A variational neural network approach for glacier modelling with nonlinear rheology

Tiangang Cui[a], Zhongjian Wang[b,*], Zhiwen Zhang[c,*]

[a]*School of Mathematics, Monash University, Victoria, Australia.*
[b]*Department of Statistics and CCAM, The University of Chicago, Chicago, IL 60637, USA.*
[c]*Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong SAR, China.*

Abstract

We propose a mesh-free method to solve the full Stokes equation for modeling the glacier dynamics with nonlinear rheology. Inspired by the Deep-Ritz method proposed in [1], we first formulate the solution to the non-Newtonian Stokes equation as the minimizer of a variational problem with boundary constraints. Then, we approximate its solution space by a deep neural network. The loss function for training the neural network is a relaxed version of the variational form, in which penalty terms are used to present soft constraints due to mixed boundary conditions. Instead of introducing mesh grids or basis functions to evaluate the loss function, our method only requires uniform sampling from the physical domain and boundaries. Furthermore, we introduce a re-normalization technique in the neural network to address the significant variation in the scaling of real-world problems. Finally, we illustrate the performance of our method by several numerical experiments, including a 2D model with the analytical solution, the Arolla glacier model with realistic scaling and a 3D model with periodic boundary conditions. Numerical results show that our proposed method is efficient in solving the non-Newtonian mechanics arising from glacier modeling with nonlinear rheology.

AMS subject classification: 35A15, 65J15, 68T99, 70K25, 76A05.

Keywords: Deep learning method; variational problems; mesh-free method; non-Newtonian mechanics; nonlinear rheology; glacier modelling.

1. Introduction

In recent years, deep neural networks (DNNs) have achieved unprecedented levels of success in a broad range of areas such as computer vision, speech recognition, natural language processing, and health sciences, producing results comparable or superior to human experts

*Corresponding author

Email addresses: tiangang.cui@monash.edu (Tiangang Cui), zhongjian@statistics.uchicago.edu (Zhongjian Wang), zhangzw@hku.hk (Zhiwen Zhang)

[30, 17]. The impacts have reached physical sciences where traditional first-principle based modeling and computational methodologies have been the norm. Thanks in part to the user-friendly open-source computing platforms from industry (e.g. TensorFlow and PyTorch), there have been vibrant activities in applying deep learning tools for scientific computing, such as approximating multivariate functions, solving ordinary/partial differential equations (ODEs/PDEs) and inverse problems using DNNs; see, e.g. [50, 13, 19, 27, 2, 64, 59] and references therein. There are many classical works on the approximation power of neural networks; see e.g. [11, 22, 14, 46]. For recent works on the expressive (approximation) power of DNNs; see, e.g. [10, 53, 62, 40, 35, 54]. In [19], the authors showed that DNNs with rectified linear unit (ReLU) activation function and enough width/depth contain the continuous piece-wise linear finite element space. Thus, one can represent a solution of PDE using the ReLU-DNN.

Solving ODEs or PDEs by a neural network (NN) approximation is known in the literature dating back at least to the 1990's; see e.g. [31, 39, 29]. The main idea in these works is to train NNs to approximate the solution by minimizing the residual of the ODEs or PDEs, along with the associated initial and boundary conditions. These early works estimate neural network solutions on a fixed mesh. Recently DNN methods are developed for Poisson and eigenvalue problems with a variational principle characterization (deep Ritz, [13]), for a class of high-dimensional parabolic PDEs with stochastic representations [18], for advancing finite element methods [20, 8, 7], for nonconvex energy minimization in simulating martensitic phase transitions [9], and for learning and generating invariant measures of stochastic dynamical systems with parameters [58]. The physics-informed neural network (PINN) method [48] and a deep Galerkin method (DGM) [55] compute PDE solutions based on their physical properties. For parametric PDEs, a deep operator network (DeepONet) learns operators accurately and efficiently from a relatively small dataset based on the universal approximation theorem of operators [36]; a Fourier neural operator method [32] directly learns the mapping from functional parametric dependence to the solutions of a family of PDEs. In [63, 1], weak adversarial network methods are studied for weak solutions and inverse problems, see also related studies on PDE recovery from data via DNN [34, 33, 47, 60] among others. In the context of surrogate modeling and uncertainty quantification (UQ), DNN methods include Bayesian deep convolutional encoder-decoder networks [64], deep multi-scale model learning [57], physics-constrained deep learning method [65], see also [27, 53, 26, 61] and references therein.

In this work, we present a deep learning method for solving problems in non-Newtonian mechanics that obey certain variational principles. In particular, we focus on nonlinear Stokes problems in which the viscosity nonlinearly depends on the strain rate. This type of problems plays a fundamental role in modelling geodynamic processes, for instance, the dynamics of glaciers [23, 45] and mantle convection [51, 38]. The solutions of these problems

typically face a combination of challenges, such as the presence of local features emerging from the nonlinear rheology, saddle point problems due to the incompressibility condition, complex problem domain with high aspect ratios, and high contrast boundary conditions. There has been a lot of effort in developing efficient numerical methods to address these challenges. For example, the work of [5, 15, 21, 24, 52, 56] designed and analyzed efficient and accurate high-order finite element discretization schemes. Delicated adaptation strategies, see [24, 49] and references therein, are developed to ensure that these high-order methods can successfully resolve local features of the flow field and a wide range of length scales. The resulting discretized nonlinear systems have to be solved using either Picard fixed-point iterations or Newton's method. Most of the recent high-performance solvers adopt modified Newton's iterations together with high-performance iterative linear solvers, see, e.g., [6, 16, 24, 37], to obtain a superlinear convergence rate.

Our work is inspired by the deep Ritz method proposed in [13] and our recent progress in developing deep learning method to solve interface problem [59]. We first formulate the PDEs into a variational problem, which leads to the objective function used by the neural network training. The second part consists of penalty terms arising from the boundary constraint of the governing PDE. In real world models, e.g. glacier sliding, the scale of the problem domain and the scale of physical parameters vary significantly. To address these challenges, we introduce a normalizing layer following the input of neural networks and design a strategy to balance penalty factors due to different boundary conditions. Using this combination of strategies, our proposed deep learning method is capable to solve nonlinear Stokes problems with different geometric scales, different parameter scales and boundary conditions using a universal configuration of network parameters.

The rest of the paper is organized as follows. In Section 2, we introduce the background of the Non-Newtonian ice flow model. In Section 3, we review the basic idea of deep neural network and formulate the variational neural network approach for solving the Non-Newtonian ice flow model. In addition, we also discuss some details of the implementation of our method. In Section 4, we present numerical results to demonstrate the accuracy of our method. Section 5 offers some concluding remarks.

2. Background

Since glaciers form one of the natural low-pass filters of atmospheric variability, modelling the mechanics of glaciers is instrumental in revealing slow changes in the climate system that might otherwise be obscured by short-term noise. We consider a canonical glacier model that treats the flow of ice as non-Newtonian, viscous, incompressible, and isothermal fluid in the steady-state [12]. In this section, we first present the strong form of the ice flow model, and then discuss its variational formulation that naturally yields the weak form of the ice flow model.

2.1. Non-Newtonian ice flow model

For an open, bounded domain $\Omega \subset \mathbb{R}^d$, we denote the velocity field (measured in meters per calendar year, i.e., $\mathrm{m\,a^{-1}}$) and the stress tensor (measured in Pa) of ice flow by $\mathbf{u} = (u_1, ..., u_d)^\top : \Omega \mapsto \mathbb{R}^d$ and $\boldsymbol{\sigma}_\mathbf{u} : \Omega \mapsto \mathbb{R}^{d\times d}$, respectively. The conservation laws of momentum and mass state that

$$-\nabla \cdot \boldsymbol{\sigma}_\mathbf{u} = \rho \mathbf{g}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where ρ is the ice density ($910\,\mathrm{kg\,m^{-3}}$) and $\mathbf{g}$ is gravitational acceleration ($9.81\,\mathrm{m\,s^{-2}}$, where s represents a second). The stress tensor is split into a deviatoric part and an isotropic pressure p, i.e., $\boldsymbol{\sigma}_\mathbf{u} = \boldsymbol{\tau}_\mathbf{u} - p\mathbf{I}$. Denoting the strain rate tensor of the velocity field by

$$\dot{\boldsymbol{\epsilon}}_\mathbf{u} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^\top), \tag{3}$$

the Glen's law of rheology [41] links the deviatoric stress to the strain rate via the constitutive equation

$$\boldsymbol{\tau}_\mathbf{u} = 2\eta(\mathbf{u})\dot{\boldsymbol{\epsilon}}_\mathbf{u} \quad \text{with} \quad \eta(\mathbf{u}) = \frac{1}{2}A^{-\frac{1}{n}}\left(\frac{1}{2}\dot{\boldsymbol{\epsilon}}_\mathbf{u} : \dot{\boldsymbol{\epsilon}}_\mathbf{u}\right)^{\frac{1-n}{2n}}, \tag{4}$$

where $\eta(\mathbf{u})$ is the effective viscosity nonlinearly depending on the strain rate and the double-dot product is the Frobenius inner product defined as

$$\boldsymbol{\sigma}_1 : \boldsymbol{\sigma}_2 = \mathrm{trace}(\boldsymbol{\sigma}_1^\top \boldsymbol{\sigma}_2) = \sum_{i,j} (\boldsymbol{\sigma}_1)_{ij}(\boldsymbol{\sigma}_2)_{ij},$$

for two second order tensors $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$. Here $A = 10^{-16}(\mathrm{Pa^{-n}a^{-1}})$ is the flow parameter and we often set $n = 3$. Using the nonlinear constitutive equation (4), the conservation of momentum (1) becomes

$$-\nabla \cdot \boldsymbol{\tau}_\mathbf{u} + \nabla p = \rho \mathbf{g}. \tag{5}$$

Without loss of generality, we assume the domain Ω is bounded by two disjoint surfaces, Γ_b the bottom boundary and $\Gamma_\mathrm{t} = \partial\Omega \setminus \Gamma_\mathrm{b}$ the top boundary. Let $\mathbf{n}$ denote the unit outward normal vector at any point on the boundary $\partial\Omega$. On the top boundary we impose the traction-free boundary condition

$$\boldsymbol{\tau}_\mathbf{u}\mathbf{n} - p_0\mathbf{n} = 0 \quad \text{on} \quad \Gamma_\mathrm{t}, \tag{6}$$

where p_0 is the atmospheric pressure. Since the atmospheric pressure is often assumed to be negligible [43], so that we adopt $p_0 = 0$ here. On the bottom boundary we impose a no-penetration condition along the outward normal direction, i.e.,

$$\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on} \quad \Gamma_\mathrm{b}, \tag{7}$$

and a sliding boundary condition along the tangential direction. Given the map to the tangential direction $\mathbf{T} = \mathbf{I} - \mathbf{n} \otimes \mathbf{n}$, the sliding boundary condition is given as

$$\mathbf{T}\big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n} - p\mathbf{n}\big) + \beta\mathbf{T}\mathbf{u} = 0 \quad \text{on} \quad \Gamma_{\mathrm{b}},$$

for some basal drag coefficient $\beta > 0$. Note that the basal drag coefficient β is a function of location in general. Since $\mathbf{T}\mathbf{n} = 0$, the sliding boundary condition does not depend on the pressure, which can be reduced to

$$\mathbf{T}\big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n}\big) + \beta\mathbf{T}\mathbf{u} = 0 \quad \text{on} \quad \Gamma_{\mathrm{b}}. \tag{8}$$

Along the tangential direction, there is a special case that $\beta \to \infty$, where the sliding boundary condition becomes a non-sliding Dirichlet boundary condition, i.e., $\mathbf{T}\mathbf{u} = 0$. In later numerical examples, our algorithm can be adapted to both cases.

To summarize, the ice flow with nonlinear rheology can be modelled by a system of equations

$$-\nabla \cdot \boldsymbol{\tau}_{\mathbf{u}} + \nabla p = \mathbf{f} \quad \text{on} \quad \Omega \tag{9}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{on} \quad \Omega \tag{10}$$

$$\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n} - p_0\mathbf{n} = 0 \quad \text{on} \quad \Gamma_{\mathrm{t}} \tag{11}$$

$$\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on} \quad \Gamma_{\mathrm{b}} \tag{12}$$

$$\mathbf{T}\big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n}\big) + \beta\mathbf{T}\mathbf{u} = 0 \quad \text{on} \quad \Gamma_{\mathrm{b}}, \tag{13}$$

where $\beta > 0$, $\mathbf{f} = \rho\mathbf{g}$, $\mathbf{T} = \mathbf{I} - \mathbf{n} \otimes \mathbf{n}$, and $\boldsymbol{\tau}_{\mathbf{u}}$ given in (4). With $\beta \to \infty$, the mixed boundary conditions in (12) and (13) simply become $\mathbf{u} = 0$ on Γ_{b}.

2.2. Variational formulation

The momentum equation (9) and the divergence-free mass equation (10) have the variational form: finding a vector function $\mathbf{u} \in H^1(\Omega)$ satisfying boundary conditions (11) and (12) and a scalar function $p \in L^2(\Omega)$ satisfying $p = p_0$ on Γ_{t} such that

$$\int_{\Omega} \big(-\nabla \cdot \boldsymbol{\tau}_{\mathbf{u}} + \nabla p - \mathbf{f} \big) \cdot \mathbf{v} dV = 0 \tag{14}$$

$$-\int_{\Omega} q \, \nabla \cdot \mathbf{u} dV = 0, \tag{15}$$

for all vector functions $\mathbf{v} \in H^1(\Omega)$ satisfying $\mathbf{v} \cdot \mathbf{n} = 0$ on Γ_{b} and $\boldsymbol{\tau}_{\mathbf{v}}\mathbf{n} - p_0\mathbf{n} = 0$ on Γ_{t} and all scalar function $q \in L^2(\Omega)$ satisfying $q = p_0$ on Γ_{t}. Applying the divergence theorem for vectors and tensors, the momentum equation (14) can also be expressed as

$$\int_{\Omega} \left(\boldsymbol{\tau}_{\mathbf{u}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{v}} - p\nabla \cdot \mathbf{v} - \mathbf{f} \cdot \mathbf{v} \right) dV - \oint_{\partial\Omega} \big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n} - p\mathbf{n} \big) \cdot \mathbf{v} dS = 0. \tag{16}$$

The solutions $(\mathbf{u}, p)$ satisfying the above variational form can be expressed as the minimizer of some energy functional subject to the divergence-free constraint (15) and some boundary conditions (11)-(13). This provides a starting point for applying deep neural network approximation to model the glacier flow. We present the energy functional and the corresponding optimization problem for the sliding and the the non-sliding boundary conditions as follows.

Sliding bottom boundary. Subject to the traction free boundary condition (11) on the top boundary, the boundary integral at the top boundary is eliminated, i.e.,

$$\oint_{\Gamma_{\mathrm{t}}} \big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n} - p\mathbf{n}\big) \cdot \mathbf{v} dS = \oint_{\Gamma_{\mathrm{t}}} \big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n} - p_0\mathbf{n}\big) \cdot \mathbf{v}\, dS = 0. \tag{17}$$

where p_0 is the atmospheric pressure at the top boundary. At the bottom boundary, applying the condition $\mathbf{v} \cdot \mathbf{n} = 0$ on Γ_{b} and the sliding boundary condition (13) we have

$$\begin{aligned}
\oint_{\Gamma_{\mathrm{b}}} \big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n} - p\mathbf{n}\big) \cdot \mathbf{v} dS &= \oint_{\Gamma_{\mathrm{b}}} \big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n}\big) \cdot \mathbf{v}\, dS \\
&= \oint_{\Gamma_{\mathrm{b}}} \Big(\mathbf{T}\big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n}\big)\Big) \cdot \Big(\mathbf{T}\mathbf{v}\Big)\, dS + \oint_{\Gamma_{\mathrm{b}}} \Big((\mathbf{I}-\mathbf{T})\big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n}\big)\Big) \cdot \Big((\mathbf{I}-\mathbf{T})\mathbf{v}\Big)\, dS \\
&= -\oint_{\Gamma_{\mathrm{b}}} \beta\big(\mathbf{T}\mathbf{u}\big) \cdot \big(\mathbf{T}\mathbf{v}\big)\, dS,
\end{aligned} \tag{18}$$

as range($\mathbf{T}$) and range($\mathbf{I} - \mathbf{T}$) are mutually orthogonal and $(\mathbf{I} - \mathbf{T})\mathbf{v} = 0$. Therefore, the boundary integral in (16) can be simplified as

$$-\oint_{\partial\Omega} \big(\boldsymbol{\tau}_{\mathbf{u}}\mathbf{n} - p\mathbf{n}\big) \cdot \mathbf{v} dS = \oint_{\Gamma_{\mathrm{b}}} \beta\big(\mathbf{T}\mathbf{u}\big) \cdot \big(\mathbf{T}\mathbf{v}\big)\, dS.$$

As shown in [12], the solutions $(\mathbf{u}, p)$ satisfying the variational form (16) can be considered as the minimizer of the the energy functional

$$\mathcal{E}_{\mathrm{s}}(\mathbf{u}) = \int_\Omega \left(\frac{2n}{1+n} A^{-\frac{1}{n}} \left(\frac{1}{2}\dot{\boldsymbol{\epsilon}}_{\mathbf{u}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{u}} \right)^{\frac{1+n}{2n}} - \rho\mathbf{g} \cdot \mathbf{u} \right) dV + \frac{1}{2} \oint_{\Gamma_{\mathrm{b}}} \beta\big(\mathbf{T}\mathbf{u}\big) \cdot \big(\mathbf{T}\mathbf{u}\big)\, dS, \tag{19}$$

subject to the divergence-free condition (10). Formulating this constrained optimization problem using the method of Lagrange multiplier, we have the Lagrangian functional

$$\mathcal{L}_{\mathrm{s}}(\mathbf{u}, p) = \mathcal{E}_{\mathrm{s}}(\mathbf{u}) - \int_\Omega p \nabla \cdot \mathbf{u} dV, \tag{20}$$

in which the pressure function p plays the role of the Lagrange multiplier. Since the directional derivative of $f(\mathbf{u}) := \frac{1}{2}\dot{\boldsymbol{\epsilon}}_{\mathbf{u}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{u}}$ along a function $\mathbf{v}$ is $f'(\mathbf{u})[\mathbf{v}] = \dot{\boldsymbol{\epsilon}}_{\mathbf{u}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{v}}$, the variation of $\mathcal{L}(\mathbf{u}, p)$ along functions $(\mathbf{v}, \mathbf{q})$ takes the form

$$\mathcal{L}'_{\mathrm{s}}(\mathbf{u}, p)[\mathbf{v}, q] = \int_\Omega \left(\boldsymbol{\tau}_{\mathbf{u}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{v}} - p\nabla \cdot \mathbf{v} - \rho\mathbf{g} \cdot \mathbf{v}\right) dV - \int_\Omega q\, \nabla \cdot \mathbf{u} dV + \oint_{\Gamma_{\mathrm{b}}} \beta\big(\mathbf{T}\mathbf{u}\big) \cdot \big(\mathbf{T}\mathbf{v}\big)\, dS.$$

The solution to the Lagrange multiplier, $(\mathbf{u}, p)$ such that $\mathcal{L}'_{\mathrm{s}}(\mathbf{u}, p)[\mathbf{v}, q] = 0$ for all $(\mathbf{v}, q)$ is equivalent to the solution to the variational form defined in (14) and (15). Thus, the velocity field $\mathbf{u}$ of the ice flow model can be obtained by minimizing the energy functional $\mathcal{E}_{\mathrm{s}}(\mathbf{u})$ subject to the divergence-free constraint, the traction-free boundary condition (11), and the Dirichlet boundary conditions (12) and (13).

Non-sliding bottom boundary. With non-sliding boundary condition at the bottom, i.e., $\mathbf{u} = 0$ on Γ_{b}, we do not need to impose the boundary condition (13) using the boundary integral as the sliding boundary case in (19). Following a similar derivation as above, the velocity field $\mathbf{u}$ of the ice flow model with non-sliding boundary can be obtained by minimizing the energy functional

$$\mathcal{E}_{\mathrm{ns}}(\mathbf{u}) = \int_{\Omega} \left(\frac{2n}{1+n} A^{-\frac{1}{n}} \left(\frac{1}{2} \dot{\boldsymbol{\epsilon}}_{\mathbf{u}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{u}} \right)^{\frac{1+n}{2n}} - \rho \mathbf{g} \cdot \mathbf{u} \right) dV, \tag{21}$$

subject to the divergence-free constraint, the traction-free boundary condition (11), and the Dirichlet boundary conditions $\mathbf{u} = 0$ on Γ_{b}. The corresponding Lagrangian functional becomes

$$\mathcal{L}_{\mathrm{ns}}(\mathbf{u}, p) = \mathcal{E}_{\mathrm{ns}}(\mathbf{u}) - \int_{\Omega} p \nabla \cdot \mathbf{u} dV. \tag{22}$$

2.3. Divergence-free solutions spaces

Instead of searching for the saddle point solution of the Lagrangian functionals, we can construct divergence-free solution spaces for the velocity field $\mathbf{u}$ to reduce the constrained optimization problems to unconstrained optimization problems. In the two dimensional case, i.e., $(x, y) \in \Omega \subset \mathbb{R}^2$, we can define a potential function $\phi : \Omega \mapsto \mathbb{R}$, which leads to a vector function

$$\mathbf{u} = \left[\frac{\partial \phi}{\partial y}, -\frac{\partial \phi}{\partial x} \right]^{\top},$$

that satisfies the divergence-free condition by construction. In the three dimensional case, i.e., $(x, y, z) \in \Omega \subset \mathbb{R}^3$, we can define a vector function $\boldsymbol{\phi} : \Omega \mapsto \mathbb{R}^3$, so that

$$\mathbf{u} = \nabla \times \boldsymbol{\phi}$$

satisfies the divergence-free condition. Using the divergence-free construction of the velocity field, we can directly minimize the energy functional $\mathcal{E}_{\mathrm{s}}(\mathbf{u})$ and $\mathcal{E}_{\mathrm{ns}}(\mathbf{u})$ subject to appropriate boundary conditions without using the Lagrangian formulation.

3. Formulation of the DNN method

We first discuss the background of DNNs and then develop the DNN-based methods for solving the Non-Newtonian ice flow problems.

3.1. DNNs and its approximation property

There are two ingredients in defining a DNN. The first one is a linear map of the form $T : R^n \to R^m$, defined as $T(x) = Ax + b$, where $A = (a_{ij}) \in R^{m\times n}$, $x \in R^n$ and $b \in R^m$. The second one is a nonlinear activation function $\sigma : R \to R$. Common examples of the activation function include the rectified linear unit (ReLU), $\sigma(x) = \max(0, x)$, the soft-plus function, $\sigma(x) = \log(e^x + 1)$, the sigmoid function, $\sigma(x) = (1 + e^{-x})^{-1}$, etc. The definition of activation function can be trivially extended to a nonlinear map $\sigma : R^n \to R^n$ by applying the scalar-valued activation function element-wise to each of its inputs.

Using above definitions, we are able to define a continuous function $F(x)$ by a composition of linear transforms and activation maps, i.e.,

$$F(x) = T^k \circ \sigma \circ T^{k-1} \circ \sigma \cdots \circ T^1 \circ \sigma \circ T^0(x), \tag{23}$$

where $T^i(x) = A_i x + b_i$ with A_i and b_i are unknown matrices and vectors to be estimated. Dimensions of A_i and b_i are chosen to make (23) meaningful. Such a DNN is called a $(k+1)$-layer DNN, which has k hidden layers. Collecting all the unknown coefficients (e.g., A_i and b_i) in (23) as $\theta \in \Theta$, where θ is a high-dimensional vector and Θ is the space of θ. The DNN representation of a continuous function can be viewed as

$$F = F(x; \theta). \tag{24}$$

We use $\mathbb{F} = \{F(\cdot, \theta) | \theta \in \Theta\}$ to denote the set of all expressible functions by the DNN parameterized by $\theta \in \Theta$. Then $\mathbb{F}$ provides an efficient way to represent unknown continuous functions, comparing with a linear solution space used in classic numerical methods, e.g., a trial space spaced by linear nodal basis functions in the FEM. In the sequel, we shall discuss the approximation property of the DNN, which is relevant to the study of the expressive power of a DNN model [10, 53].

Early investigations of the approximation property of neural networks can be found in [11, 22], where the authors analyzed the approximation property for the function class given by a feed-forward neural network with a single hidden layer. Later, many authors analyzed the error estimates of such neural networks in terms of the number of neurons, the number of network layers, and the choice of activation function; see [14, 46] for a throughout review of relevant works.

In recent years, DNNs have shown successful applications in a broad range of problems, including classification of complex systems and construction of response surfaces for high-dimensional models. Significant efforts have been devoted to investigate the dependence of the expressive power of DNNs on the network configuration. For example, in [10], the authors proved that convolutional DNNs were able to express multivariate functions given in so-called hierarchical tensor formats. In [62], the author studied the expressive power of shallow and deep neural networks with piece-wise linear activation functions and established

new rigorous upper and lower bounds for the network complexity in approximating Sobolev spaces. In [19], the authors proved that the continuous piece-wise bilinear finite element space is embedded in the space of DNNs constructed from the ReLU activation function, a sufficient width and a sufficient depth. Thus, we can uses DNNs to approximate the solution space spanned by the FEM basis.

3.2. Formulation of the variational neural network approach

The model problem of the ice flow with nonlinear rheology (13) can be solved by using numerical methods such as FEMs and FDMs. However, the nonlinearity in the problem bring essential difficulty in solving (13) by the traditional numerical methods. Inspired by the recent development of deep learning based numerical method for solving variational problems [13, 59], we will develop a variational DNN method to solve the glacier model problem defined in (13).

According to the divergence free property, i.e. $\nabla \cdot \mathbf{u} = 0$, we can represent the velocity field $\mathbf{u}$ as follows:

$$\left[\frac{\partial \phi^0}{\partial y}, -\frac{\partial \phi^0}{\partial x}\right]^\top \text{ in 2D} \quad \text{and} \quad \nabla \times \begin{pmatrix} \phi^1 \\ \phi^2 \\ \phi^3 \end{pmatrix} \text{ in 3D}, \tag{25}$$

where ϕ^i is a scalar-valued function approximated by $F^i(\mathbf{x}, \theta)$. Here, $F^i(\mathbf{x}, \theta)$ is a DNN representation with $\mathbf{x} \in \mathbb{R}^d$ as its input and scalar output defined in (23). Moreover, θ denotes all the parameters that will be determined during the training stage. Denoting the DNN representation of the velocity field by $\tilde{\mathbf{u}}$, the numerical solution of (13) can be obtained by finding $\theta \in \Theta$ that minimizes the energy functional $\mathcal{E}_{\mathrm{s}}(\tilde{\mathbf{u}})$ subject to various boundary conditions.

Ideally, we want to define $\tilde{\mathbf{u}}$ using $\theta \in \Theta_b \subset \Theta$, where Θ_b is the maximal parameter subset such that the resulting $\tilde{\mathbf{u}}$ satisfies the traction-free boundary condition (11), and the Dirichlet boundary conditions (12) and sliding bottom boundary condition (13). After parameterizing the expressible function space by $\theta \in \Theta_b$, we equivalently define the variational problem (19) as

$$\min_{\theta \in \Theta_b} J(\theta) = \mathcal{E}_{\mathrm{s}}(\tilde{\mathbf{u}}) = \int_\Omega \left(\frac{2n}{1+n} A^{-\frac{1}{n}} \left(\frac{1}{2} \dot{\boldsymbol{\epsilon}}_{\tilde{\mathbf{u}}} : \dot{\boldsymbol{\epsilon}}_{\tilde{\mathbf{u}}} \right)^{\frac{1+n}{2n}} - \rho \mathbf{g} \cdot \tilde{\mathbf{u}} \right) dV + \frac{1}{2} \oint_{\Gamma_{\mathrm{b}}} \beta\big(\mathbf{T}\tilde{\mathbf{u}}\big) \cdot \big(\mathbf{T}\tilde{\mathbf{u}}\big)\, dS. \tag{26}$$

Note that the divergence-free condition in (15) is automatically satisfied by the representation (25), and thus no additional treatment is needed.

The variational problems (26) is not convex in general and the integrals in (26) do not have a closed-form expression. Thus we numerically approximate the integrals by the Monte Carlo method and use the stochastic gradient descent (SGD) method [4] to minimize the

objective function after Monte Carlo approximation. Denoting θ_k the kth component of the high-dimensional vector θ, the derivative of $J(\theta)$ with respect to θ_k can be approximated as

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{\mathrm{vol}(\Omega)}{N} \sum_{i=1}^{N} \partial_{\theta_k} \left(\frac{2n}{1+n} A^{-\frac{1}{n}} \left(\frac{1}{2} \dot{\boldsymbol{\epsilon}}_{\tilde{\mathbf{u}}} : \dot{\boldsymbol{\epsilon}}_{\tilde{\mathbf{u}}} \right)^{\frac{1+n}{2n}} (x_i) - \rho \mathbf{g} \cdot \tilde{\mathbf{u}}(x_i) \right) + \frac{\mathrm{area}(\Gamma_b)}{N_b} \sum_{j=1}^{N_b} \partial_{\theta_k} \Big(\frac{1}{2} \beta \big(\mathbf{T}\tilde{\mathbf{u}} \big) \cdot \big(\mathbf{T}\tilde{\mathbf{u}} \big)(y_j) \Big), \tag{27}$$

where random samples $x_i \overset{i.i.d.}{\sim} \mathrm{Unif}(\Omega)$ are uniformed drawn from the physical domain Ω, random samples $y_j \overset{i.i.d.}{\sim} \mathrm{Unif}(\Gamma_b)$ are uniformed drawn from the bottom boundary Γ_b, $\mathrm{vol}(\Omega)$ is the volume of the domain and $\mathrm{area}(\Gamma_b)$ is the area of bottom boundary. In the context of deep learning method, N and N_b are called batch numbers, which mean the number of training examples utilized in one iteration.

After approximating the gradient of $J(\theta)$, we can update each component of θ as follows:

$$\theta_k^{n+1} = \theta_k^n - \eta \frac{\partial J(\theta)}{\partial \theta_k} |_{\theta_k = \theta_k^n}, \tag{28}$$

where η is the learning rate. To accelerate the training of the neural network, we use the Adam version of the SGD method [28].

In the objective function (26), it is rather challenging to restrict the neural network parameter θ to the subset Θ_b that satisfies the boundary conditions, because the boundary of the subset may have complicated geometry. To address this issue, we adopt a relaxation approach by imposing boundary conditions as penalty terms. For each of the boundary conditions in (11)-(13), we define a linear map $\mathcal{B}_j$ that maps the solution $\tilde{u}$ to the residual of the boundary constraint, which defines a boundary integral

$$B_j(\theta) = \int_{\partial D_j} \left\| \mathcal{B}_j \tilde{u}(x, \theta) \right\|^2 dx$$

for the boundary ∂D_j. Then, the boundary conditions can be imposed as soft constraints via penalty terms to the objective functional $J(\cdot)$ in (26). This leads to a new objective functional

$$\tilde{u}_\varepsilon = \arg\min_{\theta \in \Theta} \Big(J(\theta) + \sum_{j=1}^{b} \frac{1}{\varepsilon_j} B_j(\theta) \Big), \tag{29}$$

where b is the number of boundary conditions needs to be imposed. Note that when a penalty term $\varepsilon_j^{-1} B_j(\theta)$ approaches zero as $\varepsilon_j \to 0$, the corresponding boundary condition is satisfied in the weak sense.

3.3. Implementation details

In addition to the conventional DNN formulation for solving PDE, we also made some modification that are shown to be essential in resolving the scale of the problem domain and the scale of physical parameters in real world problems.

Normalizing Layer. In some glacier model problems, periodic boundary conditions are used. For example, periodic boundary conditions are imposed in the two horizontal directions in the in the 2D model presented in Example C of the benchmarks [43]. To this end, we add an extra layer between input and first dense layer of our network. That maps (x, y) to terms like $\big(\cos(2\pi x/T), \sin(2\pi x/T), \cos(2\pi y/T), \sin(2\pi y/T)\big)$, where T is the period in the horizontal dimensions.

In most of the glacier modelling problems, the vertical scaling can be two or three orders of magnitude smaller than the horizontal scaling. To resolve this scaling, we also introduce a reparametrization: $x \mapsto x/L$, where L is the diameter of the domain in corresponding dimension, to the normalization layer from input. Resolving this scaling is essential to ensure the expressibility of neural networks for the problems we are targeting here, because nonlinearity can be represented by a neural network may only exist in a bounded input domain. Taking a scalar single layer perception with the Sigmoid activation function $\sigma(z) = (1 + e^{-z})^{-1}$ as an example, we have

$$f(x) = W_2\sigma(W_1 x + b_1) + b_2.$$

When $W_1 x + b_1$ is sufficiently large (e.g., $W_1 x + b_1 \gg 5$), $f(x)$ becomes a constant approximately. Thus, to sufficiently express nonlinear functions, scale of W_1 has to be $\mathcal{O}(L^{-1})$ without any reparametrization. However, most of existing implementations of machine learning toolboxes only support single-precision float-point numbers. This leads to inaccurate representation of W_1 and the gradient of the objective function, as the length scale of most of the real-world problems can be $O(10^4) - O(10^6)$. The reparametrization is critical to ensure the expressibility of neural networks and the numerical stability in these real-world problems.

Balancing penalty parameters of boundary conditions. Except for periodic boundary conditions, as shown in (29), the other boundary conditions are enforced by penalty terms. However, there is no general framework to choose the penalty weight ε_j in the context of DNN. It may not be suitable to use the same penalty parameter ε_j for different boundary conditions, as the magnitude of the boundary maps $B_j(\theta)$ can vary significantly. For example, in the 3D model presented in Section 4.3, the term $B_j(\theta)$ is $O(10^{-4})$ for the traction-free condition at the top boundary and $O(10^6)$ for the basal friction condition at the bottom boundary. Thus, finding the correct penalty parameter ε_j for each of the boundary conditions is critical in balancing the soft constraints due to various boundary conditions. We propose to first estimate the scaling relations between each of the boundary integrals $B_j(\cdot)$ with the objective functional $J(\cdot)$ at the initialization stage of the network training. Denoting the initial parameter by θ_0, we can then assign ε_j as

$$\varepsilon_j = \frac{1}{\varepsilon_0}\frac{B_j(\theta_0)}{J(\theta_0)},$$

so that the objective functional $J(\cdot)$ and each of the boundary integrals $B_j(\cdot)$ can be approximately balanced during the training. Here the common factor ε_0 is chosen to be 50, which is an empirical constant shown to be sufficient in previous research [25] and in our numerical experiments on simpler nonlinear models.

Network Hyperparameters. In the numerical experiments of this work, we will apply the same network to represent ϕ in both two and three dimensions. After the normalizing layer, it has six latent dense layers with width 10. Between these dense layers we apply the Sigmoid activation function to guarantee the smoothness of our representation. There is no activation function between the second last layer and the output layer. We illustrate the structure of our network in Fig. 1.

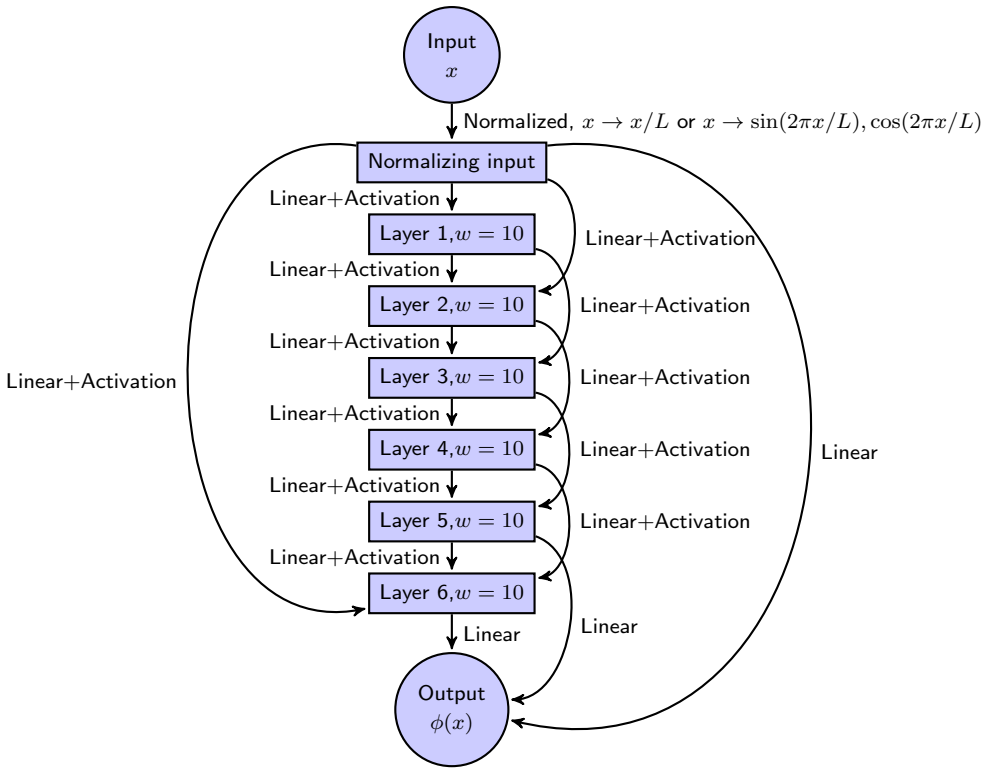


Figure 1: The network Layout of ϕ.

4. Numerical Results

In this section, we shall present numerical results in solving non-Newtonian Stokes equations to demonstrate the performance of the propose method. First we consider a two dimensional synthetic model with an analytical solution to demonstrate the efficiency and accuracy of our method. Then we apply the developed method on realistic benchmark prob-

lems [43] to demonstrate its generality. This includes a two dimensional model of the Arolla glacier and a three dimensional box model.

4.1. A 2D Model on irregular domain

We start with a two dimensional model defined on a domain that is enclosed by

$$y = 0 \quad \text{and} \quad y = \frac{1}{2}x\,(1-x) \quad \text{with} \quad x \in [0,1].$$

To setup the benchmark, we start with a ground truth potential function

$$\phi = \exp(x)(x-2)^2 y(y-1)^2,$$

which leads to an analytical expression of the velocity field

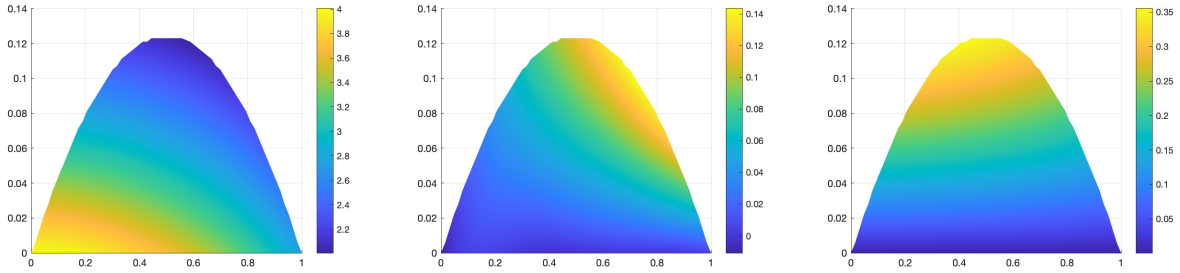
$$\begin{cases} u = \exp(x)(x-2)^2(1-4y+3y^3) \\ v = -\exp(x)(x-2)^2 y(y-1)^2 \end{cases}.$$

The above velocity field satisfies the conservation of mass condition in Eq. (10) and the no-penetration boundary condition in Eq. (12) at the bottom boundary Γ_{b}. Then, by substituting this velocity field into Eq. (9), we obtain an explicit expression of the forcing term $\mathbf{f}$ in Eq. (9), a function p_0 for the traction-free boundary condition Eq. (11) at the top boundary Γ_{t}, and a function β for the sliding boundary condition Eq. (13) at the bottom boundary Γ_{b}.

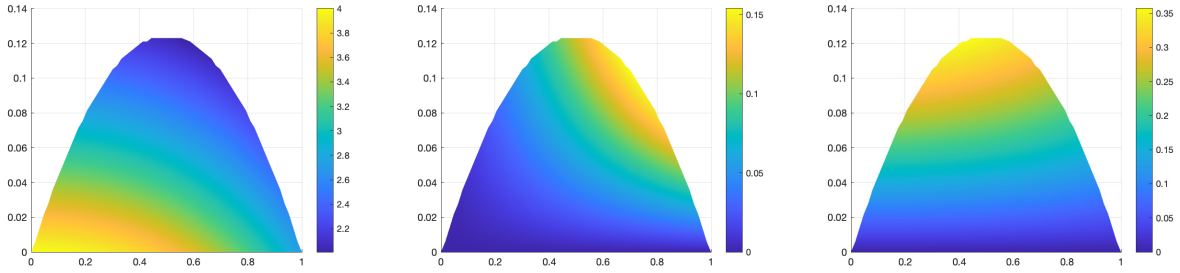
Since the shape of the domain is a semicircle in this example, it is not necessary to use normalization layers or balance regularization factors. We use a constant regularization factor $\frac{1}{\varepsilon} = 50$ for all boundary conditions. During training, in every step of stochastic gradient descent, we use 5000 random samples uniformly distributed in the domain Ω to evaluate Eq. (27) and 1000 uniform random samples on the boundary Γ to construct the soft constraints in Eq. (11)–Eq. (13). The learning rate, which is the step length of SGD, is set to be 10^{-3}. We renew the training data every 200 steps of learning. The configuration of the neural network is illustrated in Fig. 1.

In Fig. 2, we compare the results of our proposed method with the ground truth. Since the velocity field is invariant to the potential function ϕ up to a constant shift, we shift the output of ϕ learned by neural networks to ensure it has the same average as the ground truth in the comparison. For both of the velocity field and the potential function, the neural network offers comparable results with the ground truth.

In Fig. 3 we compare the relative L_2 error of the velocity field with the energy functional $\mathcal{E}_{\mathrm{s}}(\mathbf{u})$ in Eq. (19). We observe that $\mathcal{E}_{\mathrm{s}}(\mathbf{u})$ follows a similar trend to the relative L_2 error. The relative L_2 error drops to 0.005 after 10000 steps of training.



(a) u (b) v (c) ϕ rescaled



(d) ref: u (e) ref v (f) ref ϕ

Figure 2: Solution of synthetic model

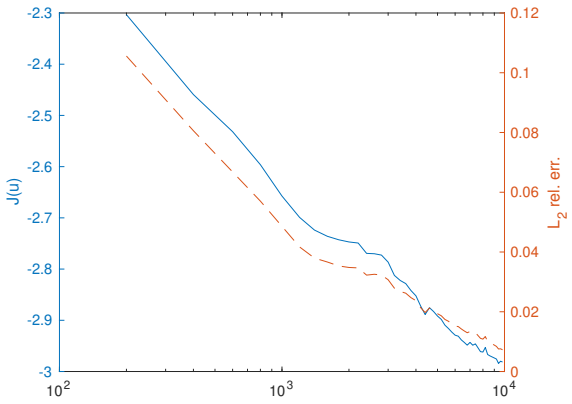
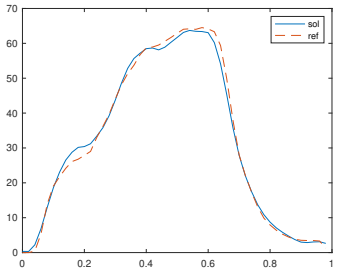
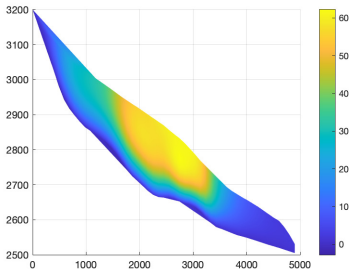


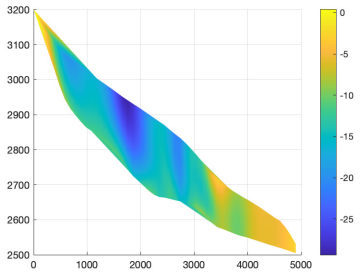
Figure 3: Training loss (Lagrangian) and validation error (L_2 relative error) with respect to training steps



(a) Surface Velocity along surface direction

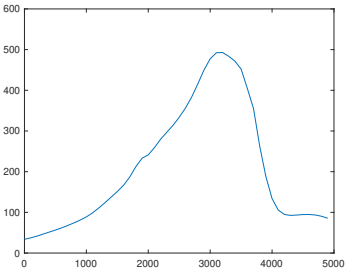


(b) u

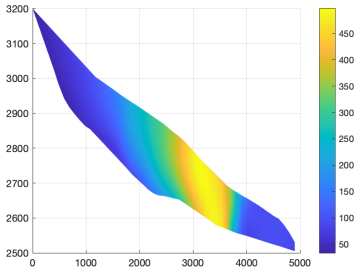


(c) v

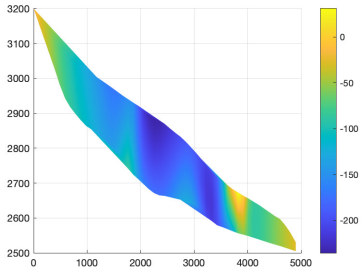
Figure 4: Arolla



(a) Surface Velocity along surface direction



(b) u



(c) v

Figure 5: Noemi's paper

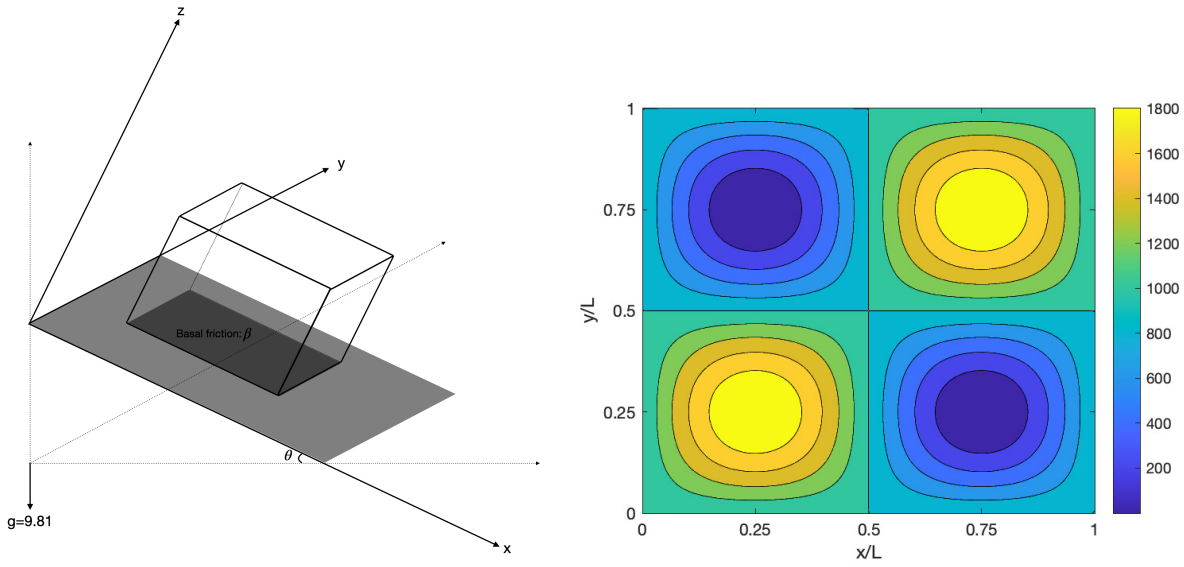
4.2. The Arolla model

The two-dimensional model of the Arolla glacier is a diagnostic experiment along the central flow-line of a temperate glacier in the European Alps (Haut Glacier d'Arolla, Switzerland), based on earlier experiments by [3, 42]. The domain of this model is enclosed by the longitudinal surface and bedrock profiles of Haut Glacier d'Arolla (as shown in Fig. 4 (b) and (c)). This model has been used as a benchmark example in various investigations of the forward modelling and the inverse modelling of glaciers, see [44] for some notable examples.

We consider two experiments in this example. The first experiment follows the setup of [43, Section 3.5], in which a no-sliding bottom boundary with $\beta = \infty$ is considered. In the second experiment, we use a sliding bottom boundary with the function

$$\exp(\beta) = \begin{cases} 1000 + 1000\sin(\frac{2\pi x}{5000}) & 0 \le x < 3750 \\ 1000(16 - \frac{x}{250}) & 3750 \le x < 4000 \\ 1000 & 4000 \le x < 5000 \end{cases}$$

which is the same as the work of [44, Section 4.2].



(a) Illustration of coordinate change

(b) Value of Basal Coefficients β

Figure 6: Setting of box Model

4.3. A 3D box model

We also consider the three dimensional box model presented in [43, Section 3]. In this model, a slab of ice sliding down a sloping bed with a constant incline angle $\alpha = 0.1$ degree is considered. The ice slab is enclosed in a box-shaped domain $[0, L] \times [0, L] \times [0, H]$, where $L = 20 \times 10^3$ metres, and $H = 10^3$ meters. After a change of coordinate that align the x-axis with the sliding surface (see 6(a)), the vector $\rho\mathbf{g}$ is given as

$$\rho\mathbf{g} = 910 \times 9.81 \times [\sin\theta, 0, -\cos\theta]\,\text{Pa},$$

where $\theta = \pi/1800$. In this example, periodic boundary conditions are applied to boundaries in the horizontal directions, the traction-free boundary conditions in Eq. (11) is applied to the top boundary, and the sliding boundary condition in Eq. (12) and Eq. (13) is prescribed at the bottom boundary. The basal sliding coefficient is defined as

$$\beta(x, y) = 1000 + 1000 \sin\left(\frac{2\pi x}{L}\right) \sin\left(\frac{2\pi y}{L}\right).$$

In Fig.6(b) we plot the value of basal friction coefficient β. Empirically speaking, the surface speed becomes larger when β is small. The ground truth solution is unknown. In Fig.7 we plot the surface velocity against y direction at the slice $x = \frac{L}{4}$. In addition we compare our prediction with various result in [43] (experiment C, $L = 20$km).

5. Conclusions

In this paper, we investigate deep-learning methods to simulate non-Newtonian ice flow models. By formulating the model problem into a variational form and constructing a

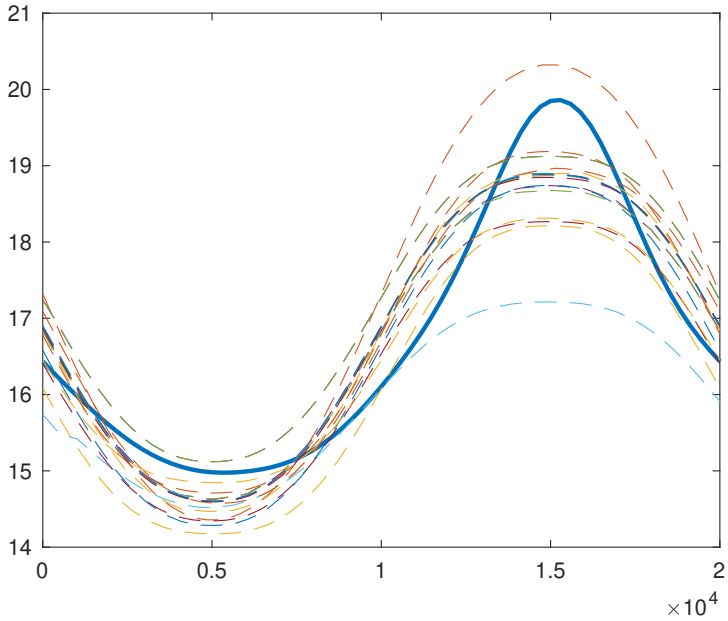


Figure 7: Surface velocity, Benchmark paper, experiment C, 20km. Solid line: Our prediction; Dashed lines: various prediction in Benchmark paper

divergence-free solution space using DNN, we convert the solution of the nonlinear PDE into the solution of an optimization problem. Despite the rather high-dimensionality of the parameter space of the DNN, the solution can still be efficiently obtained by the SGD method. In this framework, once we are able to sample from the problem domain and its boundary, no special treatment is needed to handle irregular boundaries of the problem domain. Therefore, the proposed method is easy to implement and mesh-free. To address the application in real-world computation, we introduce normalizing layers and adaptive boundary penalties in the configuration of our network. Finally, we present numerical experiments to demonstrate the performance of the proposed method. Specifically, we use the DNN method to solve ice flow model for the 2D Arrola Glacier and a 3D box model with real-world scaling. Our numerical experiments demonstrate that the DNN method provides satisfactory simulation results for glacier flows. We expect this method can be applied to a more general class of problems in non-Newtonian mechanics that satisfies certain variational principles.

Acknowledgements

The research of T. Cui is supported by the Australian Research Council under the grant DP21010309. The research of Z. Zhang is supported by Hong Kong RGC grant (Projects 17300318 and 17307921), National Natural Science Foundation of China (Project 12171406), Seed Funding Programme for Basic Research (HKU) and Seed Funding for Strategic Interdisciplinary Research Scheme 2021/22 (HKU).

References

[1] G. Bao, X. Ye, Y. Zang, and H. Zhou, Numerical solution of inverse problems by weak adversarial networks, Inverse Problems, 36(11) (2020), p. 115003.

[2] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. Brenner, Learning data-driven discretizations of PDEs, Bulletin of the American Physical Society, 63 (2018).

[3] H. Blatter, G. K. Clarke, and J. Colinge, Stress and velocity fields in glaciers: Part ii. sliding and basal stress distribution, Journal of Glaciology, 44 (1998), pp. 457–466.

[4] L. Bottou, Large-scale machine learning with stochastic gradient descent, in Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[5] J. Brown, Efficient nonlinear solvers for nodal high-order finite elements in 3d, Journal of Scientific Computing, 45 (2010), pp. 48–63.

[6] J. Brown, B. Smith, and A. Ahmadia, Achieving textbook multigrid efficiency for hydrostatic ice sheet flow, SIAM Journal on Scientific Computing, 35 (2013), pp. B359–B375.

[7] Z. Cai, J. Chen, and M. Liu, Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation, Journal of Computational Physics, (2021), p. 110514.

[8] Z. Cai, J. Chen, M. Liu, and X. Liu, Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs, Journal of Computational Physics, 420 (2020), p. 109707.

[9] X. Chen, P. Rosakis, Z. Wu, and Z. Zhang, A deep learning approach to nonconvex energy minimization for martensitic phase transitions, arXiv:2206.13937, (2022).

[10] N. Cohen, O. Sharir, and A. Shashua, On the expressive power of deep learning: A tensor analysis, in Conference on Learning Theory, 2016, pp. 698–728.

[11] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems, 2 (1989), pp. 303–314.

[12] J. K. Dukowicz, S. F. Price, and W. H. Lipscomb, Consistent approximations and boundary conditions for ice-sheet dynamics from a principle of least action, Journal of Glaciology, 56 (2010), pp. 480–496.

[13] W. E and B. Yu, The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.

[14] S. Ellacott, Aspects of the numerical analysis of neural networks, Acta Numerica, 3 (1994), pp. 145–202.

[15] H. C. Elman, D. J. Silvester, and A. J. Wathen, Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics, Numerical Mathematics and Scie, 2014.

[16] M. R. Fraters, W. Bangerth, C. Thieulot, A. Glerum, and W. Spakman, Efficient and practical newton solvers for non-linear stokes systems in geodynamic problems, Geophysical Journal International, 218 (2019), pp. 873–894.

[17] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, Deep learning, vol. 1, MIT press Cambridge, 2016.

[18] J. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences, 115 (2018), pp. 8505–8510.

[19] J. He, L. Li, J. Xu, and C. Zheng, Relu deep neural networks and linear finite elements, Journal of Computational Mathematics, 38 (2020), pp. 502–527.

[20] J. He and J. Xu, Mgnet: A unified framework of multigrid and convolutional neural network, Science China Mathematics, 62 (2019), pp. 1331–1354.

[21] V. Heuveline and F. Schieweck, On the inf-sup condition for higher order mixed fem on meshes with hanging nodes, ESAIM: Mathematical Modelling and Numerical Analysis, 41 (2007), pp. 1–20.

[22] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural networks, 2 (1989), pp. 359–366.

[23] K. Hutter, Theoretical glaciology: material science of ice and the mechanics of glaciers and ice sheets, vol. 1, Springer, 2017.

[24] T. Isaac, G. Stadler, and O. Ghattas, Solution of nonlinear stokes equations discretized by high-order finite elements on nonconforming and anisotropic meshes, with application to ice sheet dynamics, SIAM Journal on Scientific Computing, 37 (2015), pp. B804–B833.

[25] Jingrun, Chen, , 8405, , J. Chen, Rui, Du, , 8406, , R. Du, Keke, Wu, , 8407, , and K. Wu, A comparison study of deep galerkin method and deep ritz method for elliptic problems with different boundary conditions, Communications in Mathematical Research, 36 (2020), pp. 354–376.

[26] S. Karumuri, R. Tripathy, I. Bilionis, and J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, Journal of Computational Physics, 404 (2020), p. 109120.

[27] Y. Khoo, J. Lu, and L. Ying, Solving parametric PDE problems with artificial neural networks, European Journal of Applied Mathematics, Special Issue 3: Connections between Deep Learning and Partial Differential Equations, 32 (2021), pp. 421–435.

[28] D. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980, (2014).

[29] I. Lagaris, A. Likas, and D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw., 9 (1998), pp. 987–1000.

[30] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature, 521 (2015), p. 436.

[31] H. Lee and I. S. Kang, Neural algorithm for solving differential equations, Journal of Computational Physics, 91 (1990), pp. 110–131.

[32] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895, (2020).

[33] Z. Long, Y. Lu, and B. Dong, PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network, Journal of Computational Physics, 399 (2019), p. 108925.

[34] Z. Long, Y. Lu, X. Ma, and B. Dong, PDE-Net: Learning PDEs from data, International Conference on Machine Learning, (2018), pp. 3208–3216.

[35] J. Lu, Z. Shen, H. Yang, and S. Zhang, Deep network approximation for smooth functions, SIAM Journal on Mathematical Analysis, 53 (2021), pp. 5465–5506.

[36] L. Lu, P. Jin, and G. Karniadakis, Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, arXiv:1910.03193, (2019).

[37] D. A. May, J. Brown, and L. Le Pourhiet, A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous stokes flow, Computer methods in applied mechanics and engineering, 290 (2015), pp. 496–523.

[38] D. McKenzie, The generation and compaction of partially molten rock, Journal of petrology, 25 (1984), pp. 713–765.

[39] A. Meade and A. Fernandez, The numerical solution of linear ordinary differential equations by feedforward neural networks, Math. Comput. Model., 19 (1994), pp. 1–25.

[40] H. Montanelli and Q. Du, New error bounds for deep ReLU networks using sparse grids, SIAM Journal on Mathematics of Data Science, 1 (2019), pp. 78–92.

[41] W. S. B. Paterson, Physics of glaciers, Butterworth-Heinemann, 1994.

[42] F. Pattyn, Transient glacier response with a higher-order numerical ice-flow model, Journal of Glaciology, 48 (2002), pp. 467–477.

[43] F. Pattyn, L. Perichon, A. Aschwanden, B. Breuer, B. De Smedt, O. Gagliardini, G. H. Gudmundsson, R. C. Hindmarsh, A. Hubbard, J. V. Johnson, et al., Benchmark experiments for higher-order and full-stokes ice sheet models (ismip–hom), The Cryosphere, 2 (2008), pp. 95–108.

[44] N. Petra, J. Martin, G. Stadler, and O. Ghattas, A computational framework for infinite-dimensional bayesian inverse problems, part ii: Stochastic newton mcmc with application to ice sheet flow inverse problems, SIAM Journal on Scientific Computing, 36 (2014), pp. A1525–A1555.

[45] N. Petra, H. Zhu, G. Stadler, T. J. Hughes, and O. Ghattas, An inexact gauss-newton method for inversion of basal sliding and rheology parameters in a nonlinear stokes ice sheet model, Journal of Glaciology, 58 (2012), pp. 889–903.

[46] A. Pinkus, Approximation theory of the MLP model in neural networks, Acta numerica, 8 (1999), pp. 143–195.

[47] T. Qin, K. Wu, and D. Xiu, Data driven governing equations approximation using deep neural networks, Journal of Computational Physics, 395 (2019), pp. 620–635.

[48] M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378 (2019), pp. 686–707.

[49] J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. Staar, Y. Ineichen, C. Bekas, A. Curioni, and O. Ghattas, An extreme-scale implicit solver for complex pdes: highly heterogeneous flow in earth's mantle, in Proceedings of the international conference for high performance computing, networking, storage and analysis, 2015, pp. 1–12.

[50] S. Rudy, J. Kutz, and S. Brunton, Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, Journal of Computational Physics, 396 (2019), pp. 483–506.

[51] G. Schubert, D. L. Turcotte, and P. Olson, Mantle convection in the Earth and planets, Cambridge University Press, 2001.

[52] C. Schwab and M. Suri, Mixed hp finite element methods for stokes and non-newtonian flow, Computer methods in applied mechanics and engineering, 175 (1999), pp. 217–241.

[53] C. Schwab and J. Zech, Deep Learning in High Dimension, Research Report, vol. 2017 (2017).

[54] Z. Shen, H. Yang, and S. Zhang, Deep network with approximation error being reciprocal of width to power of square root of depth, Neural Computation, 33 (2021), pp. 1005–1036.

[55] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, Journal of Computational Physics, 375 (2018), pp. 1339–1364.

[56] R. Stenberg and M. Suri, Mixed *hp* finite element methods for problems in elasticity and stokes flow, Numerische Mathematik, 72 (1996), pp. 367–389.

[57] Y. Wang, S. Cheung, E. Chung, Y. Efendiev, and M. Wang, Deep multiscale model learning, Journal of Computational Physics, 406 (2020), p. 109071.

[58] Z. Wang, J. Xin, and Z. Zhang, DeepParticle: learning invariant measure by a deep neural network minimizing wasserstein distance on data generated from an interacting particle method, Journal of Computational Physics, (2022), p. 111309.

[59] Z. Wang and Z. Zhang, A mesh-free method for interface problems using the deep learning approach, Journal of Computational Physics, 400 (2020), p. 108963.

[60] K. Wu and D. Xiu, Data-driven deep learning of partial differential equations in modal space, Journal of Computational Physics, (2020), p. 109307.

[61] L. Yang, X. Meng, and G. Karniadakis, B-PINNS: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, J. Comp. Physics, 425 (2021), p. 109913.

[62] D. Yarotsky, Error bounds for approximations with deep ReLU networks, Neural Networks, 94 (2017), pp. 103–114.

[63] Y. Zang, G. Bao, X. Ye, and H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, Journal of Computational Physics, 411 (2020), p. 109409.

[64] Y. Zhu and N. Zabaras, Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification, Journal of Computational Physics, 366 (2018), pp. 415–447.

[65] Y. Zhu, N. Zabaras, P. Koutsourelakis, and P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics, 394 (2019), pp. 56–81.