Journal of Glaciology



Article

Cite this article: Jouvet G, Cordonnier G (2023). Ice-flow model emulator based on physics-informed deep learning. *Journal of Glaciology* **69**(278), 1941–1955. https://doi.org/10.1017/jog.2023.73

Received: 17 April 2023
Revised: 17 August 2023
Accepted: 18 August 2023
First published online: 26 September 2023

Keywords:
glacier flow; glacier modelling; glacier mechanics; ice-sheet modelling

Corresponding author:
Guillaume Jouvet;
Email: guillaume.jouvet@unil.ch

© The Author(s), 2023. Published by Cambridge University Press on behalf of International Glaciological Society. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

cambridge.org/jog

Ice-flow model emulator based on physics-informed deep learning

Guillaume Jouvet[1] and Guillaume Cordonnier[2]

[1]Université de Lausanne, IDYST, 1015 Lausanne, Switzerland and [2]Inria, Université Côte d'Azur, Sophia-Antipolis, France

Abstract

Convolutional neural networks (CNN) trained from high-order ice-flow model realisations have proven to be outstanding emulators in terms of fidelity and computational performance. However, the dependence on an ensemble of realisations of an instructor model renders this strategy difficult to generalise to a variety of ice-flow regimes found in the nature. To overcome this issue, we adopt the approach of physics-informed deep learning, which fuses traditional numerical solutions by finite differences/elements and deep-learning approaches. Here, we train a CNN to minimise the energy associated with high-order ice-flow equations within the time iterations of a glacier evolution model. As a result, our emulator is a promising alternative to traditional solvers thanks to its high computational efficiency (especially on GPU), its high fidelity to the original model, its simplified training (without requiring any data), its capability to handle a variety of ice-flow regimes and memorise previous solutions, and its relatively simple implementation. Embedded into the 'Instructed Glacier Model' (IGM) framework, the potential of the emulator is illustrated with three applications including a large-scale high-resolution (2400x4000) forward glacier evolution model, an inverse modelling case for data assimilation, and an ice shelf.

1. Introduction

In glacier and ice-sheet models, ice is commonly described as a viscous non-Newtonian (Glen, 1953) fluid whose motion is governed by the 3D non-linear Glen–Stokes equations (Greve and Blatter, 2009). Solving these equations usually remains very costly compared to other glacial underlying processes. To reduce the costs, the ice-flow equations are often simplified by neglecting higher-order terms in the aspect ratio of the ice domain ϵ (thickness versus length) considering it to be usually small. The truncation of the second-order terms in ϵ yields the first-order approximation (FOA) model (Blatter, 1995), which consists of a 3D non-linear elliptic equation (Colinge and Rappaz, 1999) for the horizontal velocity and remains expensive. Going one step further, the shallow ice approximation (Hutter, 1983) (SIA) is obtained after dropping the first-order terms in ϵ in the FOA model. As a result, the analytical solution of SIA is computationally inexpensive to implement. The SIA remains a reference model for many applications (e.g. Maussion and others, 2019), despite strongly simplifying mechanical assumptions and applicability limited to areas where ice flow is dominated by vertical shearing (Greve and Blatter, 2009). The transfer of numerical methods from central processing units (CPU) on graphics processing units (GPU) architectures is currently a promising approach to bypass the computational bottleneck associated with high-order modelling (Brædstrup and others, 2014), however, massive parallelisation of solvers on GPU remains a complex task (Räss and others, 2020).

As an alternative to traditional solvers, deep-learning surrogate models (or emulators) have been found very promising in reducing computational costs with minimal loss of accuracy (Brinkerhoff and others, 2021; Jouvet and others, 2022; He and others, 2023). Deep learning is based on artificial neural networks (ANNs), which are trained to capture the most essential relationship between the input and the output of an instructor model. The ANN is intended to be an efficient substitute for the original model within the range defined by the training dataset. Following this strategy, the computationally expensive Glen–Stokes model could be emulated by a simple convolutional neural network (CNN) by Jouvet and others (2022) with a speedup of several orders of magnitude and high fidelity levels in the case of mountain glaciers, and major benefits for inverse modelling purposes (Jouvet, 2023). Another key asset of ANNs is that they run very efficiently on GPUs, permitting additional significant speed-ups, especially when modelling high spatial resolution domains. However, the dependence on an instructor model makes the training of such an emulator technically difficult, not very flexible, and therefore limits its ability to generalise its validity range beyond the training data and its given spatial resolution.

In recent years, physics-informed neural networks (PINNs) have emerged as a powerful approach in surrogate modelling to directly enforce physical laws (such as partial differential equations) in the learning process instead of matching datasets generated from physical models (e.g. Raissi and others, 2019). Basic PINNs are trained to minimise the residual associated with the equations and the boundary conditions (Markidis, 2021). In contrast, variational PINNs (VPINNs) exploit the minimisation form (or equivalently the variational form) of the problem as a loss function (Kharazmi and others, 2019), which has the advantage of

involving derivatives of lower orders compared to residuals. An important aspect of VPINNs is their connections with traditional finite element methods (FEM). For example, a standard FEM solver applied to an elliptic problem represents the solution in a finite element approximation space spanned by mesh-defined basis functions and seeks the function that minimises the associated energy in the approximation space (Ern and Guermond, 2004). On the contrary, the deep Ritz method proposed by Yu (2018) (which belongs to the category of VPINN) represents the solution as a neural network in an approximation space generated by the parameters of a neural network.

In ice-flow modelling, PINNs have been used by Riel and others (2021) to learn the time evolution of drag in glacier beds from observations of ice velocity and elevation and by Riel and Minchew (2022) to calibrate ice-flow law parameters and perform uncertainty quantification. Recently, Cui and others (2022) proposed a mesh-free method to solve Glen–Stokes equations using an approach inspired by the deep Ritz method.

In this paper, we propose two different methods to compute FOA ice flow efficiently on GPU by exploiting the minimisation form associated with the FOA model and using optimisation techniques based on automatic differentiation and stochastic gradient. The first one is a conventional numerical solver, which is used mostly here as a reference to evaluate the second one. The second one on which the paper focuses is an emulator based on deep learning. In more detail, we take the CNN ice-flow emulator introduced previously by Jouvet and others (2022) and propose a new training strategy inspired by VPINN to remove the dependence on an instructor model and obtain a more generic emulator that is easier to implement and faster to train. Here, we train our CNN ice-flow emulator at minimising directly the energy instead of minimising the misfit with solutions from an instructor model as done previously (Fig. 1). A similar approach was used by Cordonnier and others (2023) for modelling terrain formation by glacial erosion. Their target was to generate realistic images in computer graphics, whereas we propose a thorough evaluation of the method and its potential for glaciological applications.

This paper is structured as follows: first, we introduce the physical model which includes the ice-flow FOA model and its minimisation formulation. Second, we describe the numerical model which includes the spatial discretisation, the energy-based FOA solver and deep-learning emulator. Last, we present and discuss our assessment results and examples of modelling applications.

2. Physical model

Let Ω be a rectangular horizontal domain supporting a glacier/volume of ice V at time t. Glacier bedrock and surface interfaces are defined by functions $b(x,\ y)$ and $s(x,\ y,\ t)$ where $(x,\ y) \in \Omega$. According to these definitions, the ice thickness h is defined as being the difference between the two: $h(x,\ y,\ t) = s(x,\ y,\ t) - b(x,\ y)$, and the three-dimensional volume of ice V is defined as

$$V = \{(x, y, z),\ b(x, y) \leq z \leq s(x, y, t),\quad (x, y) \in \Omega\},$$

which has two boundaries: the bedrock

$$\Gamma_b = \{(x, y, z),\ z = b(x, y),\quad (x, y) \in \Omega\}$$

and the surface

$$\Gamma_s = \{(x, y, z),\ z = s(x, y, t),\quad (x, y) \in \Omega\}$$

interfaces, see Figure 2. The two interfaces coincide in ice-free areas.

Given an initial glacier geometry, the time evolution in ice thickness $h(x,\ y,\ t)$ is determined by the mass conservation equation, which couples ice dynamics and surface mass balance (SMB) through:

$$\frac{\partial h}{\partial t} + \nabla \cdot (\bar{\mathbf{u}} h) = \text{SMB}, \tag{1}$$

where $\nabla\cdot$ denotes the divergence operator with respect to horizontal variables $(x,\ y)$, $\bar{\mathbf{u}} = (\bar{u},\ \bar{v})$ is the vertically averaged horizontal ice velocity field and SMB the SMB function, which consists of the integration of ice accumulation and ablation over one year. Equation (1) is generic and can be applied to model glacier evolution in number of applications provided adequate SMB and ice-flow model components. In the following, we mostly focus on developing an efficient numerical method to compute the ice flow $\bar{\mathbf{u}}$ considering it is often the most computationally expensive component in glacier evolution model (Jouvet and others, 2022).

2.1 Glen–Stokes model

The Stokes model consists of the momentum conservation equation when inertial terms are ignored, together with the incompressibility condition:

$$-\nabla \cdot \sigma = \rho \mathbf{g},\ \text{in}\ \ V, \tag{2}$$

$$\nabla \cdot \mathbf{u} = 0,\ \text{in}\ \ V, \tag{3}$$

where σ is the Cauchy stress tensor, $\mathbf{g} = (0,\ 0, -g)$, g is the gravitational constant and $\mathbf{u} = (u_x,\ u_y,\ u_z)$ is the 3D velocity field. Let τ be the deviatoric stress tensor defined by

$$\sigma = \tau - PI, \tag{4}$$

where I is the identity tensor, P is the pressure field, with the requirement that $\text{tr}(\tau) = 0$ so that $P = -(1/3)\text{tr}(\sigma)$. Glen's flow

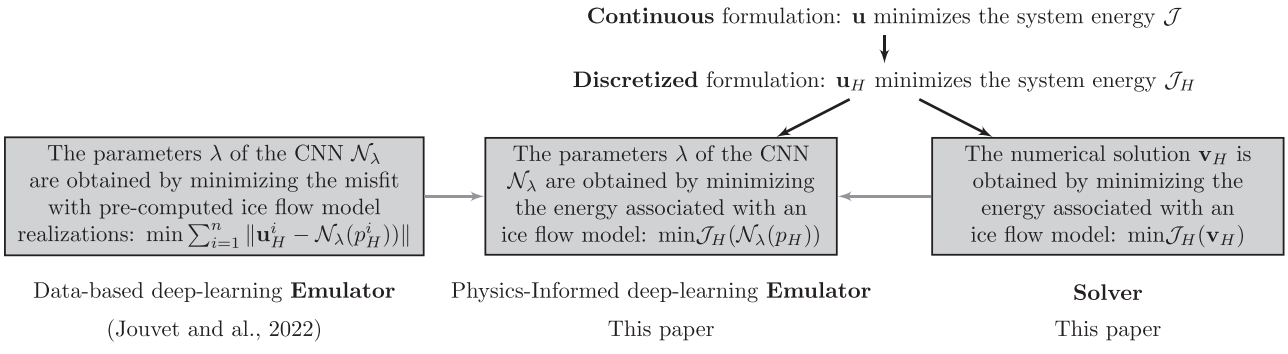


Figure 1. Our physics-informed deep-learning emulator can be seen as a fusion of data-driven deep-learning and traditional numerical-solving strategies.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

law (Glen, 1953), which describes the mechanical behaviour of ice, consists of the following non-linear relation:

$$\tau = 2\mu D(\mathbf{u}), \tag{5}$$

where $D(\mathbf{u})$ denotes the strain rate tensor defined by

$$D(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T), \tag{6}$$

μ is the viscosity defined by

$$\mu = \frac{1}{2} A^{-\frac{1}{n}} |D(\mathbf{u})|^{\frac{1}{n}-1}, \tag{7}$$

where $|Y| := \sqrt{(Y:Y)/2}$ denotes the norm associated with the scalar product $(:)$ (the sum of the element-wise product), $A = A(x,\ y) > 0$ is the Arrhenius factor and $n > 1$ is the Glen's exponent (here we take the most standard value $n = 3$). Note that A depends on the temperature of the ice (Paterson, 1994). For simplicity, this paper assumes vertically constant ice temperature; however, this assumption could be released without further difficulties.

2.2 Boundary conditions

The boundary conditions that supplement (2) and (3) are the following. Stress-free force applies to the ice–air interface,

$$\sigma \cdot \mathbf{n} = 0, \quad P = 0, \qquad \text{on } \Gamma_s, \tag{8}$$

where $\mathbf{n}$ is an outer normal vector along Γ_s. Along the lower surface interface, the non-linear Weertman friction condition reads (Hutter, 1983; Schoof and Hewitt, 2013)

$$\mathbf{u} \cdot \mathbf{n} = 0, \tag{9}$$

$$[(I - \mathbf{n}\mathbf{n}^T)\tau] \cdot \mathbf{n} = -c^{-m} |(I - \mathbf{n}\mathbf{n}^T) \cdot \mathbf{u}|^{m-1} (I - \mathbf{n}\mathbf{n}^T) \cdot \mathbf{u}, \tag{10}$$

on Γ_b for $k \in \{x,\ y\}$, where $m > 0$, $c = c(x,\ y) > 0$ and $\mathbf{n}$ is the outward normal unit vector to Γ_b. The relation (10) relates the basal shear stress $[(I - \mathbf{n}\mathbf{n}^T)\tau] \cdot \mathbf{n}$ to the sliding velocity $(I - \mathbf{n}\mathbf{n}^T) \cdot \mathbf{u}$, both of them projected onto the tangential plane. Note that $c = 0$ in case of no-sliding.

2.3 Minimisation formulation

The above-mentioned Glen–Stokes problem can be reformulated into variational and minimisation problems. We follow the derivation made by Jouvet (2016). For that, we consider the following divergence-free velocity field space (Girault and Raviart, 1986):

$$\mathcal{X} := \{\mathbf{v} \in [W^{1,1+\frac{1}{n}}(V)]^3, \quad \nabla \cdot \mathbf{v} = 0, \quad \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \Gamma_b\},$$

where $W^{1,p}$ is the appropriate Sobolev space (Adams and Fournier, 2003). The variational formulation associated with the Glen–Stokes problem writes: find $\mathbf{u} \in \mathcal{X}$ such that for all $\mathbf{v} \in \mathcal{X}$ we have:

$$\int_V A^{-\frac{1}{n}} |D(\mathbf{u})|^{\frac{1}{n}-1} (D(\mathbf{u}), D(\mathbf{v}))\, \mathrm{d}V \tag{11}$$

$$+ \int_{\Gamma_b} c^{-m} |\mathbf{u}|_M^{m-1} (\mathbf{u}, \mathbf{v})_M \, \mathrm{d}S + \rho g \int_V (\nabla s \cdot \mathbf{v})\, \mathrm{d}V = 0, \tag{12}$$

where the bilinear form $(\mathbf{a},\ \mathbf{b})_M := (M\mathbf{a}) \cdot \mathbf{b}$, and its associated norm $|\mathbf{a}|_M := \sqrt{(\mathbf{a},\ \mathbf{a})_M}$ have for matrix

$$M = \begin{pmatrix} I + (\nabla_{\mathbf{x}} b)(\nabla_{\mathbf{x}} b)^T & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix}. \tag{13}$$

The above problem is equivalent to seeking for $\mathbf{u} \in \mathcal{X}$ such that

$$\mathcal{J}(\mathbf{u}) = \min\{\mathcal{J}(\mathbf{v}), \mathbf{v} \in \mathcal{X}\}, \tag{14}$$

where the functional to be minimised is

$$\mathcal{J}(\mathbf{v}) = \int_V 2\frac{A^{-\frac{1}{n}}}{1+\frac{1}{n}} |D(\mathbf{v})|^{1+\frac{1}{n}} \, \mathrm{d}V + \int_{\Gamma_b} \frac{c^{-m}}{1+m} |\mathbf{v}|_M^{1+m} \, \mathrm{d}S + \rho g \int_V (\nabla s \cdot \mathbf{v})\, \mathrm{d}V. \tag{15}$$

It must be stressed that only the first term still depends on the vertical velocity in both formulations (12) and (15).

2.4 First-order approximation (FOA)

We introduce the aspect ratio $\epsilon = [h]/[\mathbf{x}]$ of the ice geometry V, where $[h]$ and $[\mathbf{x}]$ denote its typical height and length. It is easy to verify that in that the strain rate tensor $D(\mathbf{v})$ contains terms scaling with ϵ^{-1}, ϵ^0 and ϵ^1. As glaciers are usually thin objects with a small aspect ratio ϵ, it is a common practise to omit the highest order term. By doing so and invoking the incompressibility equation, the vertical velocity components ($\partial_x u_z$ and $\partial_y u_z$) of the strain rate tensor can be eliminated:

$$D(\mathbf{u}) = \begin{pmatrix} \partial_x u_x & \frac{1}{2}(\partial_y u_x + \partial_x u_y), & \frac{1}{2}(\partial_z u_x) \\ \frac{1}{2}(\partial_y u_x + \partial_x u_y) & \partial_y u_y & \frac{1}{2}(\partial_z u_y) \\ \frac{1}{2}(\partial_z u_x) & \frac{1}{2}(\partial_z u_y) & -\partial_x u_x - \partial_y u_y \end{pmatrix}. \tag{16}$$

In turn, this eliminates the vertical velocity component u_z from the ice-flow model. The resulting model (so-called first-order approximation, FOA, or Blatter–Pattyn model (Blatter, 1995)) is obtained by minimising the functional $\mathcal{J}$ defined in (15) with $D(\mathbf{u})$ defined by (16). Advantageously, the constraints of the functional space $\mathcal{X}$ disappear when removing the vertical component of the velocity. As a result, the FOA model consists of a three-dimensional, non-linear, elliptic and unconstrained problem, which is therefore simpler than the original Glen–Stokes problem. Provided enough friction at the bedrock (i.e. the coefficient c is not too high) and other suitable assumptions, one can show (Colinge and Rappaz, 1999; Schoof, 2006) that the functional $\mathcal{J}$ is continuous, strictly convex and coercive in the functional space $[W^{1,1+\frac{1}{n}}(V)]^2$; therefore, the FOA problem admits a unique solution.

3. Numerical model

The glacier evolution model (equipped with both the ice-flow solver and the physics-informed deep-learning emulator) is implemented in the 'Instructed Glacier Model' framework (IGM, https://github.com/jouvetg/igm), which can simulate glacier time evolution (on CPU or GPU) given an initial glacier geometry and SMB forcing (Jouvet and others, 2022). IGM is written in Python and relies on operations of the TensorFlow library to allow vector/parallel operations (such as used in neural networks) between large arrays that are computationally efficient

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

on GPU. At each time step, IGM updates in turn the SMB (e.g. equilibrium line altitude (ELA)-based parameterisation or climate-driven PDD model), the ice flow (solved or emulated according to the user's choice) and the ice thickness by solving conservation equation (1) using a first-order upwind finite-volume scheme on a regular 2D grid. Thus, the time step is computed adaptively to satisfy the CFL condition, make sure that the ice is never transported over more than one cell distance in one time step, and therefore to ensure numerical stability. We refer to Jouvet and others (2022) for more details on the transport numerical scheme. In what follows, we focus on the computation of the ice flow by numerical solving and deep-learning emulation.

3.1 Spatial discretisation

First, the horizontal rectangular domain Ω is discretised with a regular raster/structured grid of size $N_x \times N_y$ with constant cell spacing H in the x and y direction (Fig. 2, right panel). Variables such as the ice thickness h, the surface topography s, the rate factor A and the sliding coefficient c are defined at the corners of each grid cell of the horizontal grid. In the following, we use subscript H to denote these discrete quantities such as $\mathbf{u}_H$, h_H, s_H, A_H, c_H defined on the horizontal grid. Note that our choice of a structured grid (instead of any other type of discretisation) is essential to represent variables as 2D arrays and therefore to use CNNs for emulating the ice-flow mechanics later on. On the other hand, the ice thickness is discretised vertically using a fixed number of points N_z (in this paper we use $N_z = 10$). Layers are distributed according to a quadratic rule such that discretisation is fine close to the ice–bedrock interface (where the strongest gradients are expected) and coarse close to the ice–surface interface following the strategy given by (PISM, Khroulev, 2020). Subsequently, the approximation space X_H for velocities consists of piecewise linear functions defined at the corners of each grid cell in the horizontal direction and at the intersection of each layer in the vertical discretisation.

In finite elements, solving the non-linear elliptic FOA problem requires minimising the associated functional $\mathcal{J}$ in a finite-dimension approximation space X_H spanned by shape functions defined in the discretised domain instead of the full continuous solution space X. We follow a similar strategy here: given $p_H = (h_H,\ s_H,\ A_H,\ c_H)$, we seek for $\mathbf{u}_H \in X_H$ such that

$$\mathbf{u}_H = \text{argmin}\{\mathcal{J}_{p_H}(\mathbf{v}_H),\ \mathbf{v}_H \in X_H\} \tag{17}$$

where

$$\begin{aligned}\mathcal{J}_{p_H}(\mathbf{v}_H) = \int_\Omega \Bigg(& \frac{2A_H^{-\frac{1}{n}}}{1+\frac{1}{n}} \int_{s_H - h_H}^{s_H} |D_H(\mathbf{v}_H)|^{1+\frac{1}{n}}\, \mathrm{d}z \\ & + \frac{c_H^{-m}}{1+m} |\mathbf{v}_H|_M^{1+m}\, \mathrm{d}S \\ & + \rho g \int_{s_H - h_H}^{s_H} (\nabla s_H \cdot \mathbf{v}_H)\, \mathrm{d}z \Bigg)\, \mathrm{d}\Omega.\end{aligned} \tag{18}$$

For simplicity, D is approximated by a finite difference scheme on a 3D staggered grid (Fig. 2, right panel). As D involves derivatives in the three dimensions, we apply either a finite difference or cell averaging to ensure that all derivatives in (16) are approximated consistently on the same 3D staggered grid (i.e. at the centre of cells horizontally and at the middle of layers vertically). The two other terms (sliding and gravity force related) are also computed on the staggered grid (otherwise, this would cause numerical artefacts, typically chessboard modes). Due to the layer-wise vertical discretisation, we first compute the horizontal derivatives of D_H in a layer-dependent system of coordinate $(x,\ y,\ \tilde{z})$ where $\tilde{z} = z - l$ and l is the layer elevation, and transfer them in the reference system of coordinate $(x,\ y,\ z)$ using a simple rule of derivative: e.g. $\frac{\partial f}{\partial x} = \frac{\partial \tilde{f}}{\partial x} - \frac{\partial \tilde{f}}{\partial z}\frac{\partial l}{\partial x}$ for any quantity f (respectively $\tilde{f}$) defined in $(x,\ y,\ z)$ (respectively $(x,\ y,\ \tilde{z})$). Lastly, the integration of (18) is done numerically using the rectangle method. Note that

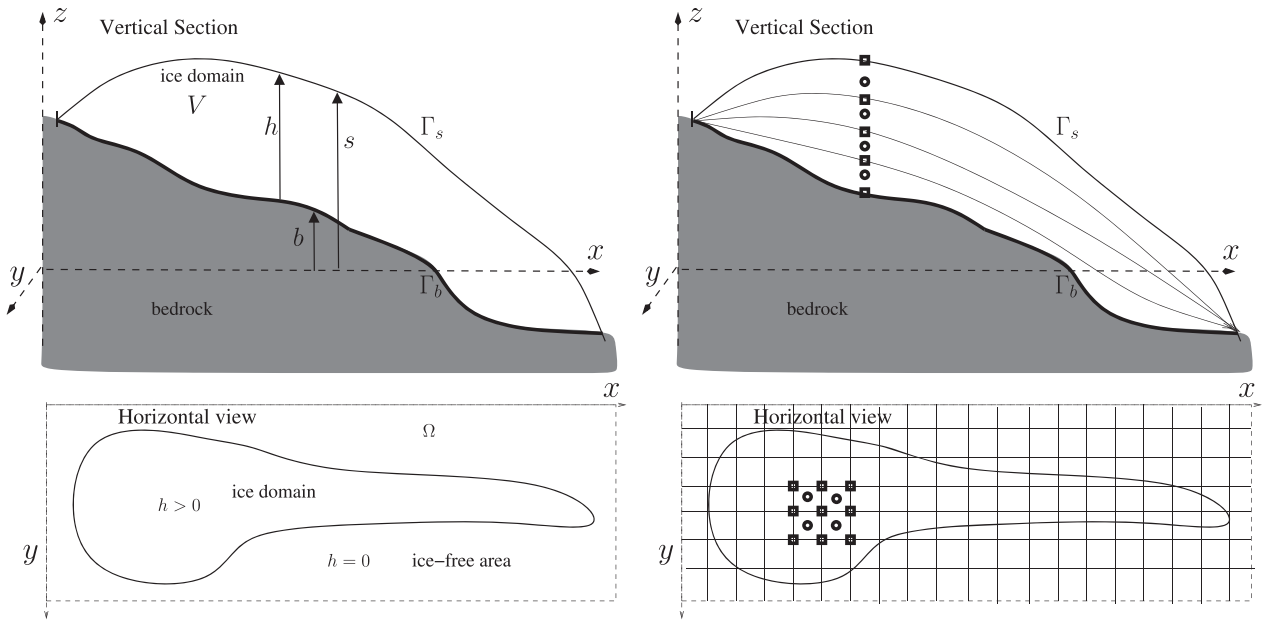


Figure 2. Cross-section and horizontal view of a glacier with notations (left panel) and its spatial discretisation (right panel), which is obtained using a regular horizontal grid and by subdividing the glacier into a pile of layers. All modelled variables (e.g. ice thickness) are computed at the corners of each cell of the 2D horizontal grid (materialised with squares) except the ice-flow velocities, which are computed on the 3D corresponding grid. In contrast, the strain rate is computed on the staggered grid at the centre of each cell and layer (vizualised with circles).

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

ice margins must be treated carefully to prevent singular vertical derivatives of D_H as the vertical step size tends to zero. To overcome this issue, we assume a minimum ice thickness of 1 m.

3.2 Solver

Our solver solves the convex optimisation problem (17) using a stochastic gradient descent method, namely the Adam optimiser (Kingma and Ba, 2014) with a step size of 1. Using the Keras (Chollet, 2015) and Tensorflow (Abadi, 2015) libraries, the derivatives of $\mathcal{J}_{p_H}$ with respect to $\mathbf{v}_H$ are obtained by automatic differentiation. When used for computing a single snapshot ice flow, the optimisation scheme is initialised with zero ice velocity. When used multiple times in a transient glacier evolution run, the gradient scheme uses the ice flow from the previous time step as initialisation to predict the next one. In the following, we refer to the 'solved' solution (in contrast to the 'emulated' solution defined in the next section), the result of the solver at convergence. Note that we found in Appendix B a very good agreement between the 'solved' and the reference solutions of the ISMIP-HOM (Pattyn, 2008) experiments. This test validates the numerical solver, as well as the implementation of the system energy in IGM, which is used for both the solver and the emulator.

3.3 Emulator

As an alternative to the previously introduced solver, we now propose an ice-flow emulator, which predicts horizontal ice flow $(\mathbf{u}_H,\ \mathbf{v}_H)$ from the input field p_H.

$$\begin{aligned} \mathcal{N}_\lambda : \{h_H, s_H, A_H, c_H, H_H\} &\longrightarrow \{\mathbf{u}_H, \mathbf{v}_H\} \\ \mathbb{R}^{N_X \times N_Y \times 5} &\longrightarrow \mathbb{R}^{N_X \times N_Y \times N_Z \times 2} \end{aligned} \tag{19}$$

where input and output can be seen as two- and three-dimensional multichannel fields, which are defined on the regular horizontal grid (Fig. 3). Having selected these input parameters allows us to develop a generic ice-flow emulator that can handle a large variety of glacier shapes, types of ice flow (from shearing to sliding dominant) and spatial resolutions. As the spatial resolution H_H is fixed in modelling application, including it as an input of the emulator is in fact not necessary. Here, we added H_H for convenience such that one can take advantage of an initial pre-trained emulator (Appendix A) irrespective of the spatial resolution.

As an emulator, we choose an ANN, which maps input to output variables by a sequential composition of linear and non-linear functions (or a sequence of network layers). Linear operations have weights $\lambda = \{\lambda_i,\ i = 1,\ ...,\ N\}$, which are optimised in the training stage. Here, we use a CNN (Long and others, 2015), which is a special type of ANN that additionally includes local convolution operations to learn spatially variable relationships (LeCun and others, 2015) and proved to be capable of learning high-order ice-flow models (Jouvet and others, 2022). Here, we retain the hyperparameters found by Jouvet and others (2022) as they provide a good trade-off between model fidelity and complexity: our CNN consists of 16 two-dimensional convolutional layers between input and output data (Fig. 3). Convolutional operations have a kernel matrix (or feature map) of size 3×3. A padding is used to conserve the frame size through the convolution operation. Convolutional operations are repeated using a sliding window with one stride across the input frame and 32 feature maps. As a non-linear activation function, we use leaky rectified linear units. As a result, our CNN has about 140 000 trainable parameters.

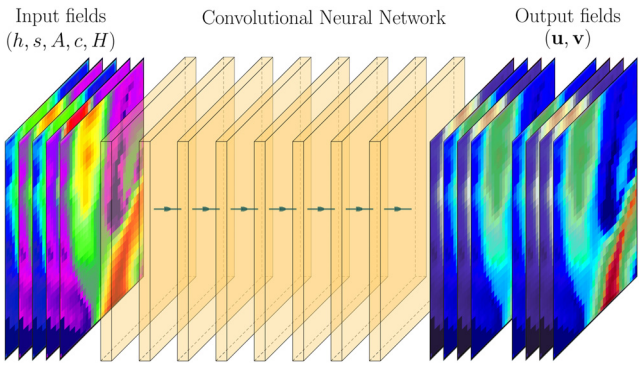


Figure 3. Our emulator consists of a CNN that maps geometrical (thickness and surface topography), ice-flow parameters (shearing and basal sliding) and spatial resolution inputs to 3D ice-flow fields.

While Jouvet and others (2022) proposed to train (19) by fitting to external ice-flow model realisations, we take here another strategy inspired from PINNs. We differ from traditional PINNs in two ways: first, PINNs usually map the coordinate of the sampling points to the physical output, which forces them to retrain the network for different settings, while our inputs are essential model parameters (the coordinates at each pixel are not explicitly passed). Second, PINNs usually minimise the residual of the equation and/or boundary conditions involved in the physical model (e.g. Markidis, 2021). Instead, we adopt the different VPINN strategy (Kharazmi and others, 2019) by minimising the energy associated with the FOA model instead of the residual (Fig. 1). In more detail, the training consists of finding the weights of the CNN $\lambda = \{\lambda_i,\ i = 1,\ ...,\ N\}$ that minimise:

$$\lambda = \text{argmin}\big(\mathcal{J}_{p_H}(\mathcal{N}_\lambda(p_H))\big), \tag{20}$$

given geometrical and glaciological input data p_H. The optimisation problem (20) is solved again using the Adam optimiser (Kingma and Ba, 2014) – the derivatives of $\mathcal{J}_H$ with respect to λ being obtained from automatic differentiation. At first view, the minimisation problem (20) is expected to be more difficult to solve than that in (17), as there is no guarantee that $\mathcal{J}_{p_H}$ is convex with respect to the training parameters λ. On the other hand, problem (20) is expected to have much fewer control parameters (the number of training parameters is on the order of 10^5) than problem (17), which may have much more control parameters $(2 \times N_z \times N_y \times N_x)$ when treating a large-scale array.

While there exist different strategies for initialising the weights of CNNs, we found in Appendix A that using a CNN pretrained over a large glacier catalogue (Fig. A1) facilitates the convergence of the emulator, presumably because the diversity of the catalogue prevents against falling into local minima. The optimisation of the CNN is therefore always initialised with pre-trained weights (Appendix A). When used for computing a single snapshot ice flow, we use an adaptive learning strategy including an exponential decay to launch the training aggressively ($\sim 10^{-4}$) for efficiency and to end it gently for fine-tuning ($\sim 10^{-5}$). When used in a transient glacier evolution run, one performs a single step of gentle ($\sim 10^{-5}$) training per iteration (or each X iteration to vary the degree of training) starting from the lastly trained emulator.

4. Results

In this section, we present fidelity and computational performance results of the 'emulated' solution towards the 'solved' solution simulations with different strengths of emulator training. For that purpose, we consider two glaciers of different sizes (i) the present-day Aletsch Glacier, Switzerland, which is the current largest glacier of the European Alps and (ii) the former Valais Glacier, Switzerland, which covered a large part of Switzerland

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

during the last glacial maximum. The experiments for these two glaciers cover different applications from individual glaciers on a small grid (244×179 at 100 m of resolution for Aletsch) relevant for the modelling of today's glaciers to large ice fields on large grid (700×700 at 200 m of resolution for Valais) more relevant for paleo glacier modelling. We conduct two kinds of experiments in turn: (i) the computation of a snapshot solution to assess the best accuracy we should expect from the emulator without consideration for the computational price, and (ii) the computation of a transient solution to assess both fidelity and computational performance in a modelling application.

4.1 Fidelity of snapshot solutions

First, we consider the topography and ice thickness of Aletsch and Valais glaciers at a given time and fix the ice-flow parameters (A, c) to constant physical values ($A = 78$ MPa^{-3} a^{-1}, $c = 10$ km MPa^{-3}). Based on these geometries, we computed two numerical solutions: (i) a 'solved' one $\mathbf{u}_H$ obtained by minimising (18) within the space of solutions X_h, (ii) an 'emulated' one $\mathcal{N}_\lambda(p_H)$ obtained by minimising (20) in the space of parameters of the CNN. Figures 4 and 5 present the results in terms of 'solved' solution at convergence (panel A), 'emulated' solution at emulator convergence (panel B), difference between 'solved' and 'emulated' solutions (panel C), decrease of the system energy during solving and training (panels D and E), and L_1 error of the emulated $\mathbf{u}_E$ towards the solved $\mathbf{u}_S$ solution through training iterations (panel F) defined by:

$$E_{L_1} = \int_\Omega \int_b^{b+h} |\mathbf{u}_E - \mathbf{u}_S|. \tag{21}$$

In general, the Adam optimiser succeeds at minimising the energy and capturing the 'solved' and the 'emulated' solutions. Indeed, the energy associated with the 'emulated' solution decreases towards a value (∼−0.34) that is relatively close to the value obtained when solving (∼−0.35), demonstrating that our CNN has learnt well to minimise the energy for the Aletsch case (Fig. 4E). Most importantly, the $L1$ error is small (∼1.2 m a^{-1}, Fig. 4F), showing that the 'emulated' solution is very similar to the reference 'solved'. An in-depth comparison of the spatial pattern of the two solutions (Fig. 4C) reveals minor and unevenly distributed discrepancies. Interestingly, the Valais glacier case shows a larger energy gap (∼ 10%, Fig. 5E), and a larger $L1$ error norm (∼3.6 m a^{-1}, Fig. 5F), which remains small considering the velocity scale (0–300 m a^{-1}). Comparing the spatial pattern of the two solutions (Fig. 5C) shows that the error is mostly concentrated on the most prominent glacier tongue. The slight deterioration of the accuracy from the Aletsch to the Valais case can be explained as follows: an emulator trained to a single glacier in a small region is naturally expected to be more accurate (as more customised) than an emulator of the same complexity trained to a larger glacier network. Note that increasing the size of the CNN (increasing the number of layers, feature maps or kernel size) have shown slight but not significant improvements of fidelity.

For computing a single snapshot ice-flow field at a given time, the solver was found to be more efficient than the emulator in terms of convergence and then in terms of computational performance, presumably due to different convexity properties. In the next section, we show that the opposite is true when we consider the evolution of a glacier over time.

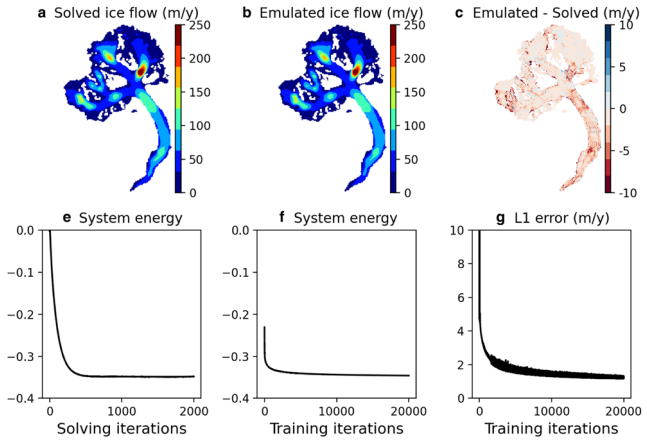


Figure 4. Results of the solver and the emulator for the snapshot experiment related to Aletsch glacier. Panels A and B show the magnitude of the ice-flow velocities obtained by solving and emulation, respectively. Panel C shows the difference between the two. Panels D and E show the decrease of the system energy through iterations. Panel F shows L_1 error of the emulated towards the solved solution through training iterations.

4.2 Fidelity of transient solutions

For each glacier (Aletsch and Valais), we now perform two kinds of transient experiments: (i) the first (referred to as 'ELA-varying') assumes fixed ice-flow parameters (A and c), and forces the SMB with time-varying ELAs; (ii) the second (referred as 'A/c-varying') assumes fixed ELA and force time-varying ice-flow parameters (A and c). The goal of these two experiments is to test the memory capacity of the deep-learning emulator. As SMB, we use a simple parameterisation based on a given ELA z_{ELA}, vertical gradients of accumulation and ablation, and maximum accumulation rate:

$$SMB(z) = \begin{cases} \min\left(0.003 \times (z - z_{\text{ELA}}), 1\right), & \text{if } z \geq z_{\text{ELA}} \\ 0.006 \times (z - z_{\text{ELA}}), & \text{otherwise.} \end{cases}$$

Prior to running experiments, we collected the bedrock topography of the two regions (Grab, 2020), initialised the model with ice-free conditions and ran it with ice-flow parameters $c = 10$ km MPa^{-3} a^{-1} and $A = 78$ MPa^{-3} a^{-1} and mass-balance parameters $z_{\text{ELA}} = 2800$ m asl, and $z_{\text{ELA}} = 2200$ m asl for Aletsch and Valais, respectively. The goal of this preliminary phase is to simulate the build-up of glaciers until they reach a steady-state shape. Then, the ELA-varying transient experiment consists of modelling 2000 years (starting from the obtained steady-state shape,

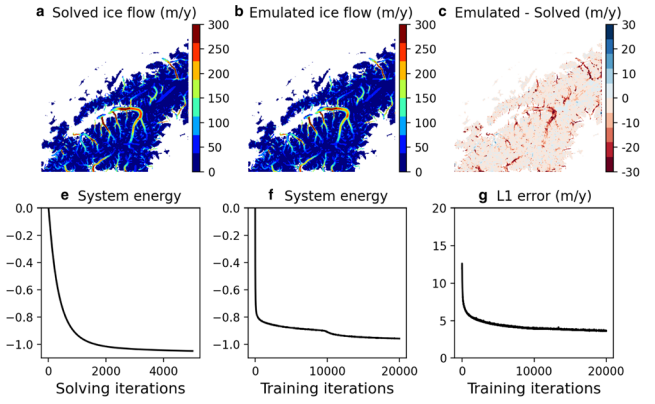


Figure 5. Results of the solver and the emulator for the snapshot experiment related to Valais glacier. The meaning of panels is similar to Figure 4.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

and keeping the parameters constant) with the following ELA parametrisation:

$$z_{\mathrm{ELA}} = 2800 + 200 \times \sin(\pi t/500)\ \mathrm{m},$$
$$z_{\mathrm{ELA}} = 2200 + 300 \times \sin(\pi t/500)\ \mathrm{m},$$

for the Aletsch and Valais glaciers, respectively. On the other hand, the A/c-varying transient experiment consists of running the model for 2000 years (starting from the obtained steady-state shape and keeping the parameters constant) with the following ice-flow parameters:

$$A = 78 + 22 \times \sin(\pi t/500)\ \mathrm{MPa}^{-3}\mathrm{a}^{-1},$$
$$c = 10 + 5 \times \sin(\pi t/500)\ \mathrm{km\ MPa}^{-3}\mathrm{a}^{-1},$$

to induce glacier variations (retreat-advance-retreat), and explore a variety of configurations for assessment.

The experiments were performed using the solver (our reference run) and the emulator (pretrained on a glacier catalogue, Appendix A) with different retraining strategies to compute the ice dynamics: (i) with no retraining at all (Experiment '0–0%'), (ii) with 100% retraining during the first 1000 years (i.e. one step of retraining per iteration) and then no retraining (Experiment '100–0%'), (iii) with 100% retraining during the first 1000 years and then 10% retraining (i.e. on step of retraining each 10 iteration) during the second 1000 years (Experiment '100–10%'). Table 1 summaries the design and the outcomes in terms of fidelity of all experiments. Figures 6 and 7 show the results of the ELA and A/c-varying experiments for Aletsch and Valais Glacier, respectively, in terms of fidelity ($L1$ error) of the 'emulated' solution $\mathbf{u}_E$ to the reference 'solved' one $\mathbf{u}_S$, and overall ice volume.

As a result, the pretrained emulator without further retraining (0–0%) captures roughly the ice flow in the ELA-varying and A/c-varying experiments of Aletsch Glacier when ice-flow parameters are fixed with an $L1$ error of $\sim$5 m a^{-1} (Fig. 6), which is fairly small compared to the velocity scale (0–200 m a^{-1}). This shows that the shape of the Aletsch Glacier is relatively well represented in the pretraining glacier catalogue (Fig. 12). Therefore, the emulator has acquired a fair knowledge to predict a solution in line with the 'solved' one. However, emulator-induced cumulative errors lead to an increasing bias in ice volume (Fig. 6). In contrast, the pretrained emulator performs very poorly with the Valais Glacier (very high $L1$ error in Fig. 7). This is likely due to the fact that the glaciers of this experiment go well beyond the glaciers in the catalogue (Fig. A1) in terms of shape, size and ice-flow behaviour.

In contrast, our results reveal that adaptive retraining of the emulator (100–0%) shows largely improved accuracy with respect to the 'solved' reference solution during the first 1000 years. Indeed, retraining damps the $L1$ error to small values: $\sim$1 and $\sim$4 m a^{-1} in the Aletsch and the Valais Glacier experiments, respectively (Figs 6, 7) in the first 1000 years when one retraining step is applied to each time step. These errors as well as the spatial patterns of the error (not shown) are very similar to the ones found in the snapshot experiments (Figs 4, 5, panels A and B) with discrepancies, mostly in the trunk of Valais Glacier. As a result, the modelled volumes agree very well with the 'solved' solution when systematic retraining is used (Figs 6, 7). It must be stressed that using more than one training iteration per time step did not show significant reduction of the $L1$ error.

Table 1. Design and results of ELA and A/c-varying experiments with various retraining strategies

Exp. name	Training level		Resulting fidelity	
	[0,1] ky	[1,2] ky	[0,1] ky	[1,2] ky
0–0%	0%	0%	Med./low	Med./low
100–0%	100%	0%	High	Med./low
100–10%	100%	10%	High	High

As systematic online retraining during the first 1000 years is a relatively costly task (next section), we analyse the effect of releasing the retraining to assess the capability of the emulator to retain the ice-flow solutions accurately (Figs 6, 7). As a result, switching off the retraining after 1000 years of simulation and repeating the experiments with the same forcing for another 1000 years (100–0%) reveal different outcomes. Indeed, the emulator 'retains' some of the relevant training in ELA-varying experiments, but deteriorates very quickly in the A/c-varying experiments, leading to notable biases in ice volume (Figs 6, 7). In contrast, the emulator remains as accurate as in the first phase when lightly retrained each 10 time steps (100–10%) in the second phase. This means that the emulator has mostly retained the geometry–ice flow relationship during the first pass and that the accuracy can be maintained with a light computationally effective retraining provided an initial systematic training.

An important parameter for online retraining is the learning rate. A too low parameter (gently learning) will result in inefficient learning and solution biases, while a too high parameter (aggressive learning) will result in erratic/non-smooth accuracy curve and deteriorated memory of the emulator (not shown). As a trade-off between the two cases, we found that a learning rate of 2×10^{-5} is optimal in all our transient experiments.

4.3 Computational performance of transient solutions

We now compare the computational performance of the three solutions: 'solved', 'emulated without online retraining' and 'emulated with online retraining' to lead the ELA and A/c-varying experiments presented in the previous section. Comparing the emulator and the solver is a challenge, as the first requires only one emulation step (the retraining does not require to be performed more than once per time iteration), while the solver may require several iterations per time step to converge. For this reason, we first discuss the costs associated with one individual step (i.e. one iteration of retraining or solving of the optimisation algorithm) before analysing the overall costs.

Table 2 gathers together the computational times needed to achieve one step of (i) solving, (ii) emulating and (iii) retraining for modelling domains of various sizes, and on both CPU and GPU architectures of the same desktop computer (equipped with a 10-core Intel CPU i9–10900 K and a 10 000 cores Nvidia GPU RTX 3090). As a result, the GPU (which has 1000 times more cores) systematically out-performs the CPU. While the CPU may be interesting for small-scale array domains, Table 2 shows that it is not a viable option to treat large-scale arrays. Therefore, we focus our performance analysis on the GPU only. We find that the emulation step is the most affordable task, followed by the solving step, which is slightly (about 30%) more expensive, and the retraining step, which is about three times more expensive than emulation regardless of the domain size. This can be explained as follows. The emulation step is inexpensive as it only requires a single pass of the CNN. On the other hand, the solving step consists of a forward evaluation of the system energy followed by the computation of the energy gradients and an update of the ice flow. Last, the retraining step is naturally expected to be more costly than the 'emulation + solving', as it combines the tasks of the two: one CNN evaluation, one system energy evaluation, the computation of the two gradients and an update of the weights of the CNN.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

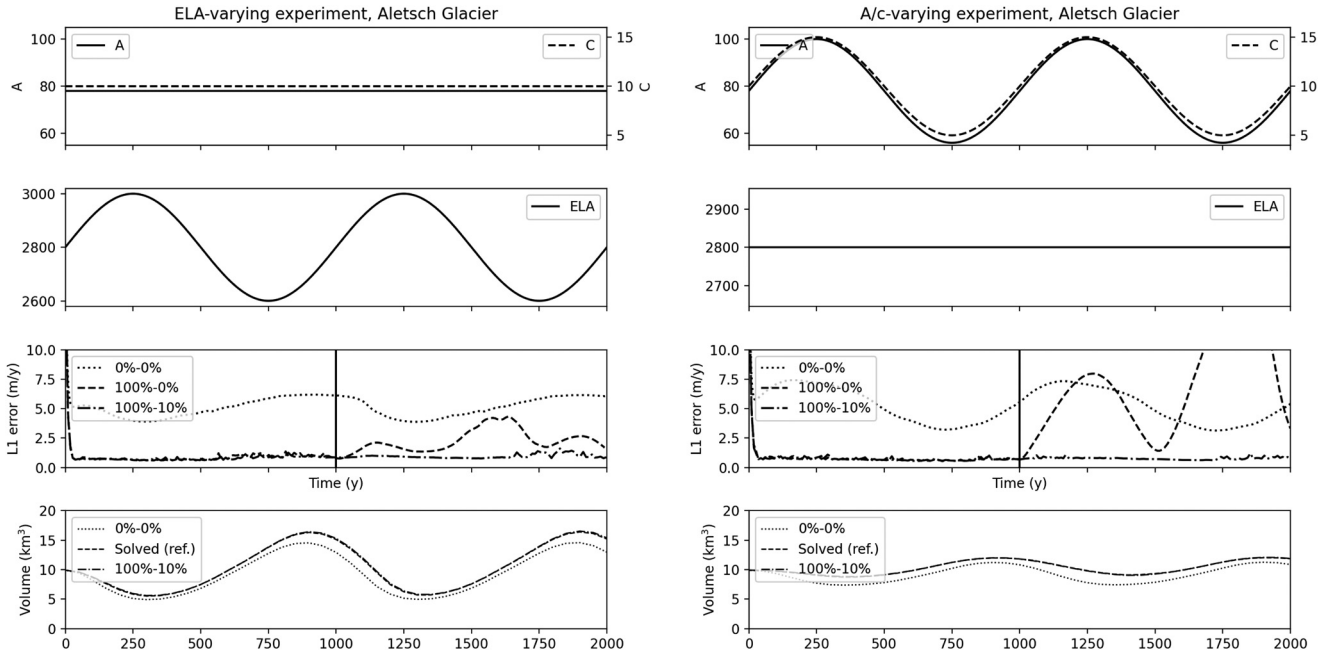


Figure 6. Transient results of the ELA-varying (left panels) and *A*/*c*-varying (right panels) transient modelling experiments for Aletsch Glacier. The panels indicate the time evolution of input parameters (ice-flow parameters and ELA), the resulting ice-flow *L*1 error between all 'emulated' solutions (with and without retraining) and the 'solved' one and the output ice volume obtained with the three modelling methods ('solved', 'emulated' with and without retraining).

Since a CNN is evaluated sequentially layer by layer, the emulation step is memory efficient. Therefore, emulation step can be performed on large arrays (i.e. we achieved 2400×4000 with our 24 Gb GPU, Table 2), while the solving and retraining steps are more memory-demanding and therefore more limited by the GPU available memory. For example, none of the solving and retraining steps for the 2400×4000 domain were achievable with our GPU (we found that a maximum grid of about 2000×2000). Hopefully, this limitation can be overcome for the retraining (and not for the solving step, Table 2) by splitting the domain into smaller patches and sequentially retraining the emulator patch-wise.

As the other modules (ice thickness and mass-balance updates) are computationally inexpensive compared to the ice-flow model, the overall cost is mainly the number of time iterations times the costs of individual emulation (with or without retraining) or solver steps. In the ELA and *A*/*c*-varying experiments related to Aletsch and Valais glaciers, the time step was on the order of 0.1 y to maintain numerical stability meaning that ~10 000 time iterations were needed per millennium of modelling irrespective of the chosen method (solver or emulation). When using the solver, several iterations were required to reach convergence at a given time step, however, this number is case-dependent: $\sim 3-4$ and more than ten iterations in the case of

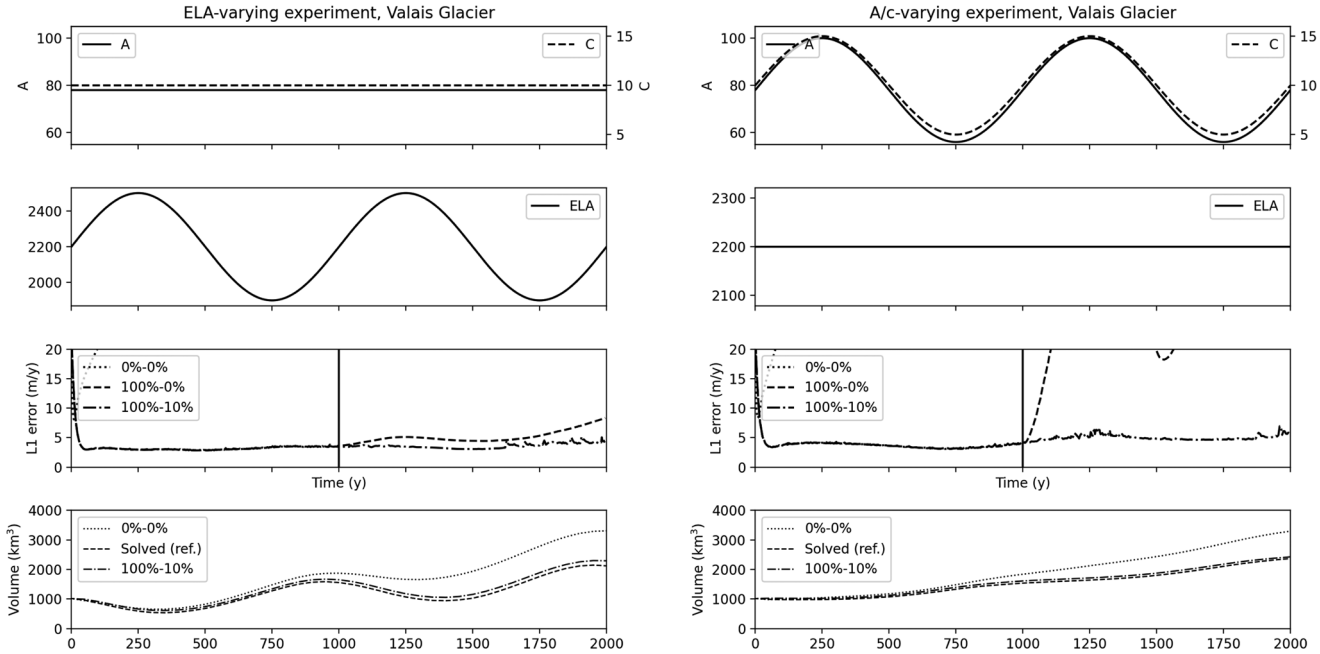


Figure 7. Transient results of the ELA-varying (left panels) and *A*/*c*-varying (right panels) transient modelling experiments for Valais Glacier. This is similar to the caption of Figure 6.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

Aletsch and Valais glaciers, respectively. These numbers should be taken with care, as a more efficient optimiser (e.g. Newton-like) may reduce the number of required iterations. In contrast with the solver, the emulation only requires one step, while the retraining can be applied infrequently while remaining effective. In our case, the best trade-off in terms of accuracy to computational performance was found using light retraining (each 10 iterations) as it maintained accuracy (Figs 6, 7) at the cost of one cheap emulation per time step plus more expensive but infrequent retraining steps (Table 2).

5. Applications

In this section, we illustrate the potential of our physics-informed ice-flow emulator for glaciological applications.

5.1 Paleo glacier modelling in the European Alps

Modelling paleo-glacier evolution is an important tool for understanding the history of glaciations. However, the long time scales and the size of the domain may render this exercise computationally very demanding. For example, the 120 000-year-long simulation of alpine glacier evolution in the Alps of Jouvet and others (2023) at 2 km with the Parallel Ice Sheet Model (PISM, Khroulev, 2020) would take several weeks of computational time on a 10 core i9–10900 K running at 3.70 GHz. It is, therefore, prohibitively expensive to explore subkilometre resolutions that would be required to resolve the complex topography of the Alps in the highest reaches. Therefore, the ice-flow emulator with online retraining is a promising approach to overcome the computational bottleneck, especially on GPU, which allows large array computations. Here, we test its capability to simulate the paleo evolution of glaciers in the entire European Alps in very high resolution (200 m) over 10 000 years encompassing the Last Glacial Maximum (LGM, about 24 000 years ago).

To this end, we took over the model setting of Jouvet and others (2023). Initialising with ice-free conditions and today's topography of the Alps as bedrock, IGM was forced with a coupled modelled paleoclimate data and PDD SMB model (Hock, 1999) from 28 000 years BP to 18 000 years BP. As a result, the 200 m IGM simulation at 21 000 years BP shows highly detailed glacier extents resolving small valleys and Nunataks (Fig. 8), and took about 2 days of computations on a ~10 000-core RTX 3090 1.70 Ghz GPU. Here, the GPU has 24 GB memory, which is key to treating very large arrays. The horizontal grid covers the entire Alps at 200 m yielding a resolution of 2400×4000. This exercise illustrates the capability of our approach to achieving very high resolutions at affordable computational costs. For comparison, PISM at a much lower resolution (2 km resolution, 240×400) would take about the same time to carry a similar simulation on a 10-core 3.70 GHz CPU. Of course, this comparison must be tempered by the fact that IGM does not include all the many physical components of PISM, especially the thermodynamics of ice, which is known to add substantial computational time.

Table 2. Computational time required (in average) to perform one emulation, retraining, solving iteration step in modelling experiments for Aletsch, Valais and the entire Alps

Exp	Step	CPU	GPU
Aletsch	Solver	125 ms	15 ms
244x179	Emulator	39 ms	11 ms
	Retrain	533 ms	29 ms
Valais	Solver	1538 ms	51 ms
700x700	Emulator	468 ms	38 ms
	Retrain	5592 ms	110 ms
Entire Alps	Solver	X	X
2400x4000	Emulator	X	360 ms
	Retrain	X	1465 ms

In the latter case, we reported 'X' when the computation was not possible, or prohibitively too expensive. The CPU (i9–10900 K) has 10 3.70 GHz cores with 64 Gb RAM while the GPU (RTX 3090) has about 10 000 1.70 GHz cores with 24 Gb RAM.

5.2 Ice-flow model inversion/data assimilation

Inverse modelling is an essential step to initialise present-day glacier models, i.e. estimate unknown variables (such as ice thickness and/or ice-flow parameters) such that the model matches at best observations (surface ice-flow velocities or pointwise ice thickness profiles). Substituting the ice-flow equations with a CNN emulator allows solving the inverse model (or the underlying optimisation problem) very efficiently by utilising automatic differentiation and stochastic gradient methods (Jouvet, 2023). Therefore, the CNN emulator trained by physics-informed deep learning can also be used in a similar way. Most importantly, one can now simultaneously optimise the CNN parameters to fit the ice physics by minimising the system energy and the CNN inputs to match observations by minimising the misfit to the data. The coupled optimisation allows to perform the inversion with an accurate and customised-to-the-glacier CNN at the same time.

As an illustration, we solve the inversion problem for Aletsch Glacier proposed by Jouvet (2023) with this new strategy. Given present-day pointwise ice thickness measurements and surface ice velocity measurements, we use the CNN trained offline over the glacier catalogue, and seek alternatively for the CNN weights λ, the ice thickness distribution h and the distributed sliding parameter c, such that both the system energy (Eqn (20)) and the mismatch between the observed and modelled quantities (Eqn (5) in Jouvet (2023)) are minimised. Note that the regularisation terms for h and c are added to enforce smoothness and ensure a unique solution. As a result, Figure 9 shows the convergence of the fields towards an optimal state and the reduction of the corresponding misfit values in terms of standard deviations. Here, the quality of data assimilation is comparable to that obtained by Jouvet (2023). However, the simultaneous emulator training/optimisation has a major benefit with respect to the former method (based on offline training): the online retraining permits to account for spatial variations of the sliding coefficient (Fig. 9, top-right panel) and makes the emulator nearly as accurate as the solver (Fig. 10). In contrast, the former emulator, which met only the glacier catalogue and spatially constant sliding coefficient at training, suffers from larger biases as observed in Appendix A.

5.3 Ice shelf

Ice shelves behave very differently to mountain glacier ice flow as modelled in the two previous applications. Indeed, they can be very fast due to the absence of friction under floating ice, and are therefore dominated by basal sliding. By contrast, friction under grounded glaciers usually induces an important vertical shearing component. Yet, modelling accurately the dynamics of ice shelves is essential to predict the evolution of the Antarctic ice sheet under climate change and the resulting sea-level rise (Seroussi and others, 2020). Here, we demonstrate that IGM equipped with the new physics-informed deep-learning emulator has an important potential for modelling ice-sheet/shelf systems by performing a simple experiment inspired by the Marine Ice Sheet Model Inter-comparison Project (MISMIP, Pattyn and others, 2012). The goal here is not to run all exercise simulations,

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

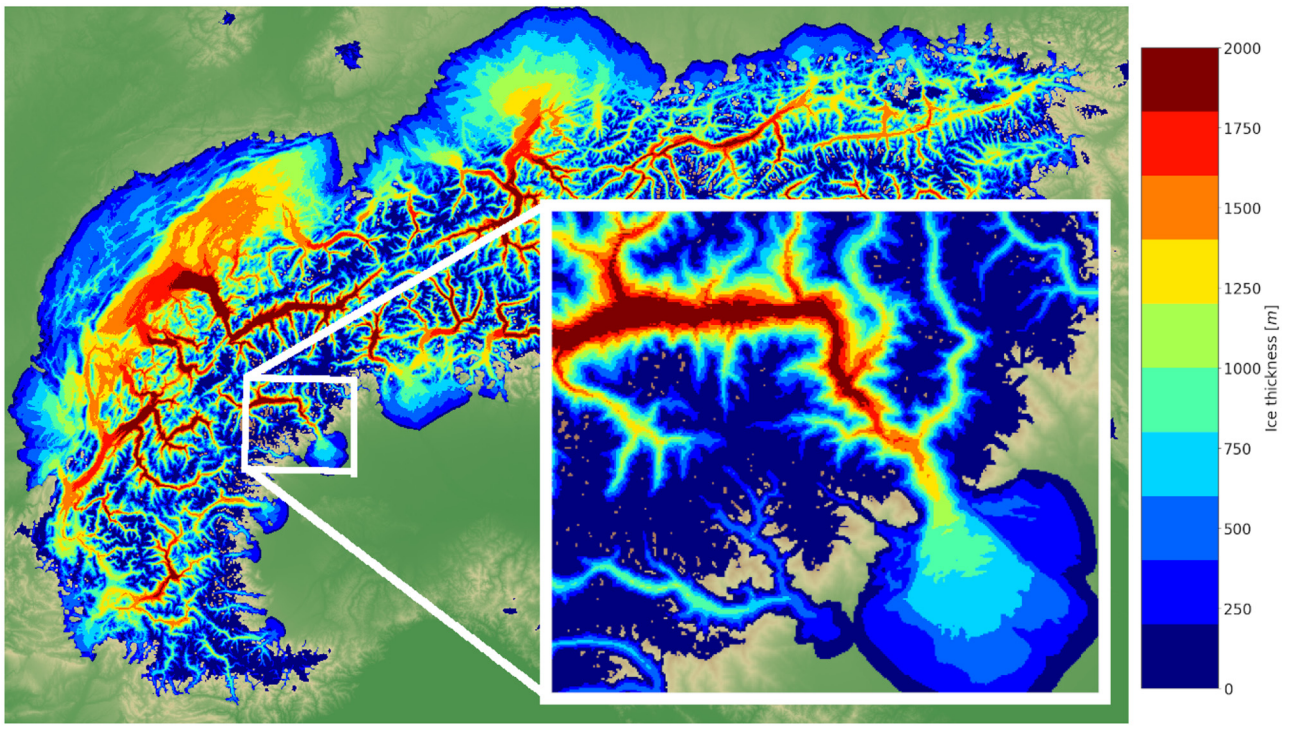


Figure 8. Ice thickness of the alpine ice field obtained at 24 000 years BP modelled with IGM at 200 m of resolution.

but only to compute the ice dynamics associated with one state to prove the capacity of the emulator to handle sliding-dominant ice flow of ice shelves.

For that purpose, we consider an idealised ice-sheet–ice-shelf geometry lying on a ramp of constant slope in the x-direction over a distance of $L_x = 1100$ km (Fig. 11). All geometrical variables are constant in the y-direction to mimic the 2D MISMIP experiment 1 (Pattyn and others, 2012). In that configuration, we distinguish the ice sheet ($x < x_{\mathrm{GL}}$) and the ice shelf ($x > x_{\mathrm{GL}}$) from the grounding location $x_{\mathrm{GL}} \sim 966.5$ km (Fig. 11). The lower surface elevation l is either the bedrock when the ice is grounded or determined by Archimedes's principle when the ice is floating: $l = \max\{b, -(\rho_{\mathrm{i}}/\rho_{\mathrm{w}})h\}$, where $\rho_{\mathrm{i}} = 910$ kg m^{-3} and $\rho_{\mathrm{w}} = 1000$ kg m^{-3} denote the densities of ice and water, respectively. Here, we use the following parameters: $A = 146.5$ MPa^{-3} a^{-1}, $m = 1/3$, $c = 71.2$ km MPa^{-3} a^{-1} where the ice is grounded and $c^{-1} = 0$ km MPa^{-3} a^{-1} where the ice is floating (no friction). In addition, we use the 'shallow shelf approximation' model (Morland, 1987) instead of the FOA by simply setting a single layer in the vertical discretisation (Fig. 2, right panel), which is equivalent to assuming vertically constant ice-flow velocities. Lastly, the function $\mathcal{J}$ defined by (15) is augmented with an additional term to account for balance stress conditions between ice and water columns at the calving front (CF) on the extreme right of the modelled domain (Fig. 11):

$$-\int_{\mathrm{CF}} \frac{1}{2}\left(1 - \frac{\rho_{\mathrm{i}}}{\rho_{\mathrm{w}}}\right)\rho_{\mathrm{i}} g h^2 v \cdot \mathbf{n}, \tag{22}$$

where $\mathbf{n}$ is an outer normal vector along CF (Schoof, 2006). The above condition was implemented along the other terms of the

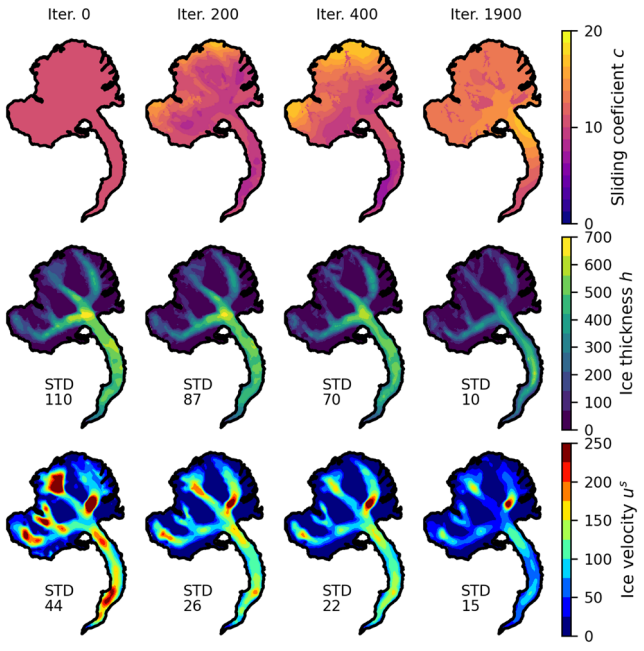


Figure 9. Evolution of the sliding distribution c (unit: km MPa^{-3} a^{-1}), the ice thickness distribution h (unit: m), as well as resulting surface ice-flow velocity field $\mathbf{u}^s$ (unit: m y^{-1}) through the iterations of the optimisation problem for Aletsch glacier. The standard deviation (STD) between the modelled and observed fields is reported at each step.

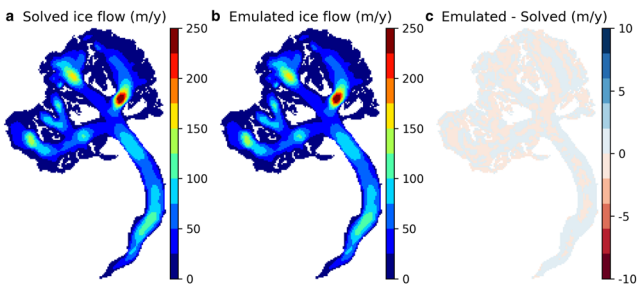


Figure 10. Surface ice-flow field of Aletsch Glacier with the parameters found after performing the simultaneous inversion and emulator training: (A) using the solver and (B) using the retrained emulator. Panel (C) shows the spatial difference between the two.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

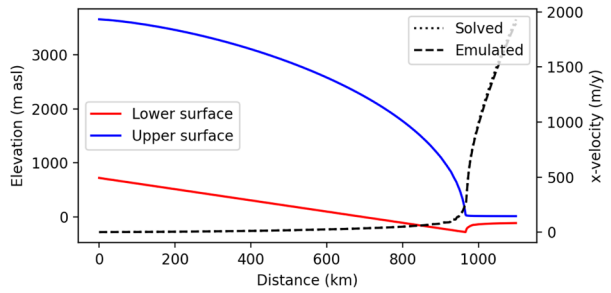


Figure 11. MISMIP-inspired ice geometry of the ice-shelf experiment along the *x*-axis, and resulting ice-flow velocities modelled from the solver and the emulator with custom training on the specific geometry.

system energy, and a 2D field (namely (22) along the calving front and zero elsewhere) was added to the emulator inputs (Eqn (19)) to control this boundary condition.

As a result, we find that after training the emulator on the specific geometry, the 'Solved' and 'Emulated' ice-flow fields along the *x*-axis are nearly identical (Fig. 11). This experiment demonstrates that the approach of the paper is not limited to grounded glacier flow, but is capable to handle the sliding-dominant flow of ice shelves.

6. Discussion and conclusions

In this paper, we have introduced both a solver and a physics-informed deep-learning emulator for modelling high-order ice flow on a regular grid that are designed to run efficiently on GPU. The solver relies on a stochastic gradient method and automatic differentiation tools to efficiently minimise the energy associated with the underlying ice-flow equations discretised by finite differences, similarly to Ritz–Galerkin methods in the finite element framework. On the other hand, the emulator relies on a CNN, which is trained to minimise the same energy. Therefore, our method (which belongs to the category of deep Ritz) can be seen as a fusion of finite element and deep-learning approaches. Here, our approximation space for the ice flow is induced by the training parameters of our CNN instead of being spanned by finite element basis functions. As a result, we have shown that our emulator can reproduce the solutions of the solver with high fidelity. Unlike the former emulator (Jouvet and others, 2022), the new one does not require any data from an external ice-flow model, as it enforces the ice-flow physics directly in learning. Here, we used a glacier catalogue to pre-train the emulator and obtain a good initial guess that facilitates convergence. However, adaptive online training within the time-stepping of a glacier evolution model does not require any data and has proven to significantly improve the emulator accuracy. This strategy makes the new emulator generic, as it allows exploration of any parameters, types of ice flow, spatial resolutions and glacier shapes, while the validity of the former emulator could not be ensured beyond the 'hull' defined by the data and its associated spatial resolution used for training. In addition, CNN training is therefore significantly easier and cheaper as no data are required. Last, our new emulator models the full 3D ice-flow field (instead of the vertical average horizontal speeds with the former version), which can be advantageous for some applications (e.g. Lagrangian 3D particle tracking).

The computational benefits of using a CNN emulator (Jouvet and others, 2022) remain unchanged. Indeed, one CNN forward evaluation can be done very efficiently, especially on GPU. In contrast, the solving and training steps are computationally more expensive (by a factor of 3 in our experiments). Therefore, to obtain the best computational performances, we mitigate the amount of training by limiting the frequency of retraining. Indeed, the memory capability of the CNN revealed in our experiments allows us to reduce the training costs for a given application. For instance, we found that a light cost-effective online retraining following a first systematic training is sufficient to maintain accuracy, as the CNN conserves most of the previously learnt solutions. Therefore, training costs can be strongly reduced in some modelling applications that meet several times similar glacier configurations (e.g. in paleo glacier modelling with repeated glacial cycle, or in parameter sensitivity analysis), yielding low overall computational costs.

There are a number of aspects that may be improved in the method presented in this paper. First, we used here the simplest finite-difference scheme to discretise the spatial derivatives in the strain rate on a staggered grid for simplicity. A more elaborated finite-element-like discretisation is expected to yield a more accurate solution, possibly slightly increasing the training costs but without affecting the emulation costs. Second, we used here the Adam optimiser as it proved to be robust and simple to implement, however, other optimisers may improve the convergence. For example, the (deterministic) L-BFGS-B optimiser has proven to be efficient at fine-optimising PINNs after an initial coarse pass with Adam to avoid local minima (Taylor and others, 2022). Similarly, the solver can be improved, and hybrid solver/emulation strategies that take advantage of two should be further investigated (e.g. using the emulator for preconditioning purpose or to help finding an initial guess). Third, here we investigated retraining strategies (to get the best accuracy while minimising the amount of retraining) in an empirical way by quantifying a posteriori the error between the emulated and solved solutions. Future research should investigate more effective and generic retraining strategies, e.g. seeking for an a priori error estimate of the neural network approximation (e.g. Minakowski and Richter, 2023) as done FEM for estimating the numerical error Ern and Guermond (2004). Lastly, the loss of accuracy with increasing domain size is another aspect of the emulator that should be improved, e.g. by using multiple region-specific emulators. It must be stressed that our CNN emulator (computationally efficient on GPU) strongly relies on the structured discretisation grid assumption. Therefore, emulating ice flow on more complex mesh (e.g. with local refinements) would require to follow a different strategy (e.g. PINNs).

Our modelling experiments have shown that the new emulator embedded in a glacier evolution model can handle very efficiently large-scale and/or high-resolution domain arrays and/or very long time scales. Therefore, our method has a high potential for paleo-glacier simulations. Additionally, we found that the emulator is suitable for both inverse and forward modelling. Therefore, the method can be very beneficial to assimilate data and run prognostic models of present-day glaciers on a global scale. Lastly, we have shown that our approach can be extended to fast-flowing ice as found in tidewater glaciers, opening promising perspectives for modelling the Antarctica and Greenland ice sheets in high spatial resolution.

Data. The code to run any solver-based or emulator-based glacier evolution simulations is open-source, relatively simple and publicly available with the 'Instructed Glacier Model' (IGM, https://github.com/jouvetg/igm).

Acknowledgements. We acknowledge Jacob Downs and an anonymous referee for their valuable comments on the original manuscript.

Author contributions. G. J. conceived the study, wrote the code, performed the simulations and wrote the article. G. C. developed simultaneously a similar approach, provided valuable feedback on the method and the results and helped to improve the manuscript.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

References

Abadi M (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Adams R and Fournier J (2003) *Sobolev Spaces*. Pure and Applied Mathematics. Elsevier Science.

Blatter H (1995) Velocity and stress fields in grounded glaciers: a simple algorithm for including deviatoric stress gradients. *Journal of Glaciology* **41**(138), 333–344. doi:10.3189/S002214300001621X

Brædstrup CF, Damsgaard A and Egholm DL (2014) Ice-sheet modelling accelerated by graphics cards. *Computers & Geosciences* **72**, 210–220. doi:10.1016/j.cageo.2014.07.019

Brinkerhoff D, Aschwanden A and Fahnestock M (2021) Constraining subglacial processes from surface velocity observations using surrogate-based Bayesian inference. *Journal of Glaciology* **67**(263), 1–19. doi:10.1017/jog.2020.112

Chollet F (2015) *Keras*. GitHub. https://github.com/fchollet/keras

Colinge J and Rappaz J (1999) A strongly nonlinear problem arising in glaciology. *M2AN Mathematical Modelling and Numerical Analysis* **33**(2), 395–406. doi:10.1051/m2an:1999122

Cordonnier G and 8 others (2023) Forming terrains by glacial erosion. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)* **42**(4), 1–14. doi: 10.1145/3592422.

Cui T, Wang Z and Zhang Z (2022) A variational neural network approach for glacier modelling with nonlinear rheology. *arXiv preprint arXiv:2209.02088*.

Ern A and Guermond JL (2004) *Theory and Practice of Finite Elements*, Vol. **159**. New York: Springer.

Girault V and Raviart P (1986) *Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms*. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer.

Glen JW (1953) Rate of flow of polycrystalline ice. *Nature* **172**, 721–722. doi:10.1038/172721a0

Grab M (2020) Swiss glacier thickness – release 2020. doi:10.3929/ethz-b-000434697.

Greve R and Blatter H (2009) *Dynamics of Ice Sheets and Glaciers*. Heidelberg: Springer Verlag.

He Q, Perego M, Howard AA, Karniadakis GE and Stinis P (2023) A hybrid deep neural operator/finite element method for ice-sheet modeling. ArXiv.

Hock R (1999) A distributed temperature-index ice- and snowmelt model including potential direct solar radiation. *Journal of Glaciology* **45**(149), 101–111. doi:10.3189/S0022143000003087

Hutter K (1983) *Theoretical Glaciology*. Dordrecht: Reidel.

Jouvet G (2016) Mechanical error estimators for shallow ice flow models. *Journal of Fluid Mechanics* **807**, 40–61. doi:10.1017/jfm.2016.593

Jouvet G (2023) Inversion of a Stokes glacier flow model emulated by deep learning. *Journal of Glaciology* **69**(273), 13–26. doi:10.1017/jog.2022.41

Jouvet G and 5 others (2022) Deep learning speeds up ice flow modelling by several orders of magnitude. *Journal of Glaciology* **68**(270), 651–664. doi:10.1017/jog.2021.120

Jouvet G and 11 others (2023) Coupled climate-glacier modelling of the last glaciation in the Alps. *Journal of Glaciology*, 1–17. https://doi.org/10.1017/jog.2023.74

Kharazmi E, Zhang Z and Karniadakis GE (2019) Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*.

Khroulev C and the PISM Authors (2020) PISM, a Parallel Ice Sheet Model v1.2: User's Manual.

Kingma DP and Ba J (2014) Adam: a method for stochastic optimization. preprint arXiv:1412.6980.

LeCun Y, Bengio Y and Hinton G (2015) Deep learning. *Nature* **521**(7553), 436–444. doi:10.1038/nature14539

Long J, Shelhamer E and Darrell T (2015) Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440. doi:10.1109/CVPR.2015.7298965

Markidis S (2021) The old and the new: can physics-informed deep-learning replace traditional linear solvers? *Frontiers in big Data* **92**, 1–15. doi: 10.3389/fdata.2021.669097

Maussion F and 9 others (2019) The Open Global Glacier Model (OGGM) v1. 1. *Geoscientific Model Development* **12**(3), 909–931. doi:10.5194/gmd12-909-2019

Minakowski P and Richter T (2023) A priori and a posteriori error estimates for the deep Ritz method applied to the Laplace and Stokes problem. *Journal of Computational and Applied Mathematics* **421**, 114845. doi:10.1016/j.cam.2022.114845

Morland L (1987) Unconfined ice-shelf flow. In van der Veen C and Oerlemans J eds. *Dynamics of the West Antarctic Ice Sheet*, volume 4 of Glaciology and Quaternary Geology. Dordrecht: Springer Netherlands, pp. 99–116.

Paterson WSB (1994) *The Physics of Glaciers*, 3rd Edn. New York: Pergamon.

Pattyn F (2008) Benchmark experiments for higher-order and full-Stokes ice sheet models (ISMIP-HOM). *The Cryosphere* **2**(2), 95–108. doi:10.5194/tc-2-95-2008

Pattyn F and 18 others (2012) Results of the Marine Ice Sheet Model Intercomparison Project, MISMIP. *The Cryosphere* **6**(3), 573–588. doi:10.5194/tc-6-573-2012

Raissi M, Perdikaris P and Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics* **378**, 686–707. doi:10.1016/j.jcp.2018.10.045

Räss L, Licul A, Herman F, Podladchikov YY and Suckale J (2020) Modelling thermomechanical ice deformation using an implicit pseudo-transient method (FastICE v1. 0) based on graphical processing units (GPUs). *Geoscientific Model Development* **13**(3), 955–976. doi:10.5194/gmd-13-955-2020

Riel B and Minchew B (2022) High-dimensional flow law parameter calibration and uncertainty quantification over Antarctic ice shelves: a variational Bayesian approach using deep learning. *Authorea Preprints*. doi:10.1002/essoar.10509116.1

Riel B, Minchew B and Bischoff T (2021) Data-driven inference of the mechanics of slip along glacier beds using physics-informed neural networks: case study on Rutford Ice Stream, Antarctica. *Journal of Advances in Modeling Earth Systems* **13**(11), 1–27. doi: 10.1029/2021MS002621

Schoof C (2006) Variational methods for glacier flow over plastic till. *Journal of Fluid Mechanics* **555**, 299–320. doi:10.1017/S0022112006009104

Schoof C and Hewitt I (2013) Ice-sheet dynamics. *Annual Review of Fluid Mechanics* **45**(1), 217–239. doi:10.1146/annurev-fluid-011212-140632

Seroussi H and 9 others (2020) ISMIP6 Antarctica: a multi-model ensemble of the Antarctic ice sheet evolution over the 21st century. *The Cryosphere* **14**(9), 3033–3070. doi:10.5194/tc-14-3033-2020

Taylor J, Wang W, Bala B and Bednarz T (2022) Optimizing the optimizer for data driven deep neural networks and physics informed neural networks. doi:10.48550/arXiv.2205.07430

Yu B (2018) The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics* **6**(1), 1–12. doi:10.1007/s40304-018-0127-z

APPENDIX A. Pre-training over a glacier catalogue

Pre-training of the CNN emulator over a glacier catalogue was found beneficial, especially to avoid local minima during online training, and improve the memory capability of the emulator. Here, we describe the implementation of the pre-training, and assess the accuracy of the pre-trained emulator with respect to the solver. To generate glacier shape inputs in an offline training process of the CNN, we use a glacier catalogue of 36 mountain glaciers at eight different times and 100 m resolution (covering advancing and retreating stages) obtained by Jouvet and others (2022) by glacier evolution simulations (Fig. A1). Further details about the construction of this catalogue are given in Appendix C of Jouvet and others (2022). The catalogue consists of a heterogeneous dataset with a large variety of possible glacier shapes (large/narrow, thin/thick, flat/steep, long/small, straight/curved glaciers, etc.).

First, we fix the ice-flow parameters (A, c) and the spatial resolution H to constant standard values ($A = 78$ MPa^{-3} a^{-1}, $c = 10$ km MPa^{-3}, $H = 100$ m) for simplicity. In a second experiment, we will vary these parameters at training.

A test glacier is selected in addition to the glacier catalogue, and a 'solved' ice-flow solution is obtained for this glacier by minimising the associated energy with the Adam optimiser. Figure A2 presents the results in terms of input data (panels A and B), 'solved' solution (panel C) and a decrease in system energy (panel D).

Aside from the solver, we have trained a CNN emulator to minimise the system energy (solving the optimisation problem (20)) over the entire glacier catalogue (excluding the test glacier, Fig. A1), and evaluated its performance to reproduce the previously 'solved' solution on a test glacier. As the size of the dataset is considerable, one used batches (a batch size of 8 was used

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

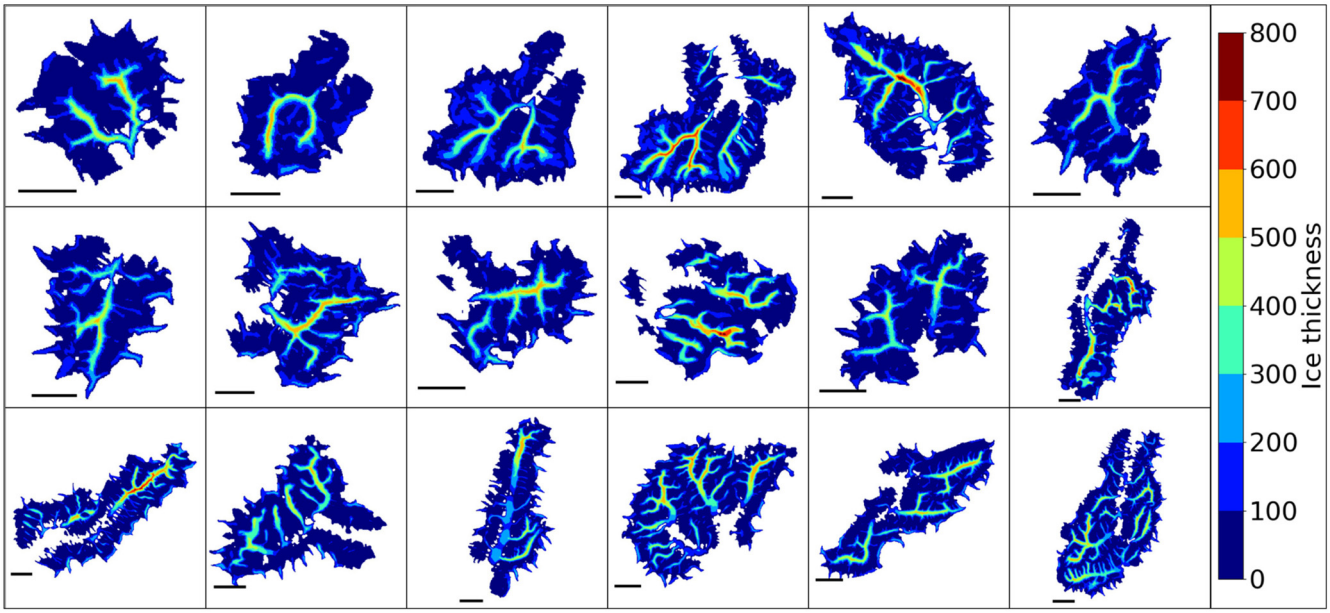


Figure A1. Ice thickness at their maximum extent of half of the glacier catalogue (18 of the 36). Each glacier shape is a snapshot of a simulation initialised with ice-free conditions, and forced with a surface mass balance that permits building and retreat in successive phases over a total of 200 years. The horizontal bar represents 5 km to give the scale of each glacier.

here) to facilitate convergence (previously only a single glacier sample was used for online training at each iteration). In addition, we used an adaptive learning rate including an exponential decay to launch the training aggressively (10^{-4}) for efficiency and to end it gently (10^{-6}) for fine-tuning. Lastly, we have re-initialized the learning-rate each 5000 training iterations to prevent falling in local minima.

Figure A3 presents the results in terms of 'emulated' solution when the training has converged (panel A), the difference between 'solved' and 'emulated' solutions (panel B), the L_1 error (panel C) and the decrease in the system energy through training iterations (panel D). As a result, the evolution of the L_1 error (panel C, Fig. A3) shows that the emulator captures well the ice flow after about 3000 iterations (the $L1$ error drops to ∼10 m/y). The effect of the adaptive learning rate (initially fixed at 10^{-4}, with exponential decay) is clearly visible: the first stage of training (iterations 0–1000) shows the largest decays and oscillations, while the last stage (iterations 4000–5000) is characterised by a smoother but slower decay. Interestingly, the energy associated with the 'emulated' solution decreases towards a value (∼−2.2) that is relatively close to the value obtained when solving (∼−2.3), demonstrating that our CNN has learnt well to minimise the energy. Although the 'emulated' and 'solved' solutions show a fair degree of similarity (compare panel C of Fig. A2 with panel A of Fig. A3), the spatial pattern of the difference between the two (Fig. A3, panel B) reveals that the error is unevenly distributed, the highest discrepancy being found on the most prominent glacier tongue. This is presumably due to the relatively poor representation of large, fast-flowing glacier tongues in the glacier catalogue compared to a smaller one (Jouvet and others, 2022).

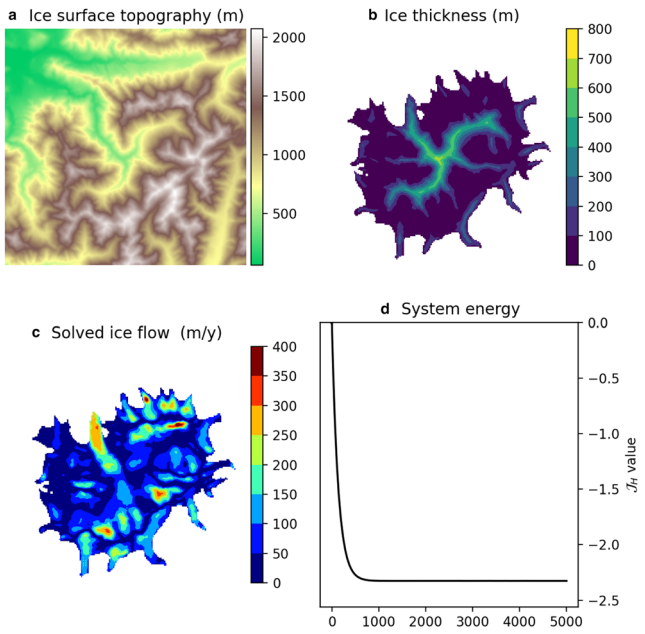


Figure A2. Results of the solver on the 'test' glacier: (A) ice surface topography and (B) ice thickness of the 'test' glacier; (C) 'solved' surface ice-flow solution at convergence; (D) evolution of the system energy through the iterations of the Adam optimiser.

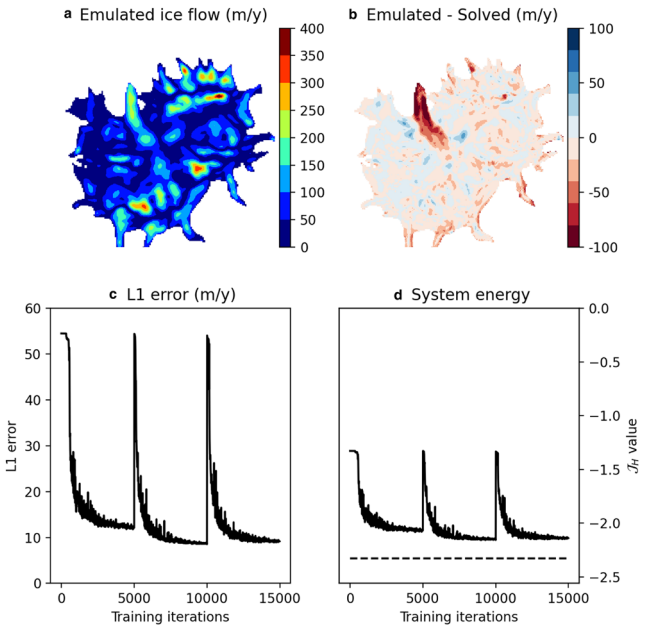


Figure A3. Results of the emulator on the 'test' glacier: (A) 'Emulated' surface ice flow at the surface of the test glacier (Fig. A2) at convergence of the offline training over the catalogue, (B) difference between the 'emulated' and 'solved' solutions, (C) evolution of the $L1$ error between the two solutions and (D) of the system energy through the training epochs. The jumps each 5000 iterations are due to the re-initialization of the learning rate.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

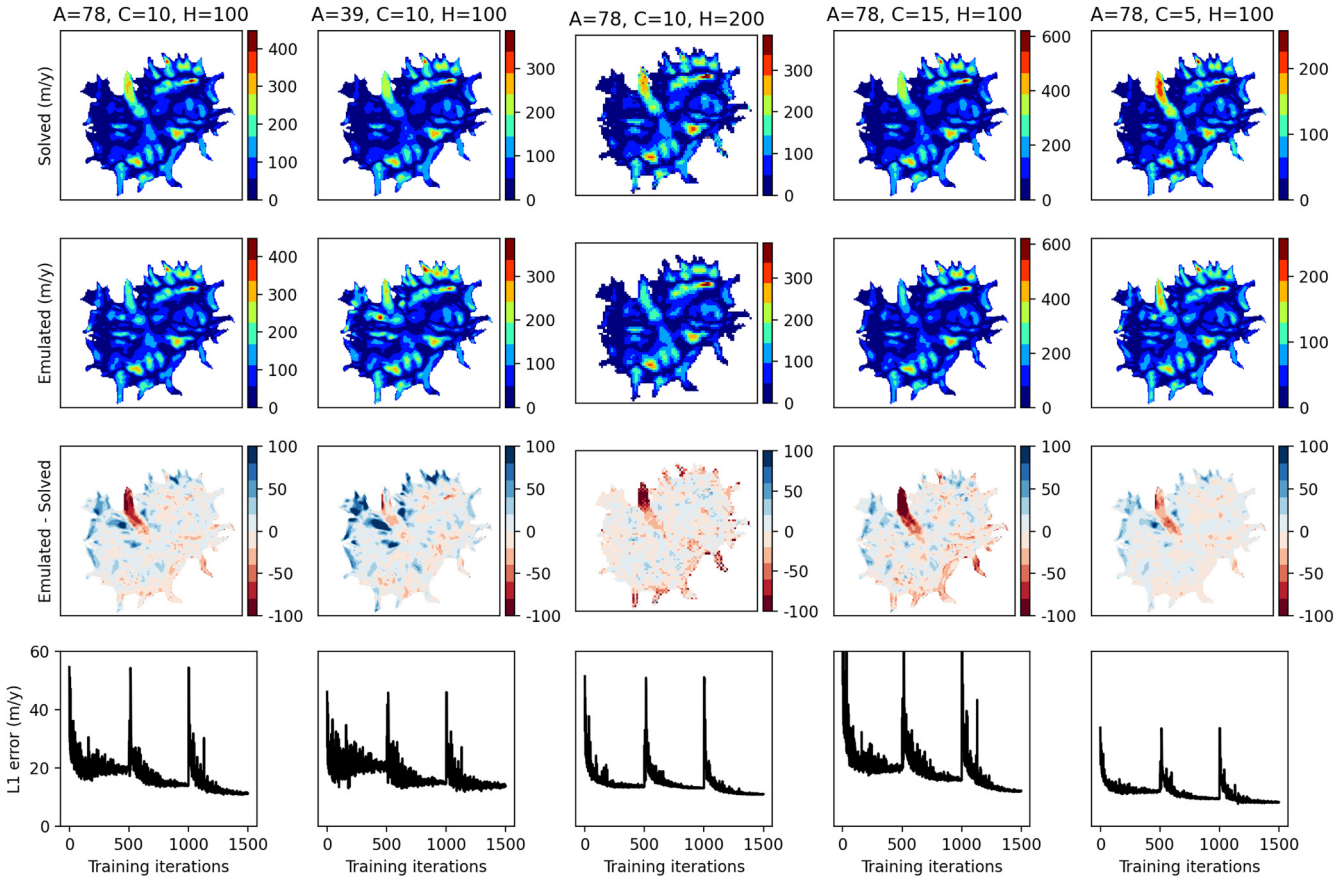


Figure A4. Results of the emulator on the 'test' glacier with varying values of A, c and H. Each column corresponds to one parameter set (A, c, H) (the first column shows the default original parameters). The first row displays the 'solved' surface ice-flow solution. The second row displays the 'emulated' solution after training over the glacier catalogue, while the third shows the difference between this solution and the 'solved' one. The last raw shows the $L1$ error through the training. The jumps each 5000 iterations are due to the re-initialization of the learning rate.

In a second experiment, we take over the emulator trained with fixed values of A, c and H, and augment the training data by sampling additional values (but spatially constant) for $A \in [20,\ 100]$ MPa^{-3} a^{-1}, $c \in [0,\ 20]$ km MPa^{-3} a^{-1} and training at a different resolution $H = 100,\ 200$ m. The ice-flow parameters (A, c) were sampled with a uniform distribution within their ranges, while the spatial resolution H_H (initially 100 m) was randomly changed to 200 m by simple data upscaling. As a result, the CNN meets a large set of input parameters in terms of glacier shape (sampling into the catalogue as before) and other parameters. To assess the performance of the emulator, we compare 'emulated' and 'solved' solutions obtained with five sets of parameters (A, c, H) for the test glacier in Figure A4. As a result, the emulator generally captures roughly the ice flow for various parameter sets (compare the first and second rows of Fig. A4). However, we find relatively high spatial discrepancies when displaying the difference between the two (third row of Fig. A4), with $L1$ errors between 10 and 20 m a^{-1}. Such a deteriorated accuracy is not surprising: the storage capacity of our CNN model

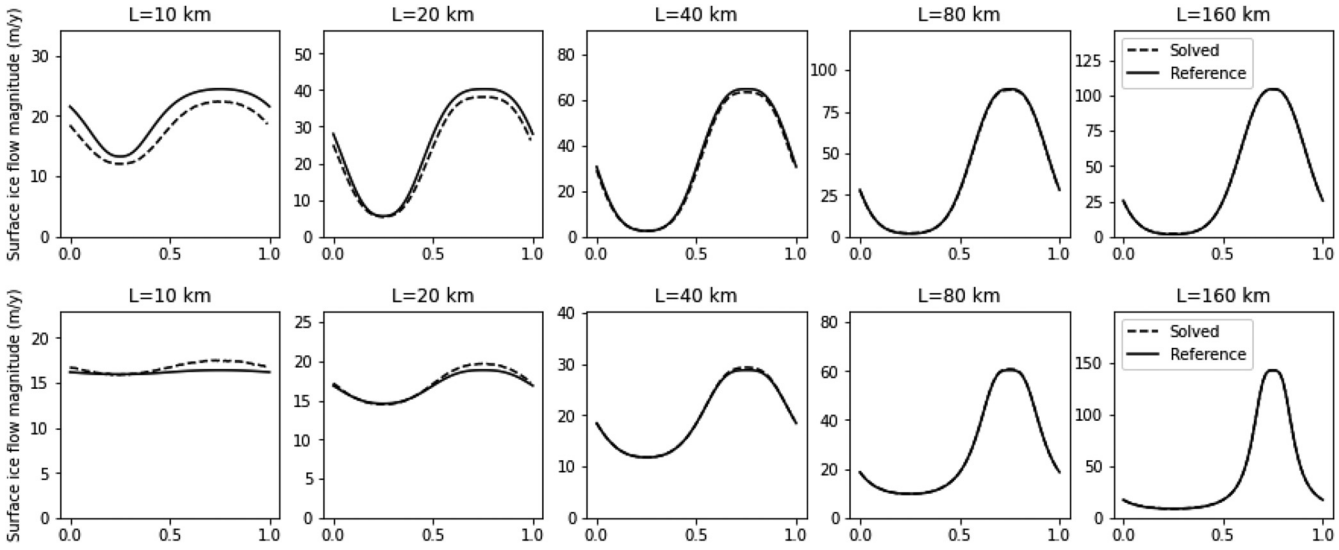


Figure B5. Surface ice-flow magnitude along the $y = L/4$ horizontal line for different length scales L = 10, 20, 40, 80 and 160 km in the ISMIP-HOM experiments A and C: comparison between 'solved' with reference solution 'oga1' obtained from Pattyn (2008). For simplicity, the x-axis was scaled with L.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press

emulator has reached its limit, and one cannot expect a model of a given size (about 140 000 parameters) to store more realisations with similar accuracy.

APPENDIX B. ISMIP-HOM validation solutions

ISMIP-HOM (Pattyn, 2008) experiments consist of modelling exercises based on various synthetic ice geometries and boundary conditions to produce different types of ice flow, which can be met in real glacier modelling. Here, we focus on ISMIP-HOM experiments A and C, which represent a wide panel of various 3D ice-flow scenarios (from shearing to sliding-dominant flows) over a square horizontal domain of length $L > 0$: $\Omega = [0, L] \times [0, L]$. In experiment A, the ice geometry is defined by

$$s(x, y) = -x \tan(0.5^\circ),$$
$$b(x, y) = s(x) - 1000 + 500 \sin(2\pi x/L) \sin(2\pi y/L),$$

and a no-slip condition is prescribed on the bedrock, while, in experiment C, the geometry is defined by

$$s(x, y) = -x \tan(0.1^\circ),$$
$$b(x, y) = s(x, y) - 1000,$$

and a slip condition is prescribed everywhere on the bedrock defined by $m = 1$ and

$$c(x, y) = [1000 \times (1 + \sin(2\pi x/L) \sin(2\pi y/L))]^{-1}.$$

In both experiments, we use $A = 100\ \mathrm{MPa}^{-3}\ \mathrm{a}^{-1}$ as Arrhenius factor in Glen flow law, and horizontal periodic boundary conditions connect the four horizontal sides of Ω, see Pattyn (2008) for further details. The squared horizontal domain Ω was divided into 100 cells in both horizontal directions to generate a regular grid, while the ice thickness is divided into 20 layers. To obtain a wide range of aspect ratios, we performed both experiments for several values of domain length L = 10, 20, 40, 80 and 160 km. Figure B5 compares the 'solved' solutions at convergence with the reference 'oga1' solution obtained from Pattyn (2008) for all experiments.

As a result, we generally find a very good agreement between the two solutions. In line with model intercomparisons (Pattyn, 2008), there are small discrepancies in the experiments that have the smallest domain length L, which are known to be more sensitive to numerical parameters and schemes. This validates our numerical solver and verifies that the system energy (18) – which is used for solving and training the CNN – is correctly implemented.

https://doi.org/10.1017/jog.2023.73 Published online by Cambridge University Press