified by the IBVP (4.2.3). The *forward* problem is only concerned with approximating a primal solution and requires that all model parameters λ are known a priori. A solution to the *inverse* problem aims to approximate the primal solution while also being tasked with learning a set of system parameters. If the λ in (4.2.3) are known, the forward problem is solved and the problem reduces to an unsupervised learning task [16]. In the inverse problem, however, the PINN has the same loss function (4.2.5) but with minor changes: Instead of knowing the system parameters λ , we establish them as trainable networks, which we detail shortly.

In addition to specifying whether a forward or inverse problem is being solved, in either case one must also choose between soft or hard enforcement of initial conditions [12], [13], [16], [20]. Soft enforcement uses loss terms to learn boundary data whereas hard enforcement encodes boundary data into a trial function to satisfy the conditions exactly. We proceed with details concerning each of these below.

4.3.3.1 Forward Problem. As in the 1D case, we begin by supposing that $\mathcal{N}(x, z, t; \theta) \approx u(x, z, t)$ for neural network $\mathcal{N}$ and solution u of the IBVP (4.3.11), (4.3.12), (4.3.14). The 2D component loss functions are therefore given by

$$MSE_{\Omega} = \frac{1}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} |\mathcal{N}_{tt} - c^2 \Delta \mathcal{N} - S|^2, \quad (4.3.17a)$$

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |-\mu \mathcal{N}_x - \bar{\sigma}_n f - g_f|^2, \quad (4.3.17b)$$

$$MSE_s = \frac{1}{N_s} \sum_{i=1}^{N_s} |-\mu \mathcal{N}_z - g_s|^2, \quad (4.3.17c)$$

$$MSE_r = \frac{1}{N_r} \sum_{i=1}^{N_r} |Z \mathcal{N}_t + \mu \mathcal{N}_x - g_r|^2, \quad (4.3.17d)$$

$$MSE_d = \frac{1}{N_d} \sum_{i=1}^{N_d} |Z \mathcal{N}_t + \mu \mathcal{N}_z - g_d|^2 \quad (4.3.17e)$$

along with initial loss terms MSE_{u_0}, MSE_{v_0} (translated to 2D) from equations (4.3.6d) and (4.3.6e), respectively. As in the 1D case, summations are over randomly distributed collocation points in the interior and domain boundaries. The 2D objective function we seek to minimize is given by $MSE = \sum_{\xi \in \chi} MSE_{\xi}$, where (as in the 1D case), subscript $\chi = \{\Omega, f, s, r, d, u_0, v_0\}$ implies that the sum is taken over all relevant subdomains.

4.3.3.2 Inverse Problem. While the primal solution to an IBVP is desirable, replacing an assumed model parameter (which often comes with much uncertainty) with a trainable network could leverage real-world data to physically constrain parameter distributions. The inverse problem is derived by making a slight modification to the forward problem (4.3.17). As we are interested in learning the fault friction parameter α , we modify the loss component (4.3.17b) to include a trainable network. Because α is depth-variable it must be explicitly trained on random collocation points along the fault. So in addition to the first network $\mathcal{N}(x, z, t; \theta) \approx u(x, z, t)$, we introduce a secondary network $\mathcal{N}^\alpha(z; \theta_\alpha) \approx \alpha(z)$. Only one loss component needs be changed from the forward problem (4.3.17), namely, component (4.3.17b), which enforces the fault friction condition. A modified friction coefficient is considered, defined by

$$\tilde{f} = f_0 + \mathcal{N}^\alpha(z) \ln \left[\frac{2\mathcal{N}_t(z, t)}{V_0} \right], \quad (4.3.18)$$

which gives rise to the modified fault loss

$$\overline{MSE}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} | -\mu \mathcal{N}_x - \bar{\sigma}_n \tilde{f} - g_f |^2. \quad (4.3.19)$$

Replacing loss component (4.3.17b) in the forward problem with (4.3.19) defines the inverse problem. We give a diagram of the fault network for the inverse problem in Figure 18 showing how the fault loss is used to update two networks during training. The inverse problem is trained over the same subdomains as the forward problem (4.3.17) but the depth coordinate for fault training data is passed to both $\mathcal{N}$ and $\mathcal{N}^\alpha$.

4.3.3.3 Soft vs. Hard Enforcement of Boundary Conditions. Both the forward and inverse problems require that we specify initial and boundary conditions, which may be enforced in two possible ways. *Soft* enforcement is done by penalizing the objective function, as the network output need not satisfy the condition exactly (as was done in the 1D example of the previous section). Up until now, we have presented soft enforcement, as given in (4.3.17), which shows component losses corresponding to the boundary and initial conditions. Soft enforcement does not give us any guarantee regarding accuracy of the condition being enforced and each additional loss term in the objective function increases the complexity of the optimization landscape.

Alternatively, we may enforce initial/boundary conditions in a manner that reduces the complexity of the network training space by encoding such conditions into the network architecture. This approach, known as *hard* enforcement, requires that we specify a trial function which

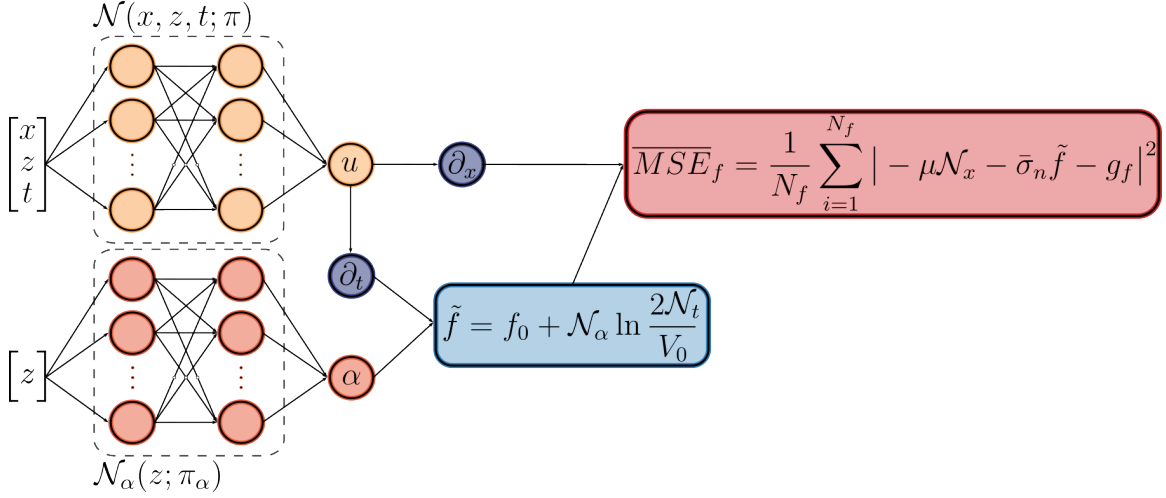


Fig. 18 Network diagram for enforcing the stress condition along the fault while also learning the depth-dependent friction parameter α . During training, the z -coordinate of a training point is passed to both $\mathcal{N}$ and $\mathcal{N}^\alpha$.

automatically satisfies the initial/boundary conditions. Here we apply this technique to enforce initial conditions, which reduces the number of constraints on the system. Let $\mathcal{N}(\mathbf{x}, t; \theta)$ be a feed-forward neural network and define the trial function $\tilde{\mathcal{N}}$ to be

$$\tilde{\mathcal{N}}(\mathbf{x}, t; \theta) = u_0(\mathbf{x}) + tv_0(\mathbf{x}) + t^2\mathcal{N}(\mathbf{x}, t; \theta) \quad (4.3.20)$$

where $\mathbf{x} = (x, z)$, so that $\tilde{\mathcal{N}}$ satisfies both initial conditions (4.3.12) exactly and is trained by training the network $\mathcal{N}$. It is worth noting that $\tilde{\mathcal{N}}$ represents an approximation to displacement and consists of a static component (first two terms) that enforce the initial conditions exactly, and a trainable component $\mathcal{N}$. The final term in equation (4.3.20) is quadratic in time to ensure that the trainable component of $\tilde{\mathcal{N}}$ (and $\tilde{\mathcal{N}}_t$) vanishes at $t = t_0$. If $\tilde{\mathcal{N}}$ is used in place of $\mathcal{N}$ in the formulation of the objective function, there is no longer a need to include loss terms for initial displacements (4.3.6d), or initial velocities (4.3.6e) so the network can be trained on a less restrictive set of conditions.

4.4 2D Verification, Validation and Applications

When computational methods for physical problems are used to address science questions, verification is an essential first step to ensure credible results [98], [143]. While validation with observational data is the focus of future work, we must first verify that our physics-informed deep learning framework is able to solve both forward and inverse problems to reasonable accuracy.

4.4.1 Verification with the Method of Manufactured Solutions. As in the 1D case, we verify the PINN framework by generating a known, manufactured solution u^e which solves

Parameter	Value
L_x, L_z	25 km
H	12 km
D	5 km
μ	32 GPa
ρ	2.67 kg/m ³
$\alpha_{\min}$	-0.005
$\alpha_{\max}$	0.015
f_0	0.6
$\bar{\sigma}_n$	50 MPa
V_0	10 ⁻⁶ m/s
D_c	2 m

Table 2. Parameter values used in the manufactured solution tests.

the IBVP (4.2.3), (4.3.12), (4.3.14). We take

$$u^e(x, z, t) = \tanh((x + z + ct)/20), \quad (4.4.1)$$

which defines the initial data u_0 and v_0 , the source term S , and all boundary data, which for the forward problem is

$$S = u_{tt}^e - c^2 \Delta u^e, \quad \text{on } \Omega \times [0, T], \quad (4.4.2a)$$

$$g_f = -\mu u_x^e - \bar{\sigma}_n f(2u_t^e), \quad \text{at } x = 0, \quad (4.4.2b)$$

$$g_s = -\mu u_z^e, \quad \text{at } z = 0, \quad (4.4.2c)$$

$$g_r = Zu_t^e + \mu u_x^e, \quad \text{at } x = L_x, \quad (4.4.2d)$$

$$g_d = Zu_t^e + \mu u_z^e, \quad \text{at } z = L_z. \quad (4.4.2e)$$

The parameters used are given in Table 2 whose values align with other recent works with similar model set-ups [148], [151], and we default to this parameter set unless stated otherwise. Note that in the case of the inverse problem, $\alpha_{\min}$ and $\alpha_{\max}$ define the manufactured exact solution $\alpha^e(z)$, which is used only to set the data in (4.4.2b); $\alpha(z)$ is a learned parameter through the use of the network $\mathcal{N}^\alpha$. In all scenarios we consider, we take the primal network $\mathcal{N} : \mathbb{R}^3 \rightarrow \mathbb{R}$ and friction network $\mathcal{N}^\alpha : \mathbb{R} \rightarrow \mathbb{R}$ to both be fully-connected, feed-forward networks with three hidden layers and 128 neurons per layer. The wave equation (4.3.11) requires displacements to be sufficiently smooth with respect to the input but no such requirement is imposed on the friction parameters. Thus we use hyperbolic tangent activation functions for the displacement network but use linear units in the friction network. We found that the friction network performed well when we used both Rectified Linear Unit (ReLU) and Sigmoid Linear Unit (SiLU) together. We use ReLU activation

Algorithm 2 Multi-network training for solving the inverse problem

```

1:  $\mathcal{N} \leftarrow \text{NEURALNETWORK}(x, z, t; \theta)$  ▷  $\theta$  initial network parameters for  $\mathcal{N}$ 
2:  $\mathcal{N}^\alpha \leftarrow \text{NEURALNETWORK}(z; \theta_\alpha)$  ▷  $\theta_\alpha$  initial network parameters for  $\mathcal{N}^\alpha$ 
3:  $\theta \leftarrow [\theta, \theta_\alpha]$ 
4:
5:  $\tilde{\mathcal{N}} \leftarrow u_0(x, z) + tv_0(x, z) + t^2\mathcal{N}(x, z, t)$  ▷ Employ hard enforcement of initial conditions
6:
7: for  $\xi$  in  $\chi$  do ▷ Loop over each subdomain and define the relevant component loss
8:   function  $\text{MSE}_\xi(\mathbf{x}, \mathbf{z}, \mathbf{t})$ 
9:      $\text{output} \leftarrow \text{CONDITION}[\xi](\tilde{\mathcal{N}}(x, z, t), \mathcal{N}^\alpha(z))$ 
10:     $\text{data} \leftarrow \text{SOURCEDATA}[\xi](x, z, t)$ 
11:    return  $\text{MEANSQUAREERROR}(\text{output}, \text{data})$ 
12:
13: for  $i=1$  to  $\text{training\_iterations}$  do
14:    $\text{Loss} \leftarrow 0$ 
15:
16:   for  $\xi$  in  $\chi$  do
17:     $\mathbf{x}_\xi \leftarrow \text{RAND}(N_\xi, \xi)$  ▷ Generate  $N_\xi$  randomly sampled points from subdomain  $\xi$ 
18:     $\text{Loss} \leftarrow \text{Loss} + \text{MSE}_\xi(\mathbf{x}_\xi)$ 
19:
20:    $\nabla \leftarrow \text{grad}(\text{Loss}, [\theta])$ 
21:    $\theta \leftarrow \text{OPTIMIZERSTEP}(\nabla)$  ▷ Update network weights using the optimization algorithm

```

for the outer hidden layers and a SiLU activation on the interior layer. Algorithm 2 is a sketch of the implemented code used to solve the inverse problem with hard enforcement of initial conditions.

$\mathcal{N}$ and (in the case of the inverse problem) $\mathcal{N}^\alpha$ are trained by minimizing the MSE (the sum of component MSE given in (4.3.17)) where all loss functions are optimized using L-BFGS [149], a quasi-Newton optimization algorithm. Minimization is done over 30 training iterations where each iteration is trained on a mini-batch of $N = 400$ interior points and 400 boundary points ($N_b = 100$ points per boundary). Additionally, if soft enforcement of initial conditions is used, we sample 400 additional interior points (200 for displacements and 200 for velocities). After the 30 training iterations are complete we evaluate each of the component loss functions using 1000 randomly sampled points. Likewise we generate the same number of sample points for $\mathcal{N}$ and $\mathcal{N}^\alpha$ and use their respective manufactured solutions to compute the relative ℓ^2 -norm of their errors, namely,

$$\|\mathcal{N} - u^e\|_{1, \text{rel}}^2 = \frac{\sum_{i=1}^N |\mathcal{N}(x_i, z_i, t_i) - u^e(x_i, z_i, t_i)|^2}{\sum_{i=1}^N |u^e(x_i, z_i, t_i)|^2}, \quad (4.4.3)$$

$$\|\mathcal{N}_\alpha - \alpha^e\|_{1, \text{rel}}^2 = \frac{\sum_{i=1}^{N_b} |\mathcal{N}^\alpha(z_i) - \alpha^e(z_i)|^2}{\sum_{i=1}^{N_b} |\alpha^e(z_i)|^2}. \quad (4.4.4)$$

All collocation points are drawn uniformly from the space-time domain and are resampled at each training iteration. This choice of sampling method, in addition to being simple to implement, has been shown to perform well compared to fixed residual point methods (e.g. uniformly-spaced grids, uniform random points, or Latin hypercube sampling for which collocation points are not resampled)[152].

Errors	Forward Problem			Inverse Problem		
	Soft	Hard	Error Ratio	Soft	Hard	Error Ratio
$\ \mathcal{N} - u^e\ _{1, \text{rel}}$	1.364e-02	7.148e-03	1.908e+00	2.931e-02	1.633e-02	1.795e+00
$\ \mathcal{N}_\alpha - \alpha^e\ _{1, \text{rel}}$				2.977e-01	7.220e-02	4.123e+00
MSE_{PDE}	1.149e-03	8.883e-03	1.293e-01	1.133e-03	6.675e-03	1.697e-01
MSE_f	2.633e-03	4.115e-02	6.399e-02	8.579e-03	1.615e-01	5.311e-02
MSE_s	8.393e-05	7.868e-04	1.067e-01	1.251e-04	6.545e-04	1.911e-01
MSE_d	8.216e-05	2.115e-03	3.885e-02	4.681e-04	7.199e-03	6.502e-02
MSE_r	1.195e-05	3.233e-04	3.698e-02	1.064e-04	8.074e-04	1.318e-01
MSE_{u_0}	1.703e-04			3.722e-04		
MSE_{v_0}	6.308e-04			1.375e-03		

Table 3. Errors in the ℓ^2 -norm and MSE (loss) associated with hard and soft enforcement of initial conditions for both the forward and inverse problem. Each term is computed over 1000 randomly sampled points and averaged across 50 trained solutions. Error ratios are computed by dividing soft enforcement error by the hard enforcement error.

Network weights are initialized using uniform Xavier initialization [150]. Variation due to weight initialization involving randomization is accounted for by averaging error data across 50 trained solutions. Table 3 presents error data for the forward and inverse problems where both hard and soft enforcement of initial conditions are considered for each. As the table illustrates, for both the forward and inverse problems, hard enforcement of initial conditions are accompanied with a smaller ℓ^2 -errors after 30 training iterations, which is most likely due at least in part to the fact that hard enforcement means that the network approximation satisfies the initial conditions exactly. Also illustrated in the table, is that all component mean-square errors (MSE) are lower for the soft enforcement of initial conditions compared to hard enforcement. Soft enforcement uses two additional loss components (i.e. those enforcing initial conditions) which results in a more biased model, or a lower variance model (i.e. the MSE is less sensitive to training data). The trade off between bias and variance (introduced via randomness in weight initialization and collocation points) may account for such lower MSE when using soft enforcement (which involves lower variances) and averaging over 50 trained models.

Next we test network performance across various subdomains and resolutions. Let $\mathcal{N}$ be a displacement network trained to solve the inverse problem using hard enforcement of initial conditions. This time, errors are measured with respect to the continuous L^2 -norm

$$\|\mathcal{N} - u^e\|_{2,\Lambda}^2 = \int_{\Lambda} |\mathcal{N}(x, z, t) - u^e(x, z, t)|^2 d\lambda, \quad (4.4.5)$$

and we consider error accumulation across Ω , $[0, T]$ and $\hat{\Omega} = \Omega \times [0, T]$, the spatial, temporal and space-time domains, respectively. The integral in (4.4.5) is approximated using Simpson's composite quadrature rule which converges to the exact integral with rate $\mathcal{O}(k^4)$, where k is the quadrature grid size (i.e. the subinterval length). To account for variations in the trained network, we average errors over points outside of the domain of integration. For example, the average temporal error given by

$$\overline{\|\mathcal{N} - u^e\|_{2,[0,T]}} = \frac{1}{N_T} \sum_{k=1}^{N_t} \left[\int_{[0,T]} |\mathcal{N}(x_k, z_k, t) - u^e(x_k, z_k, t)|^2 d\lambda(t) \right]^{1/2} \quad (4.4.6)$$

measures expected error accumulation (over time) given N_T randomly sampled spatial points.

Figure 19 shows displacement plots for the trained network and the exact solution at final time $T = 1$ in addition to network L^2 -error approximations for displacement in space, time and space-time (along with an L^2 -error approximation of the friction parameter) for increasingly higher grid resolution used in the numerical quadrature. The errors remain relatively constant with decreasing subinterval size, suggesting that the quadrature approximation is approaching the actual L^2 -error, and that the PINN approximation maintains good accuracy even when evaluated on higher-resolution grids. Figure 20(a) shows network component loss functions against training iteration, revealing a non-monotonic decrease across all components. This behavior is likely due to how data is batched during training with smaller batches tending towards smaller, less accurate loss updates while larger batches tend towards fewer, more accurate loss updates [153], [154]. Figure 20(b) illustrates convergence of the inferred parameter $\alpha = a - b$ to the exact distribution α^e . Here we show iterations 1 and 30 merely because after the first iteration the inferred parameter performs well and does not vary significantly.

4.5 Summary and Future Work

We have presented a computational framework for physics-informed neural networks (PINNs) for solving the elastodynamic wave equation with a rate-and-state frictional fault boundary in both 1D and 2D. We consider both forward and inverse problems, with the latter obtained by extending to a multi-network architecture in order to learn depth-dependent friction parameters alongside deformations in the Earth's crust. We verified the computational framework by applying the method of manufactured solutions to probe various error measurements. We show that in general, hard enforcement of boundary conditions result in trained networks that better approximate displacements and friction parameters but tend to be worse at minimizing component loss functions

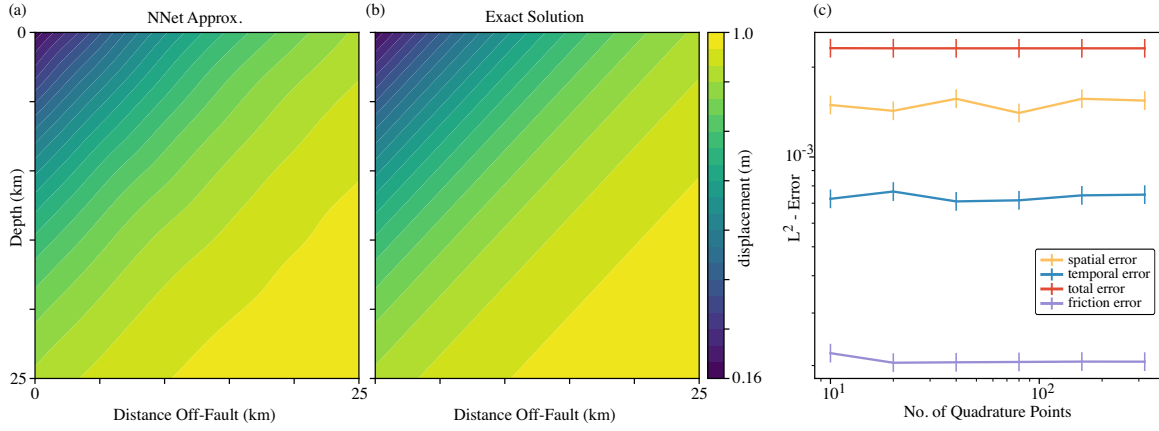


Fig. 19 (a) 2D displacement plot for a PINN trained to solve the inverse problem using hard enforcement of initial conditions compared to (b) the manufactured displacements. (c) L^2 -errors for displacement in space, time, and spacetime (along with L^2 -errors for the friction parameter) are computed on a uniform grid using Simpson's rule as a quadrature. Errors are then recorded over several mesh refinements.

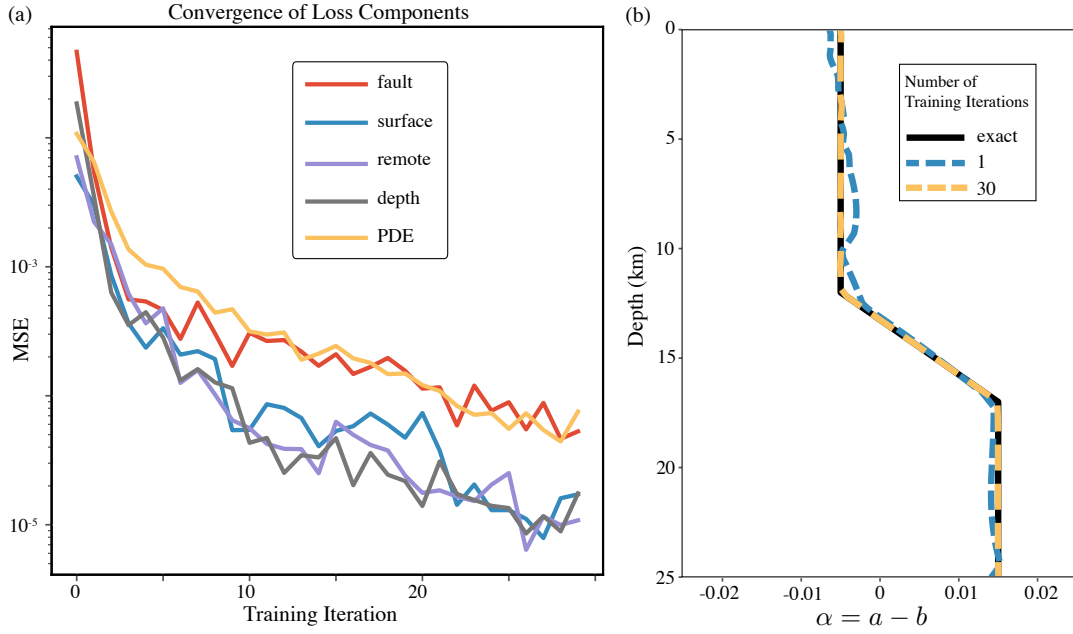


Fig. 20 2D inversion results showing (a) convergence of loss components and (b) convergence of the inferred parameter approximation.

when compared to soft enforcement. We found that a PINN defined by hard enforcement of initial conditions produces reasonable approximations for displacements and the desired friction parameter distribution. Though the network is mesh-free, the L^2 -error yields near constant values across various quadrature grids, suggesting that the PINN provides a reasonable approximation to the solution even when evaluated on increasingly finer grids. Moreover, although the network requires sufficient training iterations to properly learn displacements, the desired friction parameter is learned within the first couple of iterations. This suggests that PINNs may be a highly effective tool in inferring subsurface friction properties along faults, constrained by both physics and observational surface data.

While the PINN is shown to perform well when learning the state variable in 1D, and inferring depth-dependency of RSF parameter $a - b$ at steady-state in 2D, we plan to explore the capabilities of the 2D framework with non-steady-state-evolution. This extension requires additional networks in the 2D setting in order to approximate the state variable, and two inference networks to capture the empirical parameters a and b (which become separated across governing equations), and/or networks to learn other frictional parameters, such as D_c , whose scaling from laboratory values to actual fault zones is the subject of many studies [e.g. 155]. Approaching such a problem may be aided by a better understanding of the PINN dependence on problem configuration as well as network architecture. Additionally, it would be worthwhile to investigate methods which hybridize PINNs with traditional numerical methods similar to the discrete time PINN in Raissi, Perdikaris and Karniadakis [3]. And finally, with a PINN solution that can handle learning full rate-and-state fault friction in 2D (and eventually 3D), we would be ready to compare model outcomes against community benchmark problems concerning dynamic rupture simulations [143] and sequences of earthquakes and aseismic slip [98].

4.6 Data Availability Statement

All code for building and training models is open-source, written in Python using the machine learning library Pytorch, and is available at https://github.com/Thrased/eq_pinns

CHAPTER V

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this dissertation I present the scientific contributions I’ve made to the field of geophysical modeling and inverse problems. I focus on techniques that help circumvent the high computational cost of solving inverse problems for complex systems. First, we present a novel framework for modeling subsurface magma reservoirs surrounded by a thermo-viscoelastic material which reveals a link between the reservoir pressurization timescale and the viscoelastic transition boundary within the domain. Additionally, we employ analytic signals to compute a transfer function relating reservoir stress to maximal surface uplift. This use of complex signal analysis allows us to deconstruct surface observations into their component signals and provides an inverse transfer function for inferring stress. Second, we design an ML numerical method which solves the antiplane strike-slip problem coupled to nonlinear rate-and-state friction. By including an additional network into the system we demonstrate that PINNs are able to infer for depth-dependent friction parameters without a significant increase to the computational cost of training. Moreover, the parameter inversion network was shown to converge within the first few training iterations, which was much quicker than the training time required for the solution network. While the work in this dissertation makes valuable contributions to the modeling community, we acknowledge that there is still a lot of work to do in developing the proposed techniques.

5.2 Future Work

5.2.1 Transfer Function. The transfer function we compute is explicitly dependent on the reservoir pressurization frequency but further study is necessary to understand how reservoir geometry, depth, and temperature influence the amplitude and phase shift of the transfer function. Additionally, it may be fruitful to implement the PINN inverse approach as a means to study parametric dependence of the model. Since the ML method results in a persistent solution that is defined continuously over the domain, we can compute network derivatives with respect to each parameter and study the output dependence to help narrow the field of parameters used during model inversion.

5.2.2 PINNs. Because PINNs are so new there is much work to be done to develop them as a reliable numerical method. With respect to the PINN presented in this dissertation, the

next step is to verify its effectiveness on real-world problems. To do this, we want to look at sliding block laboratory experiments and use their gathered displacement data to test the models inference capabilities. Moreover, I think the PINN should be expanded in two key ways: hard-enforcement of boundary conditions and optimization algorithm. Currently, inclusion of boundary data in the loss function increases the complexity of the training landscape and leads to boundary conditions only being met approximately. Hard-enforcement of boundary conditions will allow for exact enforcement of data along boundaries while reducing training complexity and disqualifying the chance that the network will learn the zero solution on the domain interior. Additionally, the inclusion of a natural energy approach to precondition the gradient during training using information from the PDE could help increase the order of accuracy that is achievable by the neural network. Lastly, we would like to see further development of the PINN approach to utilize Bayesian methods for uncertainty quantification as this would allow for us to measure a level of certainty when dealing with non-unique maps between parameters and outputs.

APPENDIX A

AREA EXAM APPENDIX

A.1 Gauss-Legendre Runge-Kutta Method

Let $t > 0$ and consider the ordinary differential equation

$$\frac{d}{dt}u(t) = f(t, u) \quad (\text{A.1.1})$$

where $u : \mathbb{R} \rightarrow \mathbb{R}^d$ and $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$ are vector-valued functions. The Gauss-Legendre quadrature approximates the definite integral of a function over the symmetric interval $[-1, 1]$ on n sample points

$$\int_{-1}^1 f(x) dx \approx \sum_{i=1}^n w_i f(x_i) \quad (\text{A.1.2})$$

where $\{w_i\}_{i=1}^n$ are the quadrature weights, x_i the roots of the n^{th} degree Legendre polynomial, $P_n(x)$, and $\{k_i\}_{i=1}^n$ give the sample velocities $k_i = f(x_i)$ for $1 \leq i \leq n$. For an ODE of the form Equation A.1.1 the Runge-Kutta method approximates a single time step $t_\eta \rightarrow t_{\eta+1}$ using a sequence of q stages by $u^{\eta+c_i} = u(t_\eta + \Delta t c_i)$, $i = \{1, 2, \dots, q\}$

$$u^{\eta+c_i} = u^\eta + \Delta t \sum_{j=1}^q A_{ij} f(t_\eta + \Delta t c_j, u^{\eta+c_j}) \quad (\text{A.1.3a})$$

$$u^{\eta+\Delta t} = u^\eta + \Delta t \sum_{i=1}^q b_i f(t_\eta + \Delta t c_i, u^{\eta+c_i}) \quad (\text{A.1.3b})$$

where $\mathbf{A} \in \mathbb{R}^{q \times q}$ is the Runge-Kutta coefficient matrix, $\mathbf{b} \in \mathbb{R}^q$ the stage combination coefficients and $\mathbf{c} \in \mathbb{R}^q$ the vector of abscissa obtained by shifting the Legendre polynomial roots from $[-1, 1] \rightarrow [t_\eta, t_{\eta+1}]$. This gives the general form of RK methods and selection of $\mathbf{A}$ and $\mathbf{b}$ will determine a precise numerical scheme. The method will typically be an implicit method except if $\mathbf{A}$ satisfies $A_{ij} = 0$ for $i \leq j$ in which case the result is an explicit method. The Gauss-Legendre Runge-Kutta method defines an implicit method where $\mathbf{A}, \mathbf{b}, \mathbf{c}$ are determined using Legendre polynomials. The choice of quadrature weights and nodes in Equation A.1.2 gives the GL quadrature the capacity to perform an exact integration of polynomials with degree $2n - 1$. While this approach offers a very nice approximation with truncation error $O(\Delta t^{2q})$ when q stages are used, it requires one to solve a system of algebraic equations at each time step which becomes costly. The exact point at which the

methods shows diminishing returns will vary between problems but it is common to use a 3-stage formulation [156].

To properly construct the q -stage GLRK method, we need the roots of $P_q(x)$ along with the derivative $P'_q(x)$. Though the roots can be found using a root finding method like the Netwon method, there has been a recent development that provides explicit asymptotic forms of the GL nodes and weights which are accurate up to double-precision machine ϵ for $q \geq 21$ [17], [18], [157]. If $\{x_i\}_{i=1}^q$ are the roots of the Legendre polynomial (with $P_q(x)$ normalized to so that $P_q(1) = 1$), then the Gauss-Legendre quadrature weights are given by

$$w_i = \frac{2}{(1 - x_i^2)[P'_q(x_i)]^2}. \quad (\text{A.1.4})$$

Earlier this year, it was shown that by considering a transformation of the integration domain from $[-1, 1] \rightarrow [t_n, t_{n+1}]$ by $\xi = \frac{\Delta t}{2}(t - \bar{t}_n)$ for step size $\Delta t \equiv t_{n+1} - t_n$ and mid-point $\bar{t}_n \equiv \frac{1}{2}(t_{n+1} + t_n)$ we may use the the node points $\{\xi_i\}_{i=0}^{q-1}$ (roots of the transformed Legendre polynomial) to compute

$$A_{ij} = \frac{w_j}{2} \left(1 + \sum_{s=0}^{q-1} P_s(\xi_j) \frac{P_{s+1}(\xi_i) - P_{s-1}(\xi_i)}{2} \right), \quad (\text{A.1.5a})$$

$$b_i \equiv \frac{w_i}{2}, \quad (\text{A.1.5b})$$

demonstrating how the Gauss-Legendre Runge-Kutta method can be constructed to arbitrarily high order without much difficulty [19]. In Equation A.1.5 we use $P_{-1}(\xi) \equiv 1$. With the RK parameters properly defined, all that is left is to solve the implicit method given by using said parameters in Equation A.1.3.

The choice of an implicit method means Equation A.1.3 will be solved using an iterative root-finding method like the Newton-Raphson method. To do this, it helps to create stacks of vectors and represent the problem as a matrix-vector problem. We then define the vector stacks

$$\mathcal{U}^\eta := \bigoplus_{i=1}^q u^\eta, \quad (\text{A.1.6a})$$

$$\mathcal{U}^{\eta+c} := \bigoplus_{i=1}^q u^{\eta+c_i}, \quad (\text{A.1.6b})$$

$$F(\mathcal{U}^{\eta+c}) := \bigoplus_{i=1}^q \Delta t f(t_\eta + c_i \Delta t, u^{\eta+c_i}), \quad (\text{A.1.6c})$$

allowing Equation A.1.3a to be represented as $\mathcal{D}(\mathcal{U}^{\eta+c}) = 0$ with

$$\mathcal{D}(\mathcal{U}^{\eta+c}) = \mathcal{U}^\eta - \mathcal{U}^{\eta+c} + \Delta t(A \otimes I)F(\mathcal{U}^{\eta+c}). \quad (\text{A.1.7})$$

Then, the k^{th} step of the Newton method applied to Equation A.1.3 involves the following matrix problem and iteration update

$$\mathcal{D}'(\mathcal{U}_k^{\eta+c}) \Delta \mathcal{U}_k^{\eta+c} = \mathcal{D}(\mathcal{U}_k^{\eta+c}), \quad (\text{A.1.8})$$

$$\mathcal{U}_{k+1}^{\eta+c} = \mathcal{U}_k^{\eta+c} + \Delta \mathcal{U}_k^{\eta+c}, \quad (\text{A.1.9})$$

where the derivative $\mathcal{D}'$ is given by

$$\mathcal{D}'(\mathcal{U}^{\eta+c}) = \Delta t(A \otimes I)F'(\mathcal{U}^{\eta+c}) - I, \quad (\text{A.1.10})$$

and the derivative of F is the block-diagonal matrix

$$F'(\mathcal{U}^{\eta+c}) := \bigoplus_{i=1}^q \mathcal{J}_f(u^{\eta+c_i}) \quad (\text{A.1.11})$$

consisting the the Jacobian of f evaluated at different stages. With $f : \mathbb{R}^{d+1} \rightarrow \mathbb{R}^d$ and the choice of q stages in the RK method, the resulting system of algebraic equations from Equation A.1.8 is $qd \times qd$. Thus, a solution (probably via LU factorization) will need $\mathcal{O}((qd)^3)$ operations. Luckily it is possible to re-use the LU factorization across the Newton iterations but the system will need a new factorization computed for each time step [158], [159].

A.2 Special Considerations

There are some network limitations that arise when approximating solutions for differential equations. A popular activation function choice becomes unviable in this problem domain. The Rectified Linear Unit (ReLU) can be used in bounded-width networks to create universal approximators [9], [11] (a fact that is generalized in [4] but requires a deeper network for arbitrary activation functions) and approximation error for such network can be bounded in terms of the network width and depth [160]. Now, considering a simple neural network $\mathcal{N}(x; W, b) = \phi.(Wx + b)$ we can see that its first two input derivatives are

$$\frac{\partial}{\partial x} \mathcal{N}(x; W, b) = \phi'.(Wx + b) * .W, \quad (\text{A.2.1a})$$

$$\frac{\partial^2}{\partial x^2} \mathcal{N}(x; W, b) = \phi''.(Wx + b) * .W * .W. \quad (\text{A.2.1b})$$

That is derivatives of the network require derivatives of the activation function. This generalizes to derivatives of order k requiring the order k derivatives for the activation function [12]. Because physics-based neural net methods utilize network derivatives in construction of the loss function, ReLU activation function is not generally usable. To approximate a PDE involving derivatives of order k , the activation function must be differentiable of order $k + 1$ to account for the additional derivative taken during backpropagation.

While both of these are susceptible to the vanishing gradient problem, tanh tends to be more stable due to a larger gradient range. While we want an activation function with sufficient regularity we would also like a network that can be trained efficiently. Rather than using a fixed activation function we can modify the network to learn activation functions for us. One approach to this is to learn coefficients that approximate an optimal activation function with respect to a chosen basis [161]. While such an approach does learn the activation function it does require you to choose a basis for the approximate activation. Instead we will focus on a method that works by learning slopes of the activation functions and which have been applied to PINN [162]–[164].

A.3 Adaptive Activation Functions

Begin by modify an activation function ϕ with trainable slope $a \in \mathbb{R}$ and scaling parameter $n \geq 1$ to get a new activation function,

$$\tilde{\phi}(x) = \phi(nax), \quad (\text{A.3.1})$$

where a is now included in the set of trainable parameters. This approach can improve the efficiency of DNN [162] along with improving the robustness, and accuracy of neural net approximations for PDE [163]. Furthermore, adaptive coefficients can be included at the global, layer, and neuron levels allowing for finer tuning of the network during training. Equation A.3.1 shows a globally-adaptive activation as it uses a single shared value of a for each layer in the network.

A.3.1 Layer-Wise Locally Adaptive Activation. You could also define a scalar for each hidden layer $\{a_i\}_{i=1}^{L-1}$ where $a_i \in \mathbb{R}$ for each i and specify activation function for layer k as

$$\tilde{\phi}_k(x) = \phi(n a_k x) \quad \text{for } k = \{1, 2, \dots, L - 1\}. \quad (\text{A.3.2})$$

Since the set of training parameters is now $\theta \cup \{a_i\}_{i=1}^{L-1}$ the number of parameters needing optimization is $|\theta| + L - 1$.

A.3.2 Neuron-Wise Locally Adaptive Activation. This time each neuron is given a trainable weight. So let $\{a_i\}_{i=1}^{L-1}$ where $a_i \in \mathbb{R}^m$ and specify activation function for layer k by

$$\tilde{\phi}_k(x) = \phi(n a_k * . x) \quad \text{for } k = \{1, 2, \dots, L-1\}. \quad (\text{A.3.3})$$

Since each layer comes with m hidden neurons we get $|\theta| + m(L-1)$ trainable network parameters.

While these adaptive activation functions do improve training efficiency, [164] includes an extra term in the loss function which can contribute to the loss gradient without vanishing. The slope recovery term $\mathcal{S}$ is defined as

$$\mathcal{S}(a) := \begin{cases} \frac{L-1}{\sum_{i=1}^{L-1} \exp(a_i)} & \text{if } a_i \in \mathbb{R}, \\ \frac{L-1}{\sum_{i=1}^{L-1} \exp\left(\frac{\sum_{k=1}^m (a_i)_k}{m}\right)} & \text{if } a_i \in \mathbb{R}^m, \end{cases} \quad (\text{A.3.4})$$

and the appended term for the loss function is

$$SRT = W_a \mathcal{S}(a) \quad (\text{A.3.5})$$

with trainable weight W_a . It was observed that networks utilizing adaptive activation functions exhibited more rapid decay of the loss function over networks using fixed activation functions [162]. Moreover, locally-adaptive networks outperformed both fixed-activation and global-adaptive networks[164]

APPENDIX B

MAGMA RESERVOIR APPENDIX

B.1 Stability and Verification via Convergence Tests

Owing to the use of an explicit time-stepping scheme, it is necessary to establish conditions for which the scheme outlined in Section 3.3.2 is stable. As an initial calculation, note that

$$\mathbf{E}\mathbf{A}\underline{\sigma} = \frac{\mu}{\eta} \text{dev } \underline{\sigma}. \quad (\text{B.1.1})$$

The deviatoric operator in Equation (B.1.1) can be expressed as a matrix-vector multiplication, namely

$$\mathbf{E}\mathbf{A}\underline{\sigma} = \frac{\mu}{\eta} \mathcal{D}\underline{\sigma}, \quad (\text{B.1.2})$$

if second-order tensors are stacked into vectors (across rows and removing symmetries)

$$\underline{\sigma} = [\sigma_{rr}, \sigma_{rz}, \sigma_{zz}, \sigma_{\theta\theta}]^T, \quad (\text{B.1.3})$$

and matrix $\mathcal{D}$ is given by

$$\mathcal{D} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & 0 \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (\text{B.1.4})$$

The non-dimensionalized explicit forward-Euler discretization of the aging law (Equation (3.3.20)) can therefore be expressed as

$$\tilde{\underline{\mathbf{C}}}^{n+1} = (\mathbf{I} - \Delta t D e^{-1} \mathcal{D}) \tilde{\underline{\mathbf{C}}}^n + \Delta t D e^{-1} \mathcal{D} \tilde{\underline{\mathbf{E}}}^n, \quad (\text{B.1.5})$$

the stability of which is determined by the eigenvalues of the growth-factor matrix $\mathbf{I} - \Delta t D e^{-1} \mathcal{D}$ and whether we can bound its spectral radius using an appropriate choice for Δt . Eigenvalues for the growth-factor matrix are

$$\lambda_1 = 1, \quad (\text{B.1.6a})$$

$$\lambda_2 = 1 - \frac{2}{3}\Delta\tilde{t}De^{-1}, \quad (\text{B.1.6b})$$

$$\lambda_3 = 1 - \Delta\tilde{t}De^{-1}, \quad (\text{B.1.6c})$$

where λ_3 appears as a repeated eigenvalue. To bound their magnitudes by at most 1 demands that $\Delta\tilde{t}$ be smaller than $2De$. In addition, the time step must be sufficiently small to resolve any time-varying boundary data. In this work this amounts to resolving the sinusoidal boundary data at the reservoir wall. Since the corresponding (angular) Nyquist frequency for $\sin(\tilde{t})$ is 1, the largest time step that resolves this frequency is $\delta\tilde{t} = \pi$, and should be (in practice) a small fraction of this. A sufficient, stable time step is then chosen by

$$\Delta\tilde{t} \leq \min\{2De, \delta\tilde{t}\}. \quad (\text{B.1.7})$$

In practice we use more restrictive criteria, namely,

$$\Delta\tilde{t} \leq \min\left\{\frac{De}{4}, \frac{\delta\tilde{t}}{2}\right\}. \quad (\text{B.1.8})$$

Except for a few limiting cases, the temperature-dependent material parameters will cause $\frac{De}{4}$ to be the agent that restricts time-step. Figure B.1 illustrates the number of timesteps required to resolve pressure as a function of forcing period, illustrating the increasing computational cost associated with very small Deborah numbers. We note that for general (i.e. non-sinusoidal) pressure histories, the minimum time step should be smaller than $2De$ and also be small enough to resolve relevant features of the pressurization forcing.

We verify the accuracy of our numerical method using the method of manufactured solutions (MMS) [89] and explain this technique in the context of the dimensional problem (computationally we solve the non-dimensionalized problem). The MMS verification technique lets us choose arbitrary solution fields $u^*(r, z, t), C^*(r, z, t)$ to act as exact solutions to any initial-boundary-value problem, even those without a known analytic solution) necessary for measuring convergence. The key point is that u^* and C^* satisfy the governing equations and boundary conditions with particular choices of source terms and boundary data which we detail in this section.

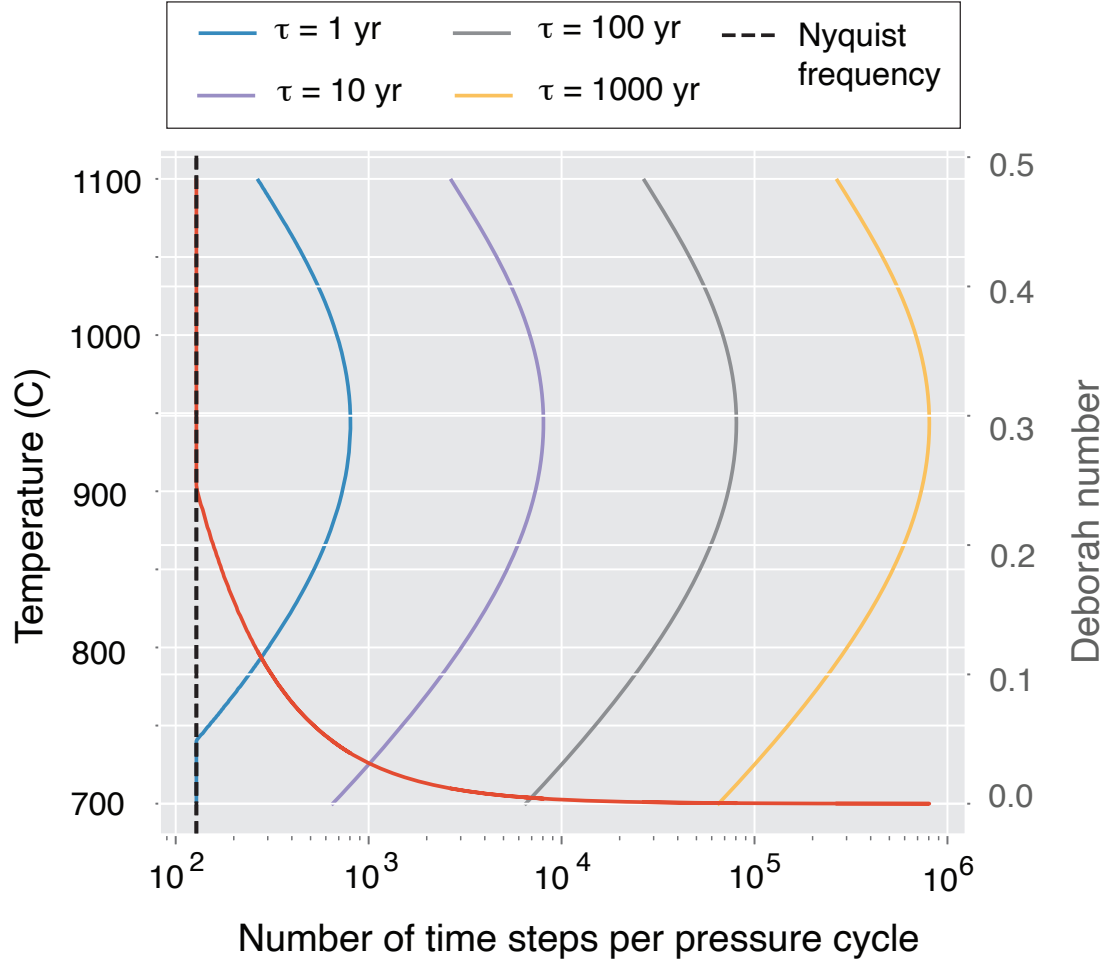


Fig. B.1 Number of timesteps required to simulate pressure forcing of various periods. Number of timesteps decreases with increasing Deborah number (red curve), until the Nyquist limit is reached (dashed curve). Number of timesteps per period is a non-monotonic function of temperature (other colored curves) because both elastic moduli and viscosity are temperature dependent.

We choose a manufactured solution to the initial-boundary-value problem Equation (3.2.1a),(3.2.4)-(3.2.8) based on the well-known solution to the pressurized magma cavity problem in an elastic half-space [60], [116] given by

$$\mathbf{u}_e = \frac{P_0 a^3}{4\mu(r^2 + z^2)^{3/2}} \begin{bmatrix} r \\ z \end{bmatrix}. \quad (\text{B.1.9})$$

which satisfies the reservoir pressure conditions Equations (3.3.10a)-(3.3.10b). Define the manufactured solutions u^*, C^* by

$$u^*(r, z, t) = (2 - e^{-t})\mathbf{u}_e, \quad (\text{B.1.10})$$

$$C^*(r, z, t) = (1 - e^{-t})\mathbf{E}\underline{\underline{\epsilon}}(\mathbf{u}_e), \quad (\text{B.1.11})$$

which satisfies equilibrium and specifies all boundary data. It does not however satisfy the aging law, and to correct for this discrepancy a source term is added, namely

$$\dot{\underline{\underline{C}}} = \mathbf{E}\mathbf{A}\underline{\underline{\sigma}} + \mathbf{G}. \quad (\text{B.1.12})$$

Here, the source term $\mathbf{G}$ is determined from the manufactured solutions to be

$$\mathbf{G} = e^{-t}\sigma^* - \frac{\mu}{\eta} \text{dev } \sigma^*, \quad (\text{B.1.13})$$

where σ^* is the manufactured stress and can be obtained by computing

$$\sigma^* = \mathbf{E}\underline{\underline{\epsilon}}(\mathbf{u}_e). \quad (\text{B.1.14})$$

All parameters used are given in Table B.1. Table B.2 shows the spatial errors $\|\underline{\underline{C}} - \underline{\underline{C}}_h\|$ and $\|\mathbf{u} - \mathbf{u}_h\|$ when computing approximations to C^* and u^* after a single time step, using a stable step size of 10^{-7} and the discrete L^2 -norm. Successive mesh refinements are made using polynomials of degree 3 as a basis for the FEM space. Convergence rates agrees with FEM theory which predict a convergence rate of $p+1$ for u^* and p for C^* when polynomials of degree p are used [165]. The same convergence pattern is observed for polynomials with degree greater than 3 except that the L^2 -error

Table B.1. Parameters used in Convergence Tests and their Symbols.

Symbol	Explanation	Value
a	Ellipse semi-major axis	4 km
b	Ellipse semi-minor axis	4 km
D	Reservoir depth beneath Earth's surface	5 km
L_r	Domain length	10 km
L_z	Domain depth	10 km
μ	shear modulus	0.5 GPa
λ	Lamé's first parameter	4 GPa
η	Viscosity	0.5 GPa-s
P_0	Chamber Pressure	10 MPa

Table B.2. Spatial convergence data, measured with respect to the discrete L^2 -norm, for a single time step of $\Delta t = 10^{-7}$ using polynomials of degree 3.

h	$\ \underline{\mathbf{C}} - \underline{\mathbf{C}}_h\ $	$\underline{\mathbf{C}}$ -rate	$\ \mathbf{u} - \mathbf{u}_h\ $	$\mathbf{u}$ -rate
$h/2$	5.25×10^{-9}		1.84×10^{-8}	
$h/4$	7.17×10^{-10}	2.87	1.31×10^{-9}	3.81
$h/8$	9.13×10^{-11}	2.97	8.41×10^{-11}	3.96
$h/16$	1.14×10^{-11}	3.00	5.24×10^{-12}	4.00

drops below machine precision leading to round-off error in the rate computation.

To measure the convergence in the temporal domain we select a single point in space and perform successive mesh refinements in time. Table B.3 shows that both $\underline{\mathbf{C}}$ and $\mathbf{u}$ exhibit rate-1 temporal convergence, consistent with forward Euler.

The benefit of convergence tests based on the MMS technique is that solutions can be manufactured for problems with more physical complexities, as opposed to relying on simple problems with known analytic solutions such as those highlighted in [86]. With MMS, rigorous convergence can be obtained at the exact theoretical rate, a desirable outcome for high-order numerical methods. A drawback is that MMS requires making specific choices for source and boundary data, which can sometimes alter the underlying physics of interest. Thus code verification can benefit further from community based efforts, as done extensively in the earthquake community

Table B.3. Temporal convergence data measured at point $(\tilde{A}, 0)$ under the discrete L^2 -norm.

Δt	$\ \underline{\mathbf{C}} - \underline{\mathbf{C}}_h\ $	$\underline{\mathbf{C}}$ -rate	$\ \mathbf{u} - \mathbf{u}_h\ $	$\mathbf{u}$ -rate
$\Delta t/2$	1.75×10^{-1}		1.18×10^{-6}	
$\Delta t/4$	8.85×10^{-2}	0.99	5.96×10^{-7}	0.99
$\Delta t/8$	4.46×10^{-2}	0.99	3.01×10^{-7}	0.99

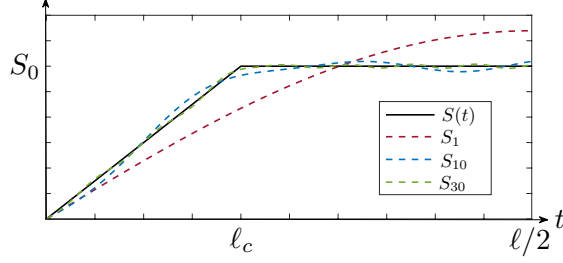


Fig. B.2 Example of a pressurization time series consisting of a linear ramp to a maximum value S_0 at $t = \ell_c$. The Fourier-sine series representation (black) along with partial sums $S_N = \sum_{n=1}^N b_n \sin(\omega_n t)$ for $N = 1, 10, 30$. With specific values $\ell = 20, \ell_c = 4$ and $S_0 = 10$, the discrete L^2 -errors are 4.97, 0.931 and 0.284, respectively, illustrating convergence of the series.

[98], [143]. In community benchmarking, all mathematical details of a problem are specified and different modeling groups compare code output and seek quantitative comparisons. These exercises can be done for problems with or without a known analytic solution.

B.2 Construction of general time-varying signals

Although we focus on sinusoidal forcing functions here, our framework may be generalized in principle to any transient deformation problem. This generalization follows from superimposing sinusoidal basis functions with appropriate weights.

The example of the linear pressure ramp followed by a hold in (3.3.7) is plotted in Figure B.2, with various partial sum approximations $S_N(t) = \sum_{n=1}^N b_n \sin(\omega_n t)$, also plotted for several N . The L^2 -error made by approximating $S(t)$ by $S_N(t)$ decreases with increasing N , illustrating convergence. So long as the stable time stepping requirements for successively higher frequency contributions are met, deformation associated with complex forcing functions can be analyzed using the frequency domain approach developed here, with output (e.g., surface deformation) following from the (3.6.2).

Bibliography

- [1] G. Bal, “Introduction to inverse problems,” *Lecture Notes-Department of Applied Physics and Applied Mathematics, Columbia University, New York*, 2012.
- [2] R. Bellman, “Dynamic programming,” *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
- [3] 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*, vol. 378, pp. 686–707, 2019. DOI: 10.1016/j.jcp.2018.10.045.
- [4] P. Kidger and T. Lyons, “Universal approximation with deep narrow networks,” in *Conference on learning theory*, PMLR, 2020, pp. 2306–2327.
- [5] K. Hornik, M. Stinchcombe and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [6] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.
- [7] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.
- [8] M. Leshno, V. Y. Lin, A. Pinkus and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.
- [9] B. Hanin and M. Sellke, “Approximating continuous functions by relu nets of minimal width,” *arXiv preprint arXiv:1710.11278*, 2017.
- [10] Z. Lu, H. Pu, F. Wang, Z. Hu and L. Wang, “The expressive power of neural networks: A view from the width,” *Advances in neural information processing systems*, vol. 30, 2017.
- [11] B. Hanin, “Universal function approximation by deep neural nets with bounded width and relu activations,” *Mathematics*, vol. 7, no. 10, p. 992, 2019.
- [12] I. E. Lagaris, A. Likas and D. I. Fotiadis, “Artificial neural networks for solving ordinary and partial differential equations,” *IEEE transactions on neural networks*, vol. 9, no. 5, pp. 987–1000, 1998. DOI: 10.1109/72.712178.

- [13] I. E. Lagaris, A. C. Likas and D. G. Papageorgiou, “Neural-network methods for boundary value problems with irregular boundaries,” *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000. DOI: 10.1109/72.870037.
- [14] J. Berg and K. Nyström, “A unified deep artificial neural network approach to partial differential equations in complex geometries,” *Neurocomputing*, vol. 317, pp. 28–41, 2018. DOI: 10.1016/j.neucom.2018.06.056.
- [15] S. Kollmannsberger, D. D’Angella, M. Jokeit, L. Herrmann *et al.*, *Deep Learning in Computational Mechanics*. Springer, 2021.
- [16] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi and F. Piccialli, “Scientific machine learning through physics-informed neural networks: Where we are and what’s next,” *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022. DOI: 10.1007/s10915-022-01939-z.
- [17] I. Bogaert, “Iteration-free computation of gauss-legendre quadrature nodes and weights,” *SIAM Journal on Scientific Computing*, vol. 36, no. 3, A1008–A1026, 2014.
- [18] I. Bogaert, B. Michiels and J. Fostier, “O(1) computation of legendre polynomials and gauss-legendre nodes and weights for parallel computing,” *SIAM Journal on Scientific Computing*, vol. 34, no. 3, pp. C83–C101, 2012.
- [19] D. W. Crews, “Arbitrarily high order implicit ode integration by correcting a neural network approximation with newton’s method,” *arXiv preprint arXiv:2203.00147*, 2022.
- [20] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021. DOI: 10.1038/s42254-021-00314-5.
- [21] W. Wang BoZhang and W. Cai, “Multi-scale deep neural network (mscalednn) methods for oscillatory stokes flows in complex domains,” *Communications in Computational Physics*, vol. 28, no. 5, pp. 2139–2157, 2020, ISSN: 1991-7120. DOI: 10.4208/cicp.0A-2020-0192.
- [22] L. Sun, H. Gao, S. Pan and J.-X. Wang, “Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data,” *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112732, 2020. DOI: 10.1016/j.cma.2019.112732.

- [23] X. Jin, S. Cai, H. Li and G. E. Karniadakis, “Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations,” *Journal of Computational Physics*, vol. 426, p. 109951, 2021. DOI: 10.1016/j.jcp.2020.109951.
- [24] S. Cai, Z. Wang, S. Wang, P. Perdikaris and G. E. Karniadakis, “Physics-informed neural networks for heat transfer problems,” *Journal of Heat Transfer*, vol. 143, no. 6, 2021. DOI: 10.1115/1.4050542.
- [25] E. Haghighat, M. Raissi, A. Moure, H. Gomez and R. Juanes, “A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics,” *Computer Methods in Applied Mechanics and Engineering*, vol. 379, p. 113741, 2021. DOI: 10.1016/j.cma.2021.113741.
- [26] S. Goswami, C. Anitescu, S. Chakraborty and T. Rabczuk, “Transfer learning enhanced physics informed neural network for phase-field modeling of fracture,” *Theoretical and Applied Fracture Mechanics*, vol. 106, p. 102447, 2020. DOI: 10.1016/j.tafmec.2019.102447.
- [27] Z. Mao, A. D. Jagtap and G. E. Karniadakis, “Physics-informed neural networks for high-speed flows,” *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020. DOI: 10.1016/j.cma.2019.112789.
- [28] E. Kharazmi, Z. Zhang and G. E. Karniadakis, “Variational physics-informed neural networks for solving partial differential equations,” *arXiv preprint arXiv:1912.00873*, 2019. DOI: 10.48550/arXiv.1912.00873.
- [29] A. D. Jagtap, E. Kharazmi and G. E. Karniadakis, “Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020. DOI: 10.1016/j.cma.2020.113028.
- [30] A. A. Ramabathiran and P. Ramachandran, “Spinn: Sparse, physics-based, and partially interpretable neural networks for pdes,” *Journal of Computational Physics*, vol. 445, p. 110600, 2021.
- [31] P. G. Ciarlet, *The finite element method for elliptic problems*. Siam, 2002, vol. 40.

- [32] A. Ern and J.-L. Guermond, *Theory and practice of finite elements*. Springer, 2004, vol. 159. DOI: 10.1007/978-1-4757-4355-5.
- [33] S. C. Brenner, L. R. Scott and L. R. Scott, *The mathematical theory of finite element methods*. Springer, 2008, vol. 3.
- [34] S. Larsson and V. Thomée, *Partial differential equations with numerical methods*. Springer, 2003, vol. 45.
- [35] J. Sirignano and K. Spiliopoulos, “Dgm: A deep learning algorithm for solving partial differential equations,” *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.
- [36] Z. Li, N. Kovachki, K. Azizzadenesheli *et al.*, “Fourier neural operator for parametric partial differential equations,” *arXiv preprint arXiv:2010.08895*, 2020.
- [37] Z. Li, N. Kovachki, K. Azizzadenesheli *et al.*, “Neural operator: Graph kernel network for partial differential equations,” *arXiv preprint arXiv:2003.03485*, 2020.
- [38] Z. Li, N. Kovachki, K. Azizzadenesheli *et al.*, “Multipole graph neural operator for parametric partial differential equations,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6755–6766, 2020.
- [39] L. Lu, P. Jin, G. Pang, Z. Zhang and G. E. Karniadakis, “Learning nonlinear operators via deepnet based on the universal approximation theorem of operators,” *Nature Machine Intelligence*, vol. 3, pp. 218–229, 2021. DOI: 10.1038/s42256-021-00302-5.
- [40] T. Chen and H. Chen, “Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems,” *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 911–917, 1995.
- [41] A. D. Back and T. Chen, “Universal approximation of multiple nonlinear operators by neural networks,” *Neural Computation*, vol. 14, no. 11, pp. 2561–2566, 2002.
- [42] S. Wang, H. Wang and P. Perdikaris, “Learning the solution operator of parametric partial differential equations with physics-informed deepnets,” *Science advances*, vol. 7, no. 40, eabi8605, 2021. DOI: 10.1126/sciadv.abi8605.
- [43] G. Kutyniok, “The mathematics of artificial intelligence,” *arXiv preprint arXiv:2203.08890*, 2022. DOI: 10.48550/arXiv.2203.08890.

- [44] D. Elbrächter, D. Perekrestenko, P. Grohs and H. Bölcskei, “Deep neural network approximation theory,” *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2581–2623, 2021.
- [45] Y. Shin, Z. Zhang and G. E. Karniadakis, “Error estimates of residual minimization using neural networks for linear pdes,” *Journal of Machine Learning for Modeling and Computing*, vol. 4, no. 4, pp. 73–101, 2023, ISSN: 2689-3967. DOI: 10.1615/JMachLearnModelComput.2023050411.
- [46] T. De Ryck, S. Lanthaler and S. Mishra, “On the approximation of functions by tanh neural networks,” *Neural Networks*, vol. 143, pp. 732–750, 2021. DOI: 10.1016/j.neunet.2021.08.015.
- [47] L. Lu, Y. Su and G. E. Karniadakis, “Collapse of deep and narrow neural nets,” *arXiv preprint arXiv:1808.04947*, 2018.
- [48] J. Shin YeonjongDarbon and G. E. Karniadakis, “On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes,” *Communications in Computational Physics*, vol. 28, no. 5, pp. 2042–2074, 2020, ISSN: 1991-7120. DOI: 10.4208/cicp.0A-2020-0193.
- [49] T. De Ryck, A. D. Jagtap and S. Mishra, “Error estimates for physics informed neural networks approximating the navier-stokes equations,” *arXiv preprint arXiv:2203.09346*, 2022. DOI: 10.48550/arXiv.2203.09346.
- [50] S. Mishra and R. Molinaro, “Estimates on the generalization error of physics-informed neural networks for approximating pdes,” *IMA Journal of Numerical Analysis*, 2022. DOI: 10.1093/imanum/drab093.
- [51] S. Mishra and R. Molinaro, “Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes,” *IMA Journal of Numerical Analysis*, vol. 42, no. 2, pp. 981–1022, 2022. DOI: 10.1093/imanum/drab032.
- [52] A. Hinrichs, E. Novak, M. Ullrich and H. Woźniakowski, “The curse of dimensionality for numerical integration of smooth functions,” *Mathematics of Computation*, vol. 83, no. 290, pp. 2853–2863, 2014.

- [53] K. A. Anderson and P. Segall, “Physics-based models of ground deformation and extrusion rate at effusively erupting volcanoes,” *Journal of Geophysical Research Solid Earth*, vol. 116, no. B7, pp. 1–20, 2011. DOI: 10.1029/2010JB007939.
- [54] S. Cianetti, C. Giunchi and E. Casarotti, “Volcanic deformation and flank instability due to magmatic sources and frictional rheology: The case of mount etna,” *Geophysical Journal International*, vol. 191, 939–953, 2012. DOI: 10.1111/j.1365-246X.2012.05689.x.
- [55] S. T. Henderson and M. E. Pritchard, “Time-dependent deformation of Uturuncu volcano, Bolivia, constrained by GPS and InSAR measurements and implications for source models,” *Geosphere*, vol. 13, no. 6, 1834–1854, 2017. DOI: 10.1130/GES01203.1.
- [56] D. Walwer, M. Ghil and E. Calais, “Oscillatory nature of the Okmok volcano’s deformation,” *Philosophical Transactions of the Royal Society A*, vol. 506, pp. 76–86, 2021. DOI: 10.1016/j.epsl.2018.10.033.
- [57] A. Gudmundsson, “Effect of tensile stress concentration around magma chambers on intrusion and extrusion frequencies,” *Journal of Volcanology and Geothermal Research*, vol. 35, no. 3, pp. 179–194, 1988. DOI: 10.1016/0377-0273(88)90015-7.
- [58] R. S. J. Sparks, K. Cashman and E. Calais, “Dynamic magma systems: Implications for forecasting volcanic activity,” *Elements*, vol. 13, no. 1, pp. 35–40, 2017. DOI: 10.2113/gselements.13.1.35.
- [59] P. Segall, “Magma chambers: What we can, and cannot, learn from volcano geodesy,” *Philosophical Transactions of the Royal Society A*, vol. 377, no. 2139, 2019. DOI: 10.1098/rsta.2018.0158.
- [60] K. Mogi, “Relations between the eruptions of various volcanoes and the deformations of the ground surfaces around them,” *Bulletin of the Earthquake Research Institute*, vol. 36, pp. 99–134, 1958.
- [61] D. F. McTigue, “Elastic stress and deformation near a finite spherical magma body: Resolution of the point source paradox,” *Journal of Geophysical Research: Solid Earth*, vol. 92, no. B12, pp. 12 931–12 940, 1987. DOI: 10.1029/JB092iB12p12931.

- [62] G. Berrino, G. Corrado, G. Luongo and B. Toro, “Ground deformation and gravity changes accompanying the 1982 Pozzuoli uplift,” *Bulletin volcanologique*, vol. 47, no. 2, pp. 187–200, 1984. DOI: 10.1007/BF01961548.
- [63] T. Mittal and M. A. Richards, “Volatile degassing from magma chambers as a control on volcanic eruptions,” *Journal of Geophysical Research Solid Earth*, vol. 124, no. 9, pp. 7869–7901, 2019. DOI: 10.1029/2018JB016983.
- [64] J. Gottsmann and H. Odbert, “The effects of thermomechanical heterogeneities in island arc crust on time-dependent preeruptive stresses and the failure of an andesitic reservoir,” *Journal of Geophysical Research: Solid Earth*, vol. 119, pp. 4626–4639, 2014. DOI: 10.1002/2014JB011079.
- [65] T. Yamasaki, T. Kobayashi, T. Wright and Y. Fukahata, “Viscoelastic crustal deformation by magmatic intrusion: A case study in the Kutcharo caldera, eastern Hokkaido, Japan,” *Journal of Volcanology and Geothermal Research*, vol. 349, pp. 128–145, 2018. DOI: 10.1016/j.jvolgeores.2017.10.011.
- [66] C. Novoa, D. Remy, M. Gerbault *et al.*, “Viscoelastic relaxation: A mechanism to explain the decennial large surface displacements at the Laguna del Maule silicic volcanic complex,” *Earth and Planetary Science Letters*, vol. 521, pp. 46–59, 2019. DOI: 10.1016/j.epsl.2019.06.005.
- [67] M. Dragoni and C. Magnanensi, “Displacement and stress produced by a pressurized, spherical magma chamber, surrounded by a viscoelastic shell,” *Physics of the Earth and Planetary Interiors*, vol. 56, no. 3, pp. 316–328, 1989. DOI: 10.1016/0031-9201(89)90166-0.
- [68] L. Karlstrom, J. Dufek and M. Manga, “Magma chamber stability in arc and continental crust,” *Journal of Volcanology and Geothermal Research*, vol. 190, no. 3, pp. 249–270, 2010. DOI: 10.1016/j.jvolgeores.2009.10.003.
- [69] P. Gregg, S. De Silva and E. Grosfils, “Thermomechanics of shallow magma chamber pressurization: Implications for the assessment of ground deformation data at active volcanoes,” *Earth and Planetary Science Letters*, vol. 384, pp. 100–108, 2013. DOI: 10.1016/j.epsl.2013.09.040.

- [70] Y. Liao, S. A. Soule, M. Jones and H. Le Mével, “The mechanical response of a magma chamber with poroviscoelastic crystal mush,” *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 4, e2020JB019395, 2021. DOI: 10.1029/2020JB019395.
- [71] A. M. Jellinek and D. J. DePaolo, “A model for the origin of large silicic magma chambers: Precursors of caldera-forming eruptions,” *Bulletin of Volcanology*, vol. 65, no. 5, pp. 363–381, 2003. DOI: 10.1007/s00445-003-0277-y.
- [72] W Degruyter and C Huber, “A model for eruption frequency of upper crustal silicic magma chambers,” *Earth and Planetary Science Letters*, vol. 403, pp. 117–130, 2014. DOI: 10.1016/j.epsl.2014.06.047.
- [73] A. Newman, T. H. Dixon, G. Ofoegbu and J. E. Dixon, “Geodetic and seismic constraints on recent activity at long valley caldera, california: Evidence for viscoelastic rheology,” *Journal of Volcanology and Geothermal Research*, vol. 105, no. 3, pp. 183–206, 2001. DOI: 10.1016/S0377-0273(00)00255-9.
- [74] F. Sigmundsson, V. Pinel, B. Lund *et al.*, “Climate effects on volcanism: Influence on magmatic systems of loading and unloading from ice mass variations, with examples from Iceland,” *Philosophical Transactions of the Royal Society A*, vol. 368, no. 1919, 2010. DOI: 10.1098/rsta.2010.0042.
- [75] T Masterlark, M Haney, H Dickinson, T Fournier and C Searcy, “Rheologic and structural controls on the deformation of okmok volcano, alaska: Fems, insar, and ambient noise tomography,” *Journal of Geophysical Research: Solid Earth*, vol. 115, pp. 1–22, 2010. DOI: 10.1029/2009JB006324.
- [76] H. Le Mével, P. Gregg and K. Feigl, “Magma injection into a long-lived reservoir to explain geodetically measured uplift: Application to the 2007-2014 unrest episode at Laguna del Maule volcanic field, Chile,” *Journal of Geophysical Research: Solid Earth*, vol. 121, pp. 6092–6108, 2016. DOI: 10.1002/2016JB013066.
- [77] A. M. Morales Rivera, F Amelung, F. Albino and P. M. Gregg, “Impact of crustal rheology on temperature-dependent viscoelastic models of volcano deformation: Application to Taal Volcano, Philippines,” *Journal of Geophysical Research: Solid Earth*, vol. 115, pp. 978–994, 2019. DOI: 10.1029/2018JB016054.

- [78] L. Karlstrom, S. R. Paterson and A. M. Jellinek, “A reverse energy cascade for crustal magma transport,” *Nature Geoscience*, vol. 10, no. 8, pp. 604–608, 2017. DOI: 10.1038/ngeo2982.
- [79] C. Del Negro, G. Currenti and D. Scandura, “Temperature-dependent viscoelastic modeling of ground deformation: Application to Etna volcano during the 1993–1997 inflation period,” *Physics of the Earth and Planetary Interiors*, vol. 172, no. 3, pp. 299–309, 2009. DOI: 10.1016/j.pepi.2008.10.019.
- [80] P. Gregg, S. De Silva, E. Grosfils and J. Parmigiani, “Catastrophic caldera-forming eruptions: Thermomechanics and implications for eruption triggering and maximum caldera dimensions on earth,” *Journal of Volcanology and Geothermal Research*, vol. 241, pp. 1–12, 2012. DOI: 10.1016/j.jvolgeores.2012.06.009.
- [81] M. Head, J. Hickey, J. Gottsmann and N. Fournier, “Exploring the impact of thermally controlled crustal viscosity on volcanic ground deformation,” *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 8, e2020JB020724, 2021. DOI: 10.1029/2020JB020724.
- [82] M Bonafede, M Dragoni and F Quarenì, “Displacement and stress fields produced by a centre of dilation and by a pressure source in a viscoelastic half-space: Application to the study of ground deformation and seismic activity at Campi Flegrei, Italy,” *Geophysical Journal International*, vol. 87, no. 2, pp. 455–485, 1986. DOI: 10.1111/j.1365-246X.1986.tb06632.x.
- [83] P. Segall, “Repressurization following eruption from a magma chamber with a viscoelastic aureole,” *Journal of Geophysical Research: Solid Earth*, vol. 121, no. 12, pp. 8501–8522, 2016. DOI: 10.1002/2016JB013597.
- [84] M. Townsend, C. Huber, W. Degruyter and O. Bachmann, “Magma chamber growth during intercaldera periods: Insights from thermo-mechanical modeling with applications to Laguna del Maule, Campi Flegrei, Santorini, and Aso,” *Geochemistry, Geophysics, Geosystems*, vol. 20, no. 3, pp. 1574–1591, 2019. DOI: 10.1029/2018GC008103.
- [85] M. Head, J. Hickey, J. Gottsmann and N. Fournier, “The influence of viscoelastic crustal rheologies on volcanic ground deformation: Insights from models of pressure and volume

- change,” *Journal of Geophysical Research: Solid Earth*, vol. 124, no. 8, pp. 8127–8146, 2019. DOI: 10.1029/2019JB017832.
- [86] J. Hickey and J. Gottsmann, “Benchmarking and developing numerical finite element models of volcanic deformation,” *Journal of Volcanology and Geothermal Research*, vol. 280, pp. 126–130, 2014. DOI: 10.1016/j.jvolgeores.2014.05.011.
- [87] M. Head, J. Hickey, J. Thompson, J. Gottsmann and N. Fournier, “Rheological controls on magma reservoir failure in a thermo-viscoelastic crust,” *Journal of Geophysical Research: Solid Earth*, e2021JB023439, 2022. DOI: 10.1029/2021JB023439.
- [88] J. Crozier and L. Karlstrom, “Evolving magma temperature and volatile contents over the 2008–2018 summit eruption of kilauea volcano,” *Science Advances*, vol. 8, no. 22, eabm4310, 2022. DOI: 10.1126/sciadv.abm4310. [Online]. Available: <https://www.science.org/doi/abs/10.1126/sciadv.abm4310>.
- [89] P. J. Roache, *Verification and validation in computational science and engineering*. Hermosa Albuquerque, NM, 1998, vol. 895.
- [90] R. Muki and E. Sternberg, “On Transient Thermal Stresses in Viscoelastic Materials With Temperature-Dependent Properties,” *Journal of Applied Mechanics*, vol. 28, no. 2, pp. 193–207, Jun. 1961. DOI: 10.1115/1.3641651.
- [91] H. C. Lau and B. K. Holtzman, ““Measures of dissipation in viscoelastic media” extended: Toward continuous characterization across very broad geophysical time scales,” *Geophysical Research Letters*, vol. 46, no. 16, pp. 9544–9553, 2019. DOI: 10.1029/2019GL083529.
- [92] H. C. Lau, B. K. Holtzman and C. Havlin, “Toward a self-consistent characterization of lithospheric plates using full-spectrum viscoelasticity,” *AGU Advances*, vol. 1, no. 4, e2020AV000205, 2020. DOI: 10.1029/2020AV000205.
- [93] J. Schöberl, *NGSolve*, <http://ngsolve.org>, 2010–2022.
- [94] J. Schöberl, “NETGEN an advancing front 2D/3D-mesh generator based on abstract rules,” *Computing and Visualization in Science*, vol. 1, no. 1, pp. 41–52, 1997. DOI: 10.1007/S007910050004.

- [95] *Bitbucket: Magmaxisym*, <https://bitbucket.org/jayggg/magmaxisym/src/master/>, Repository with python drivers for computing dynamics of viscoelastic medium surrounding an axisymmetric magma cavity., 2022.
- [96] A. Ern and J.-L. Guermond, *Finite elements i*, 2021.
- [97] K. L. Allison and E. M. Dunham, “Earthquake cycle simulations with rate-and-state friction and power-law viscoelasticity,” *Tectonophysics*, vol. 733, pp. 232–256, 2018. DOI: 10.1016/j.tecto.2017.10.021.
- [98] B. A. Erickson, J. Jiang, M. Barall *et al.*, “The community code verification exercise for simulating sequences of earthquakes and aseismic slip (SEAS),” *Seismol. Res. Lett.*, vol. 91, pp. 874–890, 2020. DOI: 10.1785/0220190248.
- [99] J. Golden and G. Graham, *Boundary Value Problems in Linear Viscoelasticity*, 1st ed. Springer-Verlag, 1988, ISBN: 9780387186153. DOI: 10.1137/1031072.
- [100] J. Hickey, J. Gottsmann and P. Mothes, “Estimating volcanic deformation source parameters with a finite element inversion: The 2001–2002 unrest at cotopaxi volcano, ecuador,” *Journal of Geophysical Research: Solid Earth*, vol. 120, no. 3, pp. 1473–1486, 2015. DOI: 10.1002/2014JB011731.
- [101] B. A. Black and M. Manga, “Volatiles and the tempo of flood basalt magmatism,” *Earth and Planetary Science Letters*, vol. 458, pp. 130–140, 2017. DOI: 10.1016/j.epsl.2016.09.035.
- [102] C. Huber, M. Townsend, W. Degruyter and O. Bachmann, “Optimal depth of subvolcanic magma chamber growth controlled by volatiles and crust rheology,” *Nature Geoscience*, vol. 12, no. 9, pp. 762–768, 2019. DOI: 10.1038/s41561-019-0415-6.
- [103] R. Bürgmann and G. Dresen, “Rheology of the lower crust and upper mantle: Evidence from rock mechanics, geodesy, and field observations,” *Annual Review of Earth and Planetary Sciences*, vol. 36, pp. 531–567, 2008. DOI: 10.1146/annurev.earth.36.031207.124326.
- [104] O. Karakas, W. Degruyter, O. Bachmann and J. Dufek, “Lifetime and size of shallow magma bodies controlled by crustal-scale magmatism,” *Nature Geoscience*, vol. 10, no. 6, pp. 446–450, 2017. DOI: 10.1038/ngeo2959.

- [105] J. Gopalakrishnan and J. E. Pasciak, “The convergence of V-cycle multigrid algorithms for axisymmetric Laplace and Maxwell equations,” *Mathematics of Computation*, vol. 75, pp. 1697–1719, 2006. DOI: 10.1090/S0025-5718-06-01884-9.
- [106] R. R. Bakker, M. Frehner and M. Lupi, “How temperature-dependent elasticity alters host rock/magmatic reservoir models: A case study on the effects of ice-cap unloading on shallow volcanic systems,” *Earth and Planetary Science Letters*, vol. 456, pp. 16–25, 2016. DOI: 10.1016/j.epsl.2016.09.039.
- [107] M. Schetzen, *Linear Time-Invariant Systems*, 1st ed. The Institute of Electrical and Electronics Engineers, 2003.
- [108] M. Townsend, “Linking surface deformation to thermal and mechanical magma chamber processes,” *Earth and Planetary Science Letters*, vol. 577, pp. 117–272, 2022. DOI: 10.1016/j.epsl.2021.117272.
- [109] X. Zhong, M. Dabrowski and B. Jamtveit, “Analytical solution for the stress field in elastic half-space with a spherical pressurized cavity or inclusion containing eigenstrain,” *Geophysical Journal International*, vol. 216, no. 2, pp. 1100–1115, 2019. DOI: 10.1093/gji/ggy447.
- [110] A. Chen, B. Erickson and J. Kozdon, “Matrix-free methods for summation-by-parts finite difference operators on GPUs,” *submitted*, 2022.
- [111] J. M. Carcione and G. Quiroga-Goode, “Some aspects of the physics and numerical modeling of biot compressional waves,” *Journal of Computational Acoustics*, vol. 03, no. 04, pp. 261–280, 1995. DOI: 10.1142/S0218396X95000136.
- [112] L. Moresi, F. Dufour and H.-B. Mühlhaus, “Mantle convection modeling with viscoelastic/brittle lithosphere: Numerical methodology and plate tectonic modeling,” *Pure and applied Geophysics*, vol. 159, no. 10, pp. 2335–2356, 2002. DOI: 10.1007/s00024-002-8738-3.
- [113] J. Gopalakrishnan and J. Guzmán, “A second elasticity element using the matrix bubble,” *IMA J. Numer. Anal.*, vol. 32, pp. 352–372, 2012. DOI: 10.1093/imanum/drq047.

- [114] T. Heister, J. Dannberg, R. Gassmüller and W. Bangerth, “High accuracy mantle convection simulation through modern numerical methods. II: Realistic models and problems,” *Geophysical Journal International*, vol. 210, no. 2, pp. 833–851, 2017. DOI: 10.1093/gji/ggx195.
- [115] B. A. Erickson and E. M. Dunham, “An efficient numerical method for earthquake cycles in heterogeneous media: Alternating subbasin and surface-rupturing events on faults crossing a sedimentary basin,” *Journal of Geophysical Research-Solid Earth*, vol. 119, no. 4, pp. 3290–3316, 2014. DOI: 10.1002/2013JB010614.
- [116] P. Segall, *Earthquake and volcano deformation*. Princeton University Press, 2010. DOI: 10.1515/9781400833856.
- [117] G. Currenti and C. A. Williams, “Numerical modeling of deformation and stress fields around a magma chamber: Constraints on failure conditions and rheology,” *Physics of the Earth and Planetary Interiors*, vol. 226, pp. 14–27, 2014. DOI: 10.1016/j.pepi.2013.11.003.
- [118] E Rivalta, F Corbi, L. Passarelli, V Acocella, T. Davis and M. A. Di Vito, “Stress inversions to forecast magma pathways and eruptive vent location,” *Science advances*, vol. 5, no. 7, eaau9784, 2019. DOI: 10.1126/sciadv.aau9784.
- [119] R. Grapenthin, B. G. Ófeigsson, F. Sigmundsson, E. Sturkell and A. Hooper, “Pressure sources versus surface loads: Analyzing volcano deformation signal composition with an application to hekla volcano, iceland,” *Geophysical Research Letters*, vol. 37, no. 20, 2010. DOI: 10.1029/2010GL044590.
- [120] B. A. Erickson, E. M. Dunham and A. Khosravifar, “A finite difference method for off-fault plasticity throughout the earthquake cycle,” *Journal of the Mechanics and Physics of Solids*, vol. 109, pp. 50–77, 2017. DOI: 10.1016/j.jmps.2017.08.002.
- [121] B. Mullet and P. Segall, “The surface deformation signature of a transcrustal, crystal mush-dominant magma system,” *Journal of Geophysical Research: Solid Earth*, vol. 127, no. 5, e2022JB024178, 2022. DOI: 10.1029/2022JB024178.
- [122] R. O’connell and B Budiansky, “Measures of dissipation in viscoelastic media,” *Geophysical Research Letters*, vol. 5, no. 1, pp. 5–8, 1978. DOI: 10.1029/GL005i001p00005.

- [123] S. Ide, G. C. Beroza, D. R. Shelly and T. Uchide, “A scaling law for slow earthquakes,” *Nature*, vol. 447, no. 7140, pp. 76–79, 2007.
- [124] B. F. Atwater, M.-R. Satoko, S. Kenji, T. Yoshinobu, U. Kazue and D. K. Yamaguchi, *The Orphan Tsunami of 1700: Japanese Clues to a Parent Earthquake in North America*, 2nd ed. University of Washington Press, 2005, ISBN: 9780295985350. [Online]. Available: <http://www.jstor.org/stable/j.ctvcwnbrv> (visited on 30/06/2023).
- [125] E. E. Brodsky, J. J. Mori, L. Anderson *et al.*, “The state of stress on the fault before, during, and after a major earthquake,” *Annual Review of Earth and Planetary Sciences*, vol. 48, no. 1, pp. 49–74, 2020. DOI: 10.1146/annurev-earth-053018-060507.
- [126] National Academies of Sciences, Engineering, and Medicine and others, *A vision for NSF Earth sciences 2020-2030: Earth in time*. Washington, DC: National Academies Press, 2020.
- [127] K. J. Bergen, P. A. Johnson, M. V. de Hoop and G. C. Beroza, “Machine learning for data-driven discovery in solid earth geoscience,” *Science*, vol. 363, no. 6433, eaau0323, 2019. DOI: 10.1126/science.aau0323.
- [128] M. Kern, *Numerical methods for inverse problems*. John Wiley & Sons, 2016.
- [129] D. Givoli, “A tutorial on the adjoint method for inverse problems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 380, p. 113 810, 2021. DOI: 10.1016/j.cma.2021.113810.
- [130] Q. Kong, D. T. Trugman, Z. E. Ross, M. J. Bianco, B. J. Meade and P. Gerstoft, “Machine Learning in Seismology: Turning Data into Insights,” *Seismological Research Letters*, vol. 90, no. 1, pp. 3–14, 2018. DOI: 10.1785/0220180259.
- [131] J.-T. Lin, D. Melgar, A. M. Thomas and J. Searcy, “Early warning for great earthquakes from characterization of crustal deformation patterns with deep learning,” *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 10, e2021JB022703, 2021.
- [132] H. Kubo, M. Naoi and M. Kano, “Recent advances in earthquake seismology using machine learning,” *Earth, Planets and Space*, vol. 76, no. 1, p. 36, 2024.

- [133] J. Zhao, H. Ling, J. Liu, J. Wang, A. F. Burke and Y. Lian, “Machine learning for predicting battery capacity for electric vehicles,” *eTransportation*, vol. 15, p. 100214, 2023. DOI: 10.1016/j.etrans.2022.100214.
- [134] M. Raissi, A. Yazdani and G. E. Karniadakis, “Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations,” *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.
- [135] R. Fukushima, M. Kano and K. Hirahara, “Physics-informed neural networks for fault slip monitoring: Simulation, frictional parameter estimation, and prediction on slow slip events in a spring-slider system,” *ESS Open Archive*, 2023. DOI: 10.22541/essoar.168988460.01601423/v1.
- [136] T. Okazaki, T. Ito, K. Hirahara and N. Ueda, “Physics-informed deep learning approach for modeling crustal deformation,” *Nature Communications*, vol. 13, no. 1, p. 7092, 2022. DOI: 10.1038/s41467-022-34922-1.
- [137] S. Karimpouli and P. Tahmasebi, “Physics informed machine learning: Seismic wave equation,” *Geoscience Frontiers*, vol. 11, no. 6, pp. 1993–2001, 2020. DOI: 10.1016/j.gsf.2020.07.007.
- [138] B. Yu *et al.*, “The deep ritz method: A deep learning-based numerical algorithm for solving variational problems,” *Communications in Mathematics and Statistics*, vol. 6, no. 1, pp. 1–12, 2018. DOI: 10.1007/s40304-018-0127-z.
- [139] E. Kharazmi, Z. Zhang and G. E. Karniadakis, “Hp-vpinns: Variational physics-informed neural networks with domain decomposition,” *Computer Methods in Applied Mechanics and Engineering*, vol. 374, p. 113547, 2021. DOI: 10.1016/j.cma.2020.113547.
- [140] J. Müller and M. Zeinhofer, “Achieving high accuracy with pinns via energy natural gradients,” *arXiv preprint arXiv:2302.13163*, 2023. DOI: 10.48550/arXiv.2302.13163.
- [141] C. H. Scholz, *The Mechanics of Earthquakes and Faulting*, 3rd ed. Cambridge University Press, 2019. DOI: 10.1017/9781316681473.
- [142] C. Marone, “Laboratory-derived friction laws and their application to seismic faulting,” *Ann. Rev. Earth Pl. Sc.*, vol. 26, no. 1, pp. 643–696, 1998. DOI: 10.1146/annurev.earth.26.1.643.

- [143] R. A. Harris, M. Barall, R. Archuleta *et al.*, “The SCEC/USGS dynamic earthquake rupture code verification exercise,” *Seismological Research Letters*, vol. 80, pp. 119–126, 2009. DOI: 10.1785/gssrl.80.1.119.
- [144] J. H. Dieterich, “Modeling of rock friction 1. Experimental results and constitutive equations,” *Journal of Geophysical Research, [Solid Earth]*, vol. 84, no. B5, pp. 2161–2168, 1979, ISSN: 2169-9356. DOI: 10.1029/JB084iB05p02161.
- [145] A. Ruina, “Slip instability and state variable friction laws,” *J. Geophys. Res. Solid Earth*, vol. 88, no. B12, pp. 10 359–10 370, 1983. DOI: 10.1029/JB088iB12p10359.
- [146] M. van den Ende, J. Chen, J.-P. Ampuero and A. Niemeijer, “A comparison between rate-and-state friction and microphysical models, based on numerical simulations of fault slip,” *Tectonophysics*, vol. 733, pp. 273–295, 2018. DOI: 10.1016/j.tecto.2017.11.040.
- [147] P. Roache, *Verification and Validation in Computational Science and Engineering*, 1st ed. Albuquerque, NM: Hermosa Publishers, 1998.
- [148] T. W. Harvey, B. A. Erickson and J. E. Kozdon, “A high-order accurate summation-by-parts finite difference method for fully-dynamic earthquake sequence simulations within sedimentary basins,” *Journal of Geophysical Research: Solid Earth*, e2022JB025357, 2023. DOI: 10.1029/2022JB025357.
- [149] R. H. Byrd, P. Lu, J. Nocedal and C. Zhu, “A limited memory algorithm for bound constrained optimization,” *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995. DOI: 10.1137/0916069.
- [150] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
- [151] B. A. Erickson, J. Jiang, V. Lambert *et al.*, “Incorporating Full Elastodynamic Effects and Dipping Fault Geometries in Community Code Verification Exercises for Simulations of Earthquake Sequences and Aseismic Slip (SEAS),” *Bulletin of the Seismological Society of America*, vol. 113, no. 2, pp. 499–523, 2023. DOI: 10.1785/0120220066.

- [152] C. Wu, M. Zhu, Q. Tan, Y. Kartha and L. Lu, “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115 671, 2023.
- [153] L. Bottou, F. E. Curtis and J. Nocedal, “Optimization methods for large-scale machine learning,” *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018.
- [154] X. Qian and D. Klabjan, “The impact of the mini-batch size on the variance of gradients in stochastic gradient descent,” *arXiv preprint arXiv:2004.13146*, 2020.
- [155] C. Marone and B. Kilgore, “Scaling of the critical slip distance for seismic faulting with shear strain in fault zones,” *Nature*, vol. 362, no. 6421, pp. 618–621, 1993. DOI: 10.1038/362618a0.
- [156] A. Iserles, *A first course in the numerical analysis of differential equations*. Cambridge university press, 2009.
- [157] N. Hale and A. Townsend, “Fast and accurate computation of gauss–legendre and gauss–jacobi quadrature nodes and weights,” *SIAM Journal on Scientific Computing*, vol. 35, no. 2, A652–A674, 2013.
- [158] G. Cooper, “On the implementation of singly implicit runge-kutta methods,” *Mathematics of computation*, vol. 57, no. 196, pp. 663–672, 1991.
- [159] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.
- [160] D. Yarotsky, “Error bounds for approximations with deep relu networks,” *Neural Networks*, vol. 94, pp. 103–114, 2017.
- [161] M. Goyal, R. Goyal and B. Lall, “Learning activation functions: A new paradigm for understanding neural networks,” *arXiv preprint arXiv:1906.09529*, 2019.
- [162] F. Agostinelli, M. Hoffman, P. Sadowski and P. Baldi, “Learning activation functions to improve deep neural networks,” *arXiv preprint arXiv:1412.6830*, 2014.
- [163] A. D. Jagtap, K. Kawaguchi and G. E. Karniadakis, “Adaptive activation functions accelerate convergence in deep and physics-informed neural networks,” *Journal of Computational Physics*, vol. 404, p. 109 136, 2020.

- [164] A. D. Jagtap, K. Kawaguchi and G. Em Karniadakis, “Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks,” *Proceedings of the Royal Society A*, vol. 476, no. 2239, p. 20 200 334, 2020.
- [165] S. Larsson and V. Thomée, *Partial differential equations with numerical methods*. Springer Science & Business Media, 2008, vol. 45. DOI: 10.1007/978-3-540-88706-5.