Concepts and capabilities of the Instructed Glacier Model (3.X.X)

Guillaume Jouvet[1], **Brandon Finley**[1], **and other authors**[1]

[1]Université de Lausanne, Switzerland

This document refer to IGM v3.X.X. Check the tag v.2.2.2 of this repo for the older release v.2.2.2.

This working document intends to be submitted for publication in a model development journal. It is opened for contributions. Contributions typically include the development of modules, or the performing and reporting of numerical experiments that assess the capabilities of IGM. All contributors will be listed as co-authors of the final paper. To date, direct contributors are (listed alphabetically): Samuel Cook, Guillaume Cordonnier, Andreas Henz, Oskar Herrmann, Tancrède Leger, Fabien Maussion, Jürgen Mey, Dirk Scherler, Claire-Mathile Stücki, Ethan Welty. Contact G. Jouvet if you wish to contribute.

Abstract. We present the concept and capabilities of the Instructed Glacier Model (IGM, https://github.com/jouvetg/igm), a Python-based modeling tool designed for efficiently simulating glacier evolution across various scales. For that purpose, IGM couples high-order ice thermomechanics, climate-driven surface mass balance, and mass conservation. Within IGM, the update of all physical model components involves a series of fully-rasterized mathematical operations performed by the TensorFlow library. This choice results in high parallelization capabilities, particularly beneficial when executed on GPU hardware. The specificity of IGM is that it models ice flow with a physics-informed convolutional neural network trained from high-order ice flow physics. Beside rasterizing its solving, this approach advantagously comes with automatic differentiation, which strongly facilitates model inversion (or data assimilation). IGM's design is focused on i) accessibility to a large community of glaciologists, ii) modularity to facilitate community development and customization, iii) reproducibility and large ensemble simulation capability, and iv) compatibility with OGGM for data access. We present IGM's physical and numerical models, the implementation and usage with a comprehensive workflow enabling the quick modeling of any mountain glacier based on the RGI ID. An ensemble of benchmark simulations demonstrates the physical and computational capabilities of IGM to simulate the evolution of paleo and contemporary glaciers.

1 Introduction

Glacier evolution models permit to reconstruct the historical behavior of glaciers and their relationship with past climates. Additionally, they are indispensable for predicting how glaciers will evolve in the future and the consequent rise in sea levels due to climate warming (Pattyn, 2018). Over the last two decades, the glaciological community has dedicated substantial efforts to the development of these models (See the review by Zekollari et al., 2022). These models are designed to encompass a wide range of pertinent physical processes, including ice flow, thermodynamics, subglacial hydrology, and their intricate interactions with various factors such as atmospheric conditions (e.g., climate-driven surface mass balance), the Earth's lithosphere, and the ocean (e.g., iceberg calving or subaquatic melting). Prominent examples of such models include Full-Stokes such as Elmer/Ice (Gagliardini et al., 2013) for a generic usage, high-order such that PISM (Winkelmann et al., 2011), CISM (Lipscomb et al., 2019), ISSM (Larour et al., 2012), which are popular in the ice sheet modelling community, and SIA-based such as OGGM (Maussion et al., 2019) or PyGEM (Rounce et al., 2020), which were designed for global glacier modelling. However, the increasing complexity of models (based on high-order mechanics) as well as the significant increase of observational data to assimilate comes with rising computational burdens. Parallel computing as well as automatic differentiation sounds like a promising way to overcome these limitations.

[GJ: UPT WITH MOST RECENT LITTERATURE, ESPPI-ML] In recent years, there has been a growing interest in employing Graphics Processing Units (GPUs) to tackle the computational bootleneck in ice flow modeling. GPUs are equipped with a larger number of cores, albeit at slower speeds compared to Central Processing Units (CPUs). GPUs have the potential of overcoming previously mentioned limitations in modeling ice flow and achieve substantial speed improvements (Räss et al., 2020). Effectively harnessing the power of GPUs hinges on the implementation of numerical methods that can be subdivided into numerous parallel tasks, a particularly challenging endeavor when dealing with the viscous behavior of ice and the underlying diffusion equations governing its motion. A natural approach involves the explicit time integration of the Shallow Ice Approximation (SIA) (Višnjević et al., 2020), the Second Order SIA (Brædstrup et al., 2014), or the Stokes model (Räss et al., 2020). While programming on GPU was a relative complex task in the past, the emergence of libraries such as TensorFlow and PyTorch in the popular Python language opens new opportunities for the development of efficient glacier ice flow model at relatively limited technical level.

Capitalizing on recent library for efficient computation on GPU and machine learning techniques, we outline the concept and demonstrate the capability of a Python-based glacier evolution model – the Instructed Glacier Model (IGM) – originally introduced by Jouvet et al. (2022) – which couples ice thermomechanics, surface mass balance, and mass conservation. The specificity of IGM is that all physical model components are updated using relatively short sequence of operations on horizontal raster grids, making the workflow efficiently parrallelizable on GPU. Model components that do not involve any horizontal diffusion (such as the surface mass balance or the Enthalpy models) are easily solved raster-wise. In contrast, we use a Convolutional Neural Network (CNN) trained to satisfy high-order ice flow physics (Jouvet and Cordonnier, 2023) to ensure the parrallization of this task. Using a CNN to model the ice flow has an other major advantage for data assimilation as embedded automatic differentiation tools permits to access all their derivatives, and therefore to perform model inversion (Jouvet, 2023). Lastly, IGM is implemented in the widely-used programming language, Python, to make it accessible to a large community of glaciologists and leverage the numerous Python libraries such as TensorFlow for efficient computation on GPU, OGGM for data access, and Hydra for parameter handling. Additionally, we have designed IGM in a modular fashion to facilitate community development and user customization of the model.

This paper is organized as follows. First, we describe the physical and numerical models and implemented in IGM. Then, we describe the implementation and the usage. Last, we demonstrate its capabilities by presenting some examples of applications.

2 Model

In the following, we use the notations $b(x,y)$, $s(x,y,t)$, and $h(x,y,t)$ to represent the glacier bedrock, upper surface, and ice thickness (Fig. 1). Here, (x,y) and t denote horizontal coordinates, and the time. We also introduce $\mathbf{u}(x,y,z,t) = (u_x, u_y, u_z)$ as the 3D velocity field of the ice, and $T(x,y,z,t)$ and $\omega(x,y,z,t)$ to represent temperature and water content, respectively.

2.1 Forward physical modelling

Here, we describe all the physical models that are included in IGM and govern the evolution of the above-defined variables (Fig. 1). Note that some model components (e.g., the Enthalpy) may be ignored for simplicity, while some others (e.g., the mass conservation) are always included.

2.1.1 Surface mass balance

IGM comes with several Surface Mass Balance (SMB) models, which compute the mass balance at the glacier surface. The SMB is the sum of the surface accumulation and ablation over one year. The SMB is computed at each grid cell of the horizontal domain and is used to update the ice thickness (Section 2.1.3). IGM includes three SMB models:

- The `smb_simple` one implements a classical linar relation between Equilibrium Line Altitude (ELA) and altitude:

$$SMB(z) = \min(\beta_{acc}(z - z_{ELA}), m_{acc}) \quad \text{if } z > z_{ELA}, \tag{1}$$

$$SMB(z) = \beta_{abl}(z - z_{ELA}) \quad \text{else}, \tag{2}$$

  where z_{ELA} is the ELA, β_{abl} and β_{acc} are ablation and accumulation gradients, and m_{acc} is the maximum SMB.

- The `smb_oggm` implements a combined accumulation / Positive Degree-Day (PDD) (Hock, 2003) to calculate the SMB based on seasonal temperature and precipitation fields. In this model, surface accumulation equals solid precipitation when the temperature is below a threshold (usually 0°C) and decreases linearly to zero in a transition zone. Conversely, surface ablation is computed in proportion to the number of PDDs. Given monthly temperature T_i and precipitation P_i spatial fields, the yearly SMB at elevation z is then computed with

$$SMB = \frac{\rho_w}{\rho_i} \sum_{i=1}^{12} \left(P_i^{sol} - d_f \max\{T_i - T_{melt}, 0\} \right), \tag{3}$$

  where P_i^{sol} is the monthly solid precipitation, T_i is the monthly temperature and T_{melt} is the air temperature

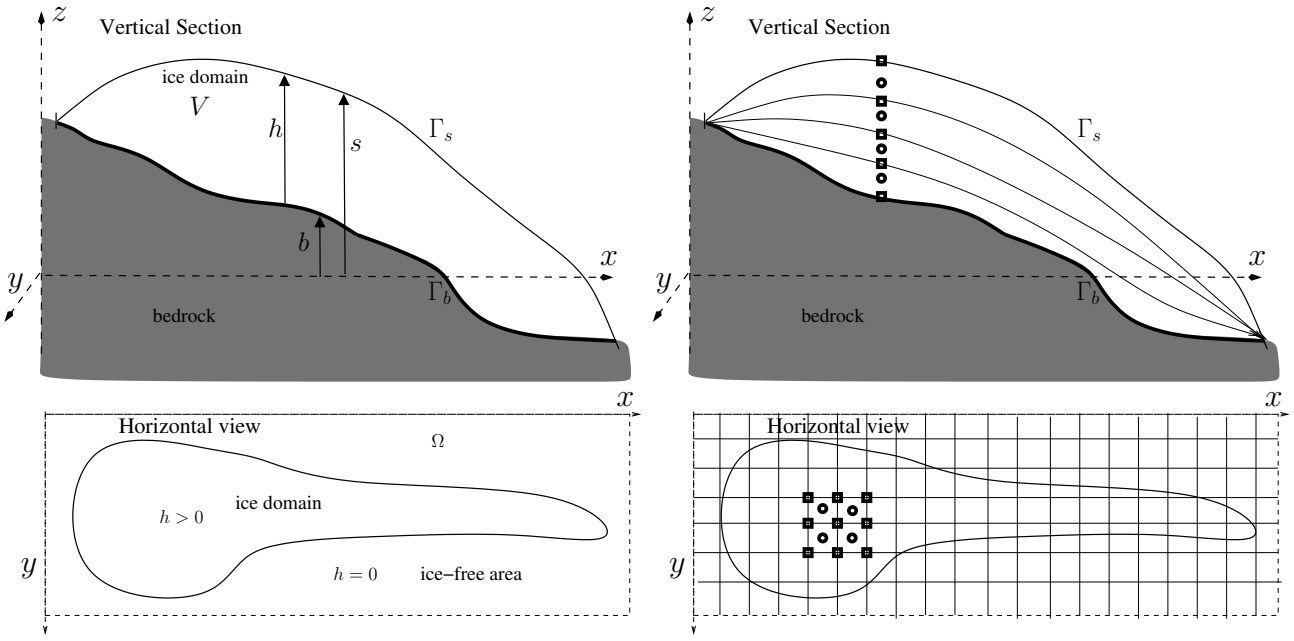

Figure 1. Cross-section and horizontal view of a glacier with notations (left panel) and its spatial discretization (right panel), which is obtained using a regular horizontal grid and by subdividing the glacier into a pile of layers. All modelled variables (e.g. ice thickness) are computed at the corners 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 cells and layers (vizualized with circles).

above which ice melt is assumed to occur, d_f is the melt factor, and $\frac{\rho_w}{\rho_i}$ is the ratio between water and ice density.

– The `smb_accme` implements an accumulation scheme similar to the `smb_oggm`, but a more elaborate PDD model (cf. Hock, 2003): The surface ablation is computed proportionally to the number of PDD, however, we track the snow layer depth, and different PDD proportionality factors or ice. The PDD integral is numerically approximated using week-long sub-intervals based on Calov and Greve (2005). Also, a fraction of the melt is assumed to refreeze.

2.1.2 Ice flow

The momentum conservation equation (assuming negligible inertial terms) and the incompressibility condition are expressed as follows:

$$-\nabla\cdot\sigma=\rho\mathbf{g}, \tag{4}$$

$$\nabla\cdot\mathbf{u}=0, \tag{5}$$

where σ is the Cauchy stress tensor, $\mathbf{g}=(0,0,-g)$, g is the gravitational constant. Let τ be the deviatoric stress tensor defined by

$$\sigma=\tau-pI, \tag{6}$$

where I is the identity tensor, p is the pressure field, with the requirement that $\mathrm{tr}(\tau)=0$ so that $p=-(1/3)\mathrm{tr}(\sigma)$. Glen's flow law (Glen, 1953), which characterizes the mechanical behavior of ice, can be formulated as the following nonlinear relationship:

$$\tau=2\mu D(\mathbf{u}), \tag{7}$$

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

$$D(\mathbf{u})=\frac{1}{2}(\nabla\mathbf{u}+\nabla\mathbf{u}^T), \tag{8}$$

μ is the viscosity defined by

$$\mu=\frac{1}{2}A^{-\frac{1}{n}}|D(\mathbf{u})|^{\frac{1}{n}-1}, \tag{9}$$

where $|Y|:=\sqrt{(Y:Y)/2}$ denotes the norm associated with the scalar product $(:)$ (the sum of the element-wise product), $A=A(x,y,z,t)$ is the Arrhenius factor and $n>1$ is the Glen's exponent. Note that A depends on the temperature of the ice (Paterson, 1994) *via* Glen-Paterson-Budd-Lliboutry-Duval law:

$$A(T,\omega)=A_c(T)(1+181.25\omega), \tag{10}$$

where $A_c(T)$ is given by the Paterson-Budd law:

$$A_c(T)=A\exp(-Q/(RT_{pa})), \tag{11}$$

where A and Q have different values below and above a threshold temperature:

$$A = 3.985 \times 10^{-13}\, s^{-1} Pa^{-3}, \quad \text{if } T < 263.15\, K \tag{12}$$

$$A = 1.916 \times 10^{3}\, s^{-1} Pa^{-3}, \quad \text{else.} \tag{13}$$

and

$$Q = 60\ \mathrm{kJmol}^{-1}, \quad \text{if } T < 263.15\, K \tag{14}$$

$$Q = 139\ \mathrm{kJmol}^{-1}, \quad \text{else.} \tag{15}$$

We may ignore temperature dependence of A for instance in the case the Enthalpy (Section 2.1.4) is not modelled.

Equations (4) to (9) describe a Stokes problem in which the unknowns are the 3D velocity field $\mathbf{u}$ and the pressure field p. To simplify the problem, we make the "hydrostatic assumption" as described by (Blatter, 1995) and neglect second-order terms in the aspect ratio of the ice domain (thickness versus length) within the strain rate tensor $D(\mathbf{u})$. By doing so and invoking the incompressibility equation, both the vertical velocity components u_z and the pressure p are eliminated from the momentum conservation equation. The resulting model, commonly referred to as the Blatter-Pattyn model (Blatter, 1995), conveniently transforms into a 3D nonlinear elliptic equation solely for the horizontal velocity components. This modification makes it easier to solve compared to the original Stokes model.

The boundary conditions that supplement (4), (5) are the following. Stress free force applies to the ice-air interface,

$$\sigma \cdot \mathbf{n} = 0, \quad p = 0, \tag{16}$$

where $\mathbf{n}$ is an outer normal vector along the surface. Along the lower surface interface, the nonlinear Weertman friction condition (e.g., Schoof and Hewitt, 2013) relates the basal shear stress τ_b to the sliding velocity $\mathbf{u}_b$ as follows:

$$\mathbf{u} \cdot \mathbf{n} = 0, \tag{17}$$

$$\tau_b = -c|\mathbf{u}_b|^{m-1}\mathbf{u}_b, \tag{18}$$

where $m > 0$, $c = c(x, y) > 0$, and $\mathbf{n}$ is the outward normal unit vector to the bedrock.

The sliding coefficient c in (18) is defined with the Mohr-Coulomb law (Cuffey and Paterson, 2010), that involves the effective pressure in the till N_{till} (Section 2.1.5):

$$c = \tau_c u_{th}^{-m} = N_{till} \tan(\phi) u_{th}^{-m}, \tag{19}$$

where ϕ is the till friction angle, and u_{th} is a parameter homegenous to ice velocity following Khroulev and the PISM Authors (2020). In the case the Enthalpy (Section 2.1.4) and the subglacial hydrology (Section 2.1.5) are not modelled, we may simply prescribe a constant sliding coefficient c, or get its spatial distribution by inverse modelling (Section 2.3).

Conditions on the top surface are free-stress:

$$A^{-\frac{1}{n}} |D(\mathbf{u})|^{\frac{1}{n}-1} D(\mathbf{u}) \cdot \mathbf{n} = 0. \tag{20}$$

[GJ:PROVIDE THE DERIVATION BLATTER/ENERGY IN APPENDIX] Following Jouvet (2016), the Blatter-Pattyn problem is rewritten as an energy-minimization problem associated with the functional:

$$\mathcal{J}(\mathbf{v}) = \int_V 2\frac{A^{-\frac{1}{n}}}{1+\frac{1}{n}} |D(\mathbf{v})|^{1+\frac{1}{n}} dV + \int_{\Gamma_b} \frac{c}{1+m} |\mathbf{v}|_M^{1+m} dS + \rho g \int_V (\nabla s \cdot \mathbf{v}) dV, \tag{21}$$

where V, and Γ_b denote the ice volume and bedrock interface.

Note that while Blatter-Pattyn model permits to obtain the horizontal components of the ice velocity (u_x, u_y), the third and vertical component u_z can be obtained by integrating vertically the imcompressibility condition (5).

2.1.3 Mass conservation

The evolution of ice thickness, denoted as $h(x, y, t)$, starting from an initial glacier shape, is governed by mass conservation, which connects elevation change, ice dynamics and SMB (Fig. 1) through:

$$\frac{\partial h}{\partial t} + \nabla \cdot (\bar{\mathbf{u}} h) = \mathrm{SMB}, \tag{22}$$

where symbol $\nabla\cdot$ represents the divergence operator for the horizontal variables (x, y), $\bar{\mathbf{u}} = (\bar{u}, \bar{v})$ denotes the vertically-averaged horizontal ice velocity field, while SMB represents the surface mass balance.

2.1.4 Ice enthalpy

In this section, we model ice enthalpy following Aschwanden et al. (2012). This approach enables us to simultaneously model ice temperature and water content when the temperature reaches the pressure melting point, thereby conserving energy. Ice enthalpy influences the dynamical model in two ways: variations in temperature and water content lead to ice softening or hardening (Eq. (10)), while enthalpy affects basal sliding conditions throught basal till water layer (Eq. (19)). The enthalpy, denoted as E, is a variable defined throughout the ice and is a function of both temperature, T, and water content, ω:

$$E(T, \omega, p) = \begin{cases} c_i(T - T_{\mathrm{ref}}), & \text{if } T < T_{\mathrm{pmp}}, \\ E_{\mathrm{pmp}} + L\omega, & \text{if } T = T_{\mathrm{pmp}},\ 0 \le \omega, \end{cases} \tag{23}$$

where c_i is the heat capacity, T_{ref} is a reference temperature, and L is the latent heat of fusion. Additionally the temperature T_{pmp} and enthalpy E_{pmp} at pressure-melting point of

ice are defined by

$$T_{\rm pmp} = T_0 - \beta p, \tag{24}$$
$$E_{\rm pmp} = c_i(T_{\rm pmp}(p) - T_{\rm ref}), \tag{25}$$

where $T_0 = 273.15$ K is the melting temperature at standard pressure, and $\beta = 7.9 \times 10^{-8}$ K Pa^{-1} is the Clausius-Clapeyron constant.

According to the definition of enthalpy provided above, we have two possible modes: i) When the ice is cold, meaning it is below the melting point, the enthalpy is simply proportional to the temperature minus a reference temperature. ii) When the ice is temperate, the enthalpy continues to increase. In this case, the additional component $L\omega$ accounts for the creation of water content through energy transfer. Therefore, one can infer the value of enthalpy, denoted as E, from both temperature, T, and water content, ω, and *vice-versa*.

The melting point temperature at pressure is adjusted for hydrostatic pressure $p = \rho g d$ using the following equation:

$$T_{pmp} = T_0 - \beta \rho g d, \tag{26}$$

where d represents the depth. Therefore, the "pressure-adjusted" temperature, denoted as T_{pa}, is defined as the temperature with a shift such that its melting point temperature reference is always zero:

$$T_{pa} = T + \beta \rho g z.$$

The enthalpy model consists of the following advection-diffusion equation, with horizontal diffusion being neglected:

$$\rho_i \left(\frac{\partial E}{\partial t} + u_x \frac{\partial E}{\partial x} + u_y \frac{\partial E}{\partial y} + u_z \frac{\partial E}{\partial z} \right) - \frac{\partial}{\partial z} \left(K_{c,t} \frac{\partial E}{\partial z} \right) \tag{27}$$
$$= \phi - \rho_w L D_w(\omega), \tag{28}$$

where ρ_i is the ice density, $K_{c,t}$ equals $K_c = k_i/c_i$ if the ice is cold ($E < E_{pmp}$) or $K_t = \epsilon k_i/c_i$ otherwise. Using Glen's flow law (Eq. (7)), the strain heating ϕ is defined by

$$\phi = D(\mathbf{U})\tau = A^{-1/n} |D(\mathbf{u})|^{1+1/n}. \tag{29}$$

The last source term $-\rho_w L D_w(\omega)$ in (28) permits to remove the water in temperate ice, $D_w(\omega)$ being a drainage function (Aschwanden et al., 2012).

At the top ice surface, the enthalpy equation is constrained by the surface temperature (or equivalently, the enthalpy) provided by the climate forcing, which is enforced as a Dirichlet condition. At the glacier bed, there are multiple boundary conditions for the enthalpy equation (Aschwanden et al., 2012; Wang et al., 2020):

- cold base and dry: $K_c \frac{\partial E}{\partial z} = Q_{\rm geo} + Q_{\rm fh}$ if $E_b < E_{\rm pmp}$ and $W_{till} = 0$,
- cold base and wet: $E_b = E_{\rm pmp}$ if $E_b < E_{\rm pmp}$ and $W_{till} > 0$,
- temperate base and cold ice: $E_b = E_{\rm pmp}$ if $E_b \geq E_{\rm pmp}$ and $W_{till} > 0$, zero temperate basal layer,
- temperate base, temperate ice: $K_t \frac{\partial E}{\partial z} = 0$ if $E_b \geq E_{\rm pmp}$ and $W_{till} > 0$, non-zero temperate basal layer,

where $Q_{\rm geo}$ and $Q_{\rm fh}$ are the geothermal heat flux, and the frictional heat flux, respectively. Using Weertmann law (18), the latter is computed as follows:

$$Q_{\rm fh} = \tau_b \cdot \mathbf{u}_b = c|\mathbf{u}_b|^{m+1}. \tag{30}$$

When the temperature hits the pressure-melting point at the glacier bed (i.e. $E \geq E_{\rm pmp}$), the basal melt rate is calculated via the following equation:

$$m_b = \frac{1}{\rho_i L}(Q_{fr} + Q_{geo} - K_{t,c}\frac{\partial E}{\partial z}). \tag{31}$$

The basal melt rate is further increased to account for the drainage of the water content generated throughout the entire column (last term of Eq. (28)).

2.1.5 Subglacial hydrology

Following Bueler and van Pelt (2015), the basal water thickness in the till W_{till} is computed from the basal melt rate as follows:

$$\frac{\partial W_{till}}{\partial z} = \frac{m_b}{\rho_w} - C_{dr}, \tag{32}$$

where C_{dr} is a simple drainage parameter. The till is assumed to be saturated when it reaches the value $W_{till}^{max} = 2$ m, therefore, the till water thickness is capped to this value. The effective thickness of water within the till N_{till} is computed from the saturation ratio $s = W_{till}/W_{till}^{max}$ by the formula (Bueler and van Pelt, 2015):

$$N_{till} = \min\left\{ p, N_0 \left(\frac{\delta P}{N_0} \right)^s 10^{(e_0/C_c)(1-s)} \right\}, \tag{33}$$

where p is the ice overburden pressure and the remaining parameters are constant.

2.1.6 Lagrangian modelling

While IGM's models are formulated in a Eulerian framework, IGM also includes a Lagrangian module to track the trajectory of (physical or virtual) particles advected by the ice flow. Doing so is particularly useful for studying the evolution of tracers within the ice such as debris, morainic material, or simply to track the age and properties of ice. The trajectory of the particle passing $\bar{\mathbf{x}}$ at time $\bar{t}$ on the time interval $I_{\bar{t}}$ solves the ordinary differential equation:

$$\begin{cases} \frac{d\mathbf{x}}{dt}(t) & = \mathbf{u}(\mathbf{x}(t), t), \qquad \text{on } I_{\bar{t}}, \\ \mathbf{x}(\bar{t}) & = \bar{\mathbf{x}}. \end{cases} \tag{34}$$

2.2 Forward numerical modelling

In IGM, the horizontal modeled domain is assumed to be a rectangle. IGM deals with rastered data defined on a regular grid of dimensions $N_x \times N_y$ and uniform spacing along both the x and y axes (Fig. 1, right panel). Key variables such as ice thickness h, surface topography s, or sliding coefficient c are defined on this grid. It is important to note that our choice of a structured grid, rather than any other type of more complex discretization, is crucial for representing variables as 2D arrays. This structure allows us to employ Convolutional Neural Networks (CNN) for emulating the mechanics of ice flow (Section 3.2.3). On the other hand, the discretization of ice thickness occurs vertically using a fixed number of points denoted as N_z (Fig. 1, left panel). These layers can be distributed in a non-uniform manner, e.g. to ensure finer discretization near the ice-bedrock interface, where the steepest gradients are expected, and coarser near the ice-surface interface following the strategy proposed in the Parallel Ice Sheet Model (PISM, Khroulev and the PISM Authors, 2020). Note that in the special case of $Nz = 2$, the ice velocity profile from the bottom to the top of the ice is assumed to vary polynomially following the Shallow Ice Approximation (SIA) formula similarly to the approach proposed by Dias dos Santos et al. (2022). In the case of a single layer ($Nz = 1$), the ice flow is assumed to be vertically uniform, and the ice flow model reduces to the Shallow Shelf Approximation (SSA).

IGM employs a time-advancing algorithm that permits to update SMB, iceflow, possibility enthalpy, and ice thickness over time. For efficiency reason, the key is to perform all these updates raster-wise with the Tensorflow library, i.e. in parallel over all cells of the horizontal grid, however, making sure that these updates involve a relatively short sequence of operations: in $\mathcal{O}$(CNN's number of layers) for the ice flow, in $\mathcal{O}$(SMB time step) for the SMB, and in $\mathcal{O}(Nz)$ for the enthalpy. For clarity and modularity, IGM separates these steps into distinct modules within the modeling framework.

2.2.1 Surface mass balance

The SMB models presented in Section 2.2.1 are implemented in IGM applying formula (1)-(3) on the entire grid. Therefore the computation of the SMB is fully rasterized and can be computed in parallel on the GPU. Only the PDD model with tracking of the snow layer (`smb_accmelt`, which applies different PDD parameters on snow and ice) needs to be computed sequentially in time, and can therefore not be parallelized. Its execution costs scales therefore linearly with the number of time steps used for the SMB and can therefore be considered as a bottleneck if small time steps are used (typically daily or weekly).

2.2.2 Ice flow

[TODO: ADD THE CASE $Nz = 1, 2$]

Following Jouvet and Cordonnier (2023), the ice flow dynamics in 3D is modelled using a Physics-Informed Convolutional Neural Network (CNN). The CNN predicts horizontal ice flow $(\mathbf{u}_H, \mathbf{v}_H)$ from input fields which includes ice thickness h_H, surface topography s_H, ice flow parameters A_H and sliding coefficient c_H, and spatial grid resolution H_H (Fig. 2). The CNN is trained to minimize the energy (21) associated with the Blatter-Pattyn ice flow model. The optimisation problem is solved using the Adam optimiser (Kingma and Ba, 2014) – the derivatives of the energy with respect to the weights of the CNN being obtained from automatic differentiation.

This approach offers a GPU-accelerated alternative to traditional solvers, along with the ability to memorize previous solutions and take advantage of this for accuracy improvement (Jouvet and Cordonnier, 2023). In IGM, we initially load a pretrained iceflow emulator (to start with a good initial guess) provided with the IGM package, and retrain it regularly within the time loop to maintain its accuracy. This frequent retraining permits the CNN to adapt to the new glacier geometries met during the glacier evolution. Note that the retraining costs are about 3 times more than a CNN evaluation (since it requires to pass over the CNN in the two directions), we strive to find a trade-off frequency: not too frequent to mitigate the costs, and not to rare to maintain the accuracy of the emulator. It should be stressed that the retraining task is memory consuming since it involves storing all the operations of the CNN. Therefore, the maximal size of the raster grid that can be used with IGM is limited by the available GPU memory, otherwise a more costly strategy should be used to retrain sequentially and patch-wise the CNN.

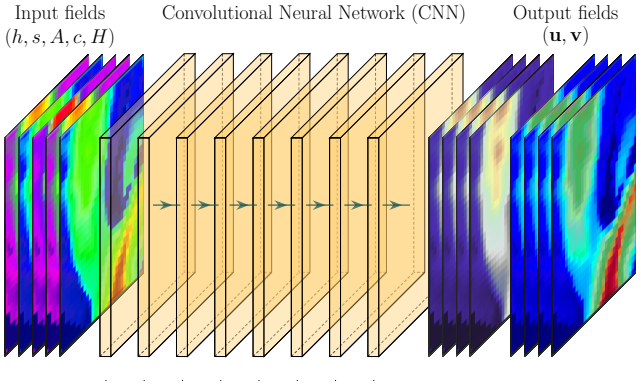

Figure 2. Our iceflow emulator consists of a CNN that maps geometrical (thickness and surface topography), ice flow parameters (shearing and basal sliding), and spatial resolution inputs to the horizontal ice flow field in 3D.

Should we require the vertical ice flow component (as required by the Enthalpy or the particle tracking), its computation is achieved by integrating numerically the incom-

pressibility condition (5) over the vertical discretization layers (Fig. 1, right pannel) from the bottom to the top of the ice.

2.2.3 Mass conservation

Solving mass conservation equation (22) permits to update ice thickness from the iceflow and the SMB. This is done using an explicit first-order upwind finite-volume scheme on the 2D working grid. In this scheme, ice mass is permitted to move between adjacent cells where thickness and velocities are defined. This movement is determined by edge-defined fluxes, which are inferred from depth-averaged velocities and ice thickness in the upwind direction. The resulting scheme is both mass-conserving and parallelizable, thanks to its fully explicit nature. However, it is subject to a Courant-Friedrichs-Lewy (CFL) condition, yielding to an upper-bound on the time step. This condition guarantees that no more than the entire content of one cell is transferred to a neighboring cell within a single iteration.

The time step is therefore adjusted to remain below a designated maximum time step:

$$\Delta t \leq C \times \frac{\Delta x}{\|\bar{\mathbf{u}}\|_{L^\infty}}, \tag{35}$$

where $C < 1$ and Δx is the grid cell spacing. This CFL condition is key to ensure stability in the transport scheme for ice thickness evolution.

2.2.4 Ice enthalpy and subglacial hydrology

The 3D enthalpy field needs to be updated based on the mechanical state by solving the advection-diffusion equation (27)-(28). This process includes applying the top and bottom boundary conditions described in Section 2.1.4. To achieve this, the variable E (and equivalently, T and ω) is defined on the same 3D structured grid than the ice flow field $\mathbf{u}$. [TODO: CHANGE THIS, THE ENTALPY MUST HAVE ITS OWN GRID]. Solving equations (27)-(28) is done in two steps through time splitting operators. First, an explicit first-order upwind finite-volume scheme is employed to solve the advective component in the horizontal direction, following a similar scheme as the one used in the mass conservation equation:

$$E^{n+\frac{1}{2}} = E^n - dt\left(u_x^n \frac{\partial E^n}{\partial x} + u_y^n \frac{\partial E^n}{\partial y}\right), \tag{36}$$

In a second time we solve the remaining advection-diffusion equation with respect to the vertical direction in an implicit manner:

$$\rho_i \left(\frac{E^{n+1} - E^{n+\frac{1}{2}}}{dt} + u_z \frac{\partial E^{n+1}}{\partial z}\right) \tag{37}$$

$$- \frac{\partial}{\partial z}\left(K_{c,t} \frac{\partial E^{n+1}}{\partial z}\right) = \phi^n - \rho_w L D_w(\omega^{n+\frac{1}{2}}). \tag{38}$$

This is done by finite differences, where the advection term is approximated using an upwind method, while the diffusion term is approximated using a centered scheme. Once the top and bottom boundary conditions are incorporated, the discretization leads to solving $N_y \times N_x$ tridiagonal systems, each with a size equal to the number of layers N_z. Solving these tridiagonal systems is accomplished using the tridiagonal matrix algorithm (or Thomas algorithm). This algorithm requires approximately $3 \times N_z$ operations per column, translating to a total of $3 \times N_z \times N_y \times N_x$ operations in total. Tensorflow ensures parallelism of operations between column-wise problems. Following solving the enthalpy equation, the process involves computing the basal melt rate using (31), updating it along with the enthalpy to account for water drainage along the ice column. This step also calculates the resulting water thickness from (32) and sliding coefficient from (19).

In summary, assuming known enthalpy E^n, ice thickness geometry h^{n+1}, iceflow $\mathbf{u}^{n+1}$, vertical ice flow u_z^{n+1}, updating the enthalpy at time t^{n+1} requires to perform the following sub-steps:

- Compute the mean surface temperature T_s^n to enforce the upper surface Dirichlet boundary condition. This temperature is capped at 0°C to maintain the temperature of ice below pressure-melting point.
- Compute the vertical discretization with the new ice geometry h^{n+1}.
- Compute the temperature T_{pmp}^n and enthalpy E_{pmp}^n at pressure melting point using (26).
- Compute the ice temperature T^n from the enthalpy E^n using (23).
- Compute the Arrhenius factor $A(T^n)$ from temperature T^n using (12)-(13).
- Compute the 3D strain heat ϕ^{n+1} from ice flow field $\mathbf{u}^{n+1}$ and Arrhenius factor $A(T^n)$ using (29).
- Compute the 2D basal frictional heat Q_{fh}^{n+1}, from basal velocity field $\mathbf{u}^{n+1}$ and sliding coefficient c^n using (30).
- Compute the surface enthalpy E_s^n from the surface temperature T_s^n using (23).
- Compute the updated enthalpy $E^{n+\frac{1}{2}}$ after solving one explicit step for the horizonal advection (Eq. (36)).
- Compute the updated enthalpy E^{n+1} field solving the advection-diffusion equation (Eq. (38)).
- Compute the basal melt rate from (31).
- Compute the water thickness in the till W^{n+1} solving (32) explicity.
- Compute the sliding parametrization c^{n+1} using (19).

2.2.5 Particle tracking

A particle tracking routine calculates the time-space trajectory of virtual tracers that are advected by the ice flow. Thanks to TensorFlow's GPU implementation, a substantial number of particles can be efficiently computed in parallel. For that purpose, one uses a fourth order Runge-Kutta method to solve the ordinary differential equation (34) and obtain an approximation of the trajectory. Presently, there are two implementations available:

- In the `simple` implementation, particles are advected in the horizontal plane using the horizontal velocity field, with interpolation performed bilinearly. In the vertical direction, particles are tracked along the ice column, scaling their position between 0 and 1, where 0 represents the bed and 1 corresponds to the top surface. The evolution of the particles within the ice column over time is determined by the SMB: when the SMB is positive (indicating ice accumulation), the particles move deeper within the ice column, reducing their relative height. Conversely, when the SMB is negative (indicating ice ablation), the particles rise within the ice column, increasing their relative height.
- In the `3d` implementation, the method advects particles from the the 3D ice flow velocity field. Unlike the `simple` method, one needs the vertical ice velocity component to be computed.

2.3 Inverse modelling

Inverse modelling (or data assimilation) serves to find optimal ice thickness, top ice surface, and ice flow parameters that align with observational data in a preliminary step. These observations can include surface ice speeds, ice thickness profiles, and top ice surface data. The goal is to find the fields that best explain the observed data while remaining consistent with the ice flow used in the forward modeling (Jouvet, 2023; Jouvet and Cordonnier, 2023). In the most general case, the corresponding optimization problem consists of finding spatially varying fields (h, c, A, s) that minimize the cost function

$$\mathcal{J}(h,c,A,s) = \mathcal{C}^u + \mathcal{C}^h + \mathcal{C}^s + \mathcal{C}^d + \mathcal{R}^h + \mathcal{R}^c + \mathcal{R}^A + \mathcal{P}^h, \tag{39}$$

where $\mathcal{C}^u$ is the misfit between modeled $\mathbf{u}^s$ and observed $\mathbf{u}^{s,obs}$ surface ice velocities

$$\mathcal{C}^u = \int_\Omega \frac{1}{2\sigma_u^2} \left|\mathbf{u}^{s,obs} - \mathbf{u}^s\right|^2, \tag{40}$$

$\mathcal{C}^h$ is the misfit between modeled and observed h^{obs} ice thickness available profiles:

$$\mathcal{C}^h = \int_\Omega \frac{1}{2\sigma_h^2} |h^{obs} - h|^2, \tag{41}$$

where h^{obs} is a rasterized representation of ice thickness profiles (the pixels with missing data are ignored in the above integral), and $\mathcal{C}^s$ is the misfit between the modeled and observed s^{obs} top ice surface:

$$\mathcal{C}^s = \int_\Omega \frac{1}{2\sigma_s^2} \left|s - s^{obs}\right|^2, \tag{42}$$

where $\mathcal{C}^d$ is a misfit term between the modeled and observed flux divergence d^{obs}:

$$\mathcal{C}^d = \int_\Omega \frac{1}{2\sigma_d^2} \left|\nabla\cdot(h\bar{\mathbf{u}}) - d^{obs}\right|^2, \tag{43}$$

where $\mathcal{R}^h$ is a regularization term to enforce smoothness of $b = s - h$ and convexity of h:

$$\mathcal{R}^h = \alpha_h \int_{h>0} \left(|\nabla b|^2 - \gamma h\right), \tag{44}$$

where $\mathcal{R}^c$ and $\mathcal{R}^A$ are regularization terms to enforce smooth sliding coefficient c and Arrhenius factor A:

$$\mathcal{R}^c = \alpha_c \int_{h>0} |\nabla c|^2, \qquad \mathcal{R}^A = \alpha_A \int_{h>0} |\nabla A|^2, \tag{45}$$

where $\mathcal{P}^h$ is a penalty term to enforce nonnegative ice thickness, and zero thickness outside a given mask:

$$\mathcal{P}^h = 10^{10} \times \left(\int_{h<0} h^2 + \int_{\mathcal{M}^{\text{ice-free}}} h^2 \right). \tag{46}$$

Hereabove, we denote σ_u, σ_h, σ_d, σ_s as the user-defined confidence levels (possibly spatially varying) errors of observations for $\mathbf{u}_s^{obs}$, h_p^{obs}, d^{obs}, and s^{obs}, respectively, and $\alpha_h, \gamma, \alpha_c, \alpha_A > 0$ are fixed parameters.

Solving the optimization problem for the functional (39) is done using the Adam optimizer (Kingma and Ba, 2014), which leverages the automatic differentiation tools provided by the Tensorflow, and the description of the ice flow model as a neural network (Section 3.2.3). In addition, the optimization is combined with retraining of the ice flow emulator to ensure that the ice flow model remains accurate throughout the optimization process. We refer to Jouvet and Cordonnier (2023) for a detailed explanation of the methodology.

Note that the inverse modelling is designed to infer initial distributed sliding coefficient c and/or the ice flow parameters A. It is therefore not designed to be combined with the enthalpy and subglacial hydrology models, which are computing prognostically these fields.

3 Implementation and usage of IGM

IGM is implemented in a Python package, which can be installed using pip directly from the source: https://github.com/instructed-glacier-model/igm or from PyPi:

```
pip install igm-model
```

Once installed, we run IGM by calling the main script `igm_run`, which accepts a wide range of parameters that are passed in a `params.yaml` file stored in folder `experiment` (Fig. 3):

```
igm_run +experiment=params
```

IGM involves running several tasks such as loading geological and climate-related data, initializing fields that describe the glacier's geometry and thermo-mechanical state with a possible optimization procedure, updating these fields through a time iteration loop driven by external forcing, and outputting results at regular time intervals. Recognizing the similarity in the tasks performed by different model components, we have organized IGM in a module-wise fashion. Each module handles a specific aspect of the glacier evolution process, making the model modular and easy to customize: The *inputs* modules serve to the load data (e.g., glacier bedrock, ice surface velocities, ...), the *processes* modules implement physical mechanisms in a decoupled manner (e.g., ice flow, mass conservation, ...), and *outputs* modules that serve to write or plot model results. The main Python script `igm_run` permits to load all *inputs*, *processes*, *outputs* modules (Fig. 4) and their parameters, initialize, update them within a time loop and finalize them.

To handle parameters, IGM relies on the Python library `hydra` (https://hydra.cc/), which can manage complex configurations. In the parameter file `params.yaml` (Fig. 3), the user first define the *inputs*, *processes*, *outputs* modules picked within the pool of IGM modules (with the possibility of adding used-defined ones, see Section 3.4). Then, the user specificy the parameters of each module, which vary from the default values. Note that parameter values passed in the command line will override those provided in the YAML parameter file, while the YAML parameter file, in turn, overrides the default IGM parameters.

The folder organization of IGM experiments is as follows:

- Folder `experiment` contains parameter files.
- Folder `data` contains the input data (if any).
- Folder `user` contains user-defined modules (if any).
- Folder `output` or `multirun` is created by IGM to store output data.

In the following sections, we give a brief description of the most important IGM modules, each identified by a keyword, and we refer to the documentation (https://instructed-glacier-model.github.io/igm-doc/) of each module for more details.

```
# @package _global_

defaults:
  - override /input:
      - load_ncdf
  - override /modules:
      - smb_simple
      - iceflow
      - time
      - thk
  - override /output:
      - write_ncdf

processes:
  time:
    start: 1880.0
    end: 2020.0
    save: 5.0
```

Figure 3. Example of IGM parameter file `params.yaml`.

3.1 Inputs modules

IGM comes with input modules that can read raster data from files (`load_ncdf`, `load_tif`, `local`), or collect them from a database (`oggm_shop`).

[TODO: give local the functionalities of loadncdf and loadtif] Modules `load_ncdf`, `load_tif`, `local` were built to load spatial 2D raster data from a NetCDF or Tiff file in IGM. The module is expected to import at least the basal topography, which is represented by variable `topg`, but can provide other fields such as the initial ice thickness `thk` and more. IGM has naming convention that must be followed to be recognized. Any field present in the NetCDF file will be passed and be available in IGM. At this stage, raster data can be resampled, coarsened or cropped to a specific regions.

Module `oggm_shop` utilizes the Open Glacier Global Model (OGGM Maussion et al., 2019) (and then depends on python package `oggm`) to acquire data for contemporary glaciers. Users provide the RGI ID of the glacier of interest(https://www.glims.org/maps/glims). The data provided by OGGM are either already processed for certain spatial resolution and margin size, otherwise can be re-processed on demand to obtain certain user-defined spatial resolutions and/or margin sizes. Upon running this module, its automatically downloads an ensemble of data related to the specified glacier, which is then stored in a folder named after the RGI ID in folder `data`. Users can select 2D gridded variables of interest from the available products of the `oggm_shop` to be loaded in IGM.

3.2 Processes modules

Processes modules implement update rules for all modelled variables within the time loop. Modules correspond to spe-

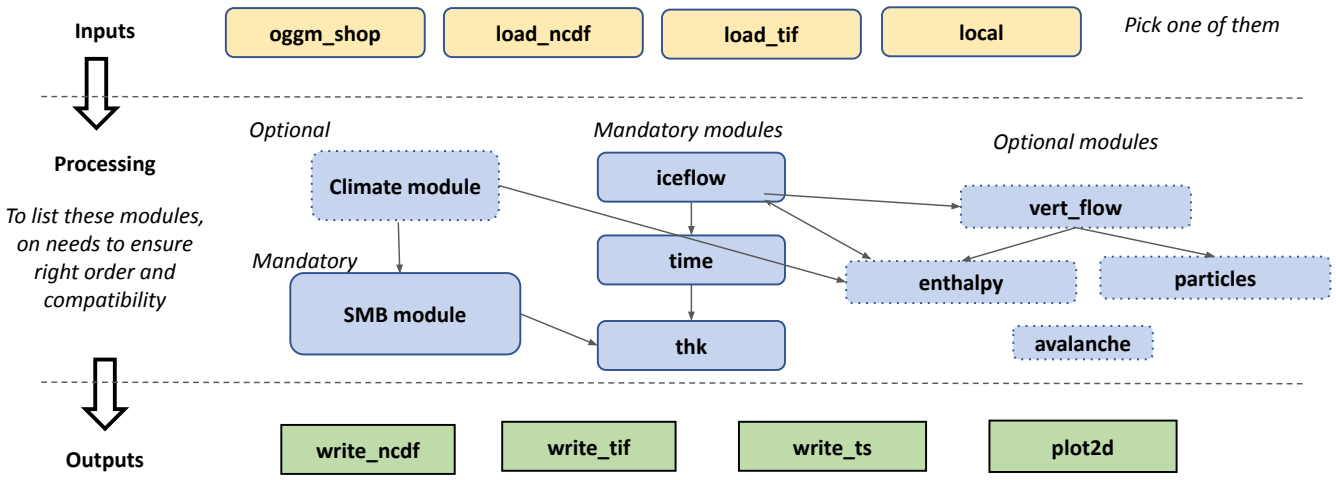

Figure 4. Flowchart of IGM modules. [TODO: UPDATE]

cific physical components. For instance, the `iceflow` module updates the ice velocity, the `thk` module updates the ice thickness, and so on. Processes modules are conveniently listed in order: for instance, `iceflow` must come before `thk` as the latter needs the first.

3.2.1 Climate

Climate modules `clim_XXX` implement climate forcing in IGM as required by the SMB or enthalpy models. They supply distributed field of near surface air temperature, and precipitation at a given time resolution (e.g. daily, weekly or monthly). Module `clim_oggm` reads monthly time series of historical climate data collected by the `oggm_shop` module, and generates monthly 2D raster fields of precipitation, mean temperature, and temperature variability extrapolated to the entire glacier surface using a reference height and a vertical lapse rates. Module `clim_deltat` loads a climate snapshot (associated to a certain period) and superposes a time-varying temperature offset (climate signal) that is often taken from paleo cliamte proxies (Seguinot et al., 2018). Module `clim_glacialindex` loads two climate snapshots (associated to a certain periods) and interpolate them using a climate signal and a glacial index approach (Jouvet et al., 2023).

3.2.2 Surface mass balance

Surface mass balance (SMB) modules `smb_XXX` provide a rule to update the SMB. Module `smb_simple` implements the SMB model defined by (1)-(2). To that aim, the four parameters z_{ELA}, β_{abl}, β_{acc}, m_{acc} are provided in an array for some times, and the module interpolates linearly them to provide a continous-in-time parametrization of the SMB. Module `smb_oggm` implements a monthly PDD model calibrated on geodetic mass balance data (Hugonnet et al., 2021) by OGGM (Maussion et al., 2019). The yearly SMB at any point is computed with (3) from monthly temperature and precipitation fields as described in Section 3.2.1. Laslty, module `smb_accmelt` implements the combined snow accumulation / PDD model with the tracking of the snow layer.

3.2.3 Ice flow

Module `iceflow` is central for both inverse and forward modelling. In the inverse model, it permits to carry the data assimilation while ensuring the consistence with the ice flow physics (Section 2.3). In the forward model, it permits to use and retrain the emulator to update the ice flow.

The inversion modelling can be activated as a preliminary step to a forward or prognostic model run that assumes isothermal ice (i.e. without additional `enthalpy` module, since the the thermal condition are inferred from the inversion rather than computed from the enthalpy module). Running this module requires that necessary observational data have been loading in IGM using an inputs module such as ice surface ice velocities $\mathbf{u}^{s,obs}$, the top surface elevation s^{obs}, some ice thickness profiles h_p^{obs}, observed flux divergence d^{obs}, glacier mask from RGI outlines. A formerly-inferred ice thickness field (Farinotti et al., 2019; Millan et al., 2022) can be used to initialize the inverse model. All the observational data (including individual ice thickness profiles) are rasterized on the same grid, with possible no-data (NaN) values that are ignored in the optimization. While the optimization problem is presented in its most general form (Section 2.3), the user selects a sub-ensemble of controls and constraints within the cost function based on the availability of data. Maintaining a balance between controls and costs/constraints is crucial to maintaining the well-posedness of the problem and guarding against multiple solutions. There are a couple parameters that need adjustement including i) confidence levels (i.e. tolerance to fit the data) $\sigma^u, \sigma^h, \sigma^s, \sigma^d$ or

ii) regularization weights $\alpha^h, \alpha^c, \alpha^A$. The data assimilation is monitored through the iterations thanks to misfit cost component.

The forward model relies on the ice flow emulator, which delivers ice flow field computationally efficiently. The user must first define physical parameters related to the ice flow (unless those were optimized in the data assimilation step), as well as parameters related to the vertical discretization (e.g., the number of vertical layers). Two other numerical parameters may need adjustements: i) the learning rate for the retraining of the CNN, which controls the strengh of retraining, ii) the retraining frequency, which needs to be sufficiently frequent to maintain accuracy, but not too much to mitigate computational costs. When treating large arrays, retraining must be done sequentially patch-wise for memory reason. The maximum size of patches is controlled by a parameter, which can be adpated to the size of the GPU memory. Last, it is possible to use a solver (instead of an emulator) to compute the iceflow, as well as using both the emulator and the solver (in diagnostic mode) to assess the fidelity of the iceflow emulator to the solver.

3.2.4 Time

Module `time` computes the time step (and update the time) with the following criteria: i) it is adjusted precisely to allow matching with specified saving times, whose the frequency is user-defined, ii) it remains below a designated maximum time step (often taken to one year to ensure that the time step is never too large), iii) it complies to the CFL condition (35), which ensures stability in the transport scheme for ice thickness evolution. The simulation starting and ending time parameters are part of this module.

3.2.5 Ice thickness

This module solves the mass conservation equation (22) using an explicit first-order upwind finite-volume scheme (as described in Section 2.2.3). It permits to update ice thickness based on the results obtained from the `iceflow` and `smb_XXX` modules.

3.2.6 Vertical ice flow

Since the `iceflow` module is based on the Blatter-Pattyn ice flow model and predicts only the horizontal components of ice velocity, an additional module, `vert_flow` additionally computes the vertical component by integrating the incompressibility condition (5). Module `vert_flow` module is mandatory in various other modules, such as the `particle` and the `enthalpy` module.

3.2.7 Enthalpy

Module `enthalpy` permits to update the 3D fields enthalpy (or equivalently the temperature and water content), and the 2D water thickness in the till given these quantities at previous time step, as well the 3D iceflow following the numerical scheme described in Section 3.2.7. In turn, this permits to update the sliding coefficient as well as the Arrhenius factor, which are used in the `iceflow` module.

3.2.8 Particles

Module `particles` permits to seed and track the position of virtual particles advected by the ice flow following the two methods described in Section 2.2.5. By default, the module seeds the accumulation area at regular intervals, however, the seeding can be easily customized.

3.3 Outputs modules

Outputs modules permit to monitor the results of IGM. The frequency of saving is determined by a parameter defined within the `time` module (as this time needs to be reached exactly). Modules `write_ncdf` and `write_tif` are responsible for writing 2D field variables that are specified in the parameter lists in ncdf or tif format files. Module `plot2d` module generates 2D plan-view plots of user-defined variables, which can be displayed in real-time or saved as image files.

3.4 User modules

Beside the modules listed above, IGM users can write their own inputs, processes, or outputs modules to load differently the data, couple new processes, or output the results in a different format. For each processes module, the user must provide three functions that are called at initialization, during the time loop, and at the end: `initialize(cfg,state)`, `update(cfg,state)`, and `finalize(cfg,state)`, while only one function `rune(cfg,state)` is required for inputs and outputs modules. Herabove, `cfg` contains all the parameters needed for the simulation, organized by attributes in a hierachical manner. They include "core" parameters (independant to any modules), and module-related ones (e.g., `cgf.processes.time.start` is the initial simulation time). On the other hand, `state` holds all the variables at a specific time (e.g., `state.U` and `state.thk` denotes the 3D ice flow and the 2D ice thickness fields). These variables are updated throughout the simulation to represent the evolving state of the glacier.

4 Applications

[TO COMPLETE]

4.1 Enthalpy experiments

[GJ: TOCHECK]

Following (Wang et al., 2020), we carry the two benchmark experiments proposed by (Kleiner et al., 2015) and one

from (Hewitt and Schoof, 2017) to verify the implementation of the enthalpy formulation.

Experiment A considers a flat parallel-sided slab with a constant thickness h = 1000 m without any ice dynamics so that only the vertical heat diffusion is modelled. We perform a transient simulation during 300'000 years with a constant geothermal heat flux at the base. The surface temperatures are prescribed to -30°C all the time except between $t = 100'000$ years and $t = 150'000$ years. Fig. 5 shows the time series of the simulated basal temperatures, basal melt rates, and basal water layer thicknesses. As a result, they compare very well with with reference solutions shown in Fig 1 of (Kleiner et al., 2015).

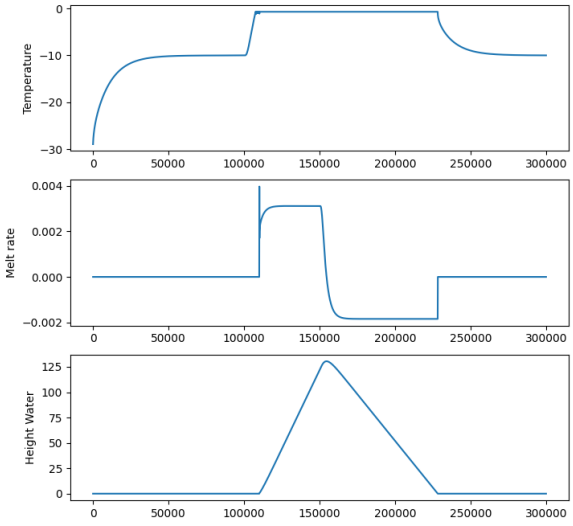

Figure 5. Result of experiment A of (Kleiner et al., 2015).

Experiment B also considers a parallel-sided slab but with a constant thickness $h = 200$ m and a slope of 4 °. The horizontal ice velocity is defined by a SIA-like profile, while the vertical ice velocity is constant. The enthalpy at the surface is prescribed with -3°C and zero water content. We assume zero geothermal heat flux and basal sliding velocity. Here, only the strain heating acts as heat source. We initialize the transient simulation with a constant enthalpy field with -1.5°C and zero water content. We assume the thermal diffusivity of temperate ice to be 100'000 times lower than the one of the cold ice. We run the simulation until reaching a steady state, which is after 1000 years. As a result, they compare well with the reference solution shown in Fig 4 of (Kleiner et al., 2015).

Last, we reproduced the experiment proposed by Hewitt and Schoof (2017), which consider an idealized ice cap geometry. Again, the horizontal ice velocity and the strain heating are defined based on SIA-like ice flow. We impose a constant temperature on the top surface of the ice cap, while we assume the bedrock to be at the pressure melting point. The simulation is initialized with a constant enthalpy field with -11°C and zero water content. The simulation is run until steady state. Fig. (7) shows the result at steady-state with a drainage function. As a result, it compares reasonably with the reference solution shown in Fig 7b of (Hewitt and Schoof, 2017). It is likely that a higher number of layers would imporve the match between the two.

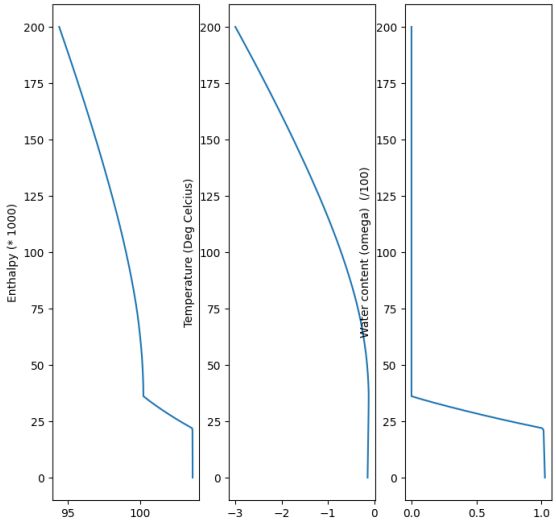

Figure 6. Result of experiment B of (Kleiner et al., 2015).

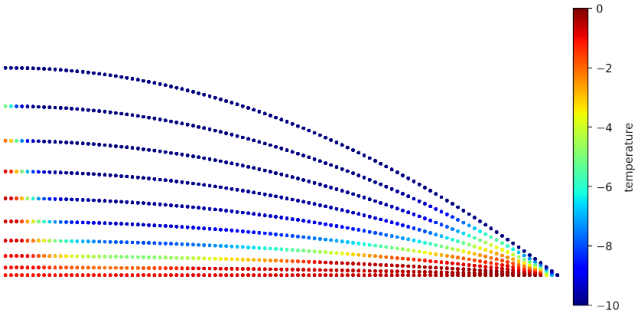

Figure 7. Result of the ice cap experiment of (Hewitt and Schoof, 2017).

5 Conclusions

TEXT

Code availability. TEXT

Data availability. TEXT

Code and data availability. TEXT

Sample availability. TEXT

Video supplement. TEXT

Appendix A

A1

Author contributions. TEXT

Competing interests. TEXT

Disclaimer. TEXT

Acknowledgements. TEXT

References

Aschwanden, A., Bueler, E., Khroulev, C., and Blatter, H.: An enthalpy formulation for glaciers and ice sheets, Journal of Glaciology, 58, 441–457, https://doi.org/10.3189/2012JoG11J088, 2012.

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

Brædstrup, C. F., Damsgaard, A., and Egholm, D. L.: Ice-sheet modelling accelerated by graphics cards, Computers & Geosciences, 72, 210–220, https://doi.org/10.1016/j.cageo.2014.07.019, 2014.

Bueler, E. and van Pelt, W.: Mass-conserving subglacial hydrology in the Parallel Ice Sheet Model version 0.6, Geoscientific Model Development, 8, 1613–1635, https://doi.org/10.5194/gmd-8-1613-2015, 2015.

Calov, R. and Greve, R.: Correspondence: A semi-analytical solution for the positive degree-day model with stochastic temperature variations, J. Glaciol, 51, 173–175, https://doi.org/10.3189/172756505781829601, 2005.

Cuffey, K. and Paterson, W.: The physics of glaciers, Academic Press, glacier depth average velocity, 2010.

Dias dos Santos, T., Morlighem, M., and Brinkerhoff, D.: A new vertically integrated MOno-Layer Higher-Order (MOLHO) ice flow model, The Cryosphere, 16, 179–195, 2022.

Farinotti, D., Huss, M., Fürst, J. J., Landmann, J., Machguth, H., Maussion, F., and Pandit, A.: A consensus estimate for the ice thickness distribution of all glaciers on Earth, Nature Geoscience, 12, 168–173, https://doi.org/10.1038/s41561-019-0300-3, 2019.

Gagliardini, O., Zwinger, T., Gillet-Chaulet, F., Durand, G., Favier, L., de Fleurian, B., Greve, R., Malinen, M., Martín, C., Råback, P., Ruokolainen, J., Sacchettini, M., Schäfer, M., Seddik, H., and Thies, J.: Capabilities and performance of Elmer/Ice, a new-generation ice sheet model, Geoscientific Model Development, 6, 1299–1318, https://doi.org/10.5194/gmd-6-1299-2013, 2013.

Glen, J. W.: Rate of Flow of Polycrystalline Ice, Nature, 172, 721–722, https://doi.org/10.1038/172721a0, 1953.

Hewitt, I. J. and Schoof, C.: Models for polythermal ice sheets and glaciers, The Cryosphere, 11, 541–551, 2017.

Hock, R.: Temperature index melt modelling in mountain areas, Journal of Hydrology, 282, 104–115, https://doi.org/10.1016/S0022-1694(03)00257-9, 2003.

Hugonnet, R., McNabb, R., Berthier, E., Menounos, B., Nuth, C., Girod, L., Farinotti, D., Huss, M., Dussaillant, I., Brun, F., et al.: Accelerated global glacier mass loss in the early twenty-first century, Nature, 592, 726–731, https://doi.org/10.1038/s41586-021-03436-z, 2021.

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

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

Jouvet, G. and Cordonnier, G.: Ice-flow model emulator based on physics-informed deep learning, Journal of Glaciology, pp. 1–15, https://doi.org/10.1017/jog.2023.73, 2023.

Jouvet, G., Cordonnier, G., Kim, B., Lüthi, M., Vieli, A., and Aschwanden, A.: Deep learning speeds up ice flow modelling by several orders of magnitude, Journal of Glaciology, 68, 651–664, https://doi.org/10.1017/jog.2021.120, 2022.

Jouvet, G., Cohen, D., Russo, E., Buzan, J., Raible, C. C., Haeberli, W., Kamleitner, S., Ivy-Ochs, S., Imhof, M. A., Becker, J. K., et al.: Coupled climate-glacier modelling of the last glaciation in the Alps, Journal of Glaciology, pp. 1–15, 2023.

Khroulev, C. and the PISM Authors: PISM, a Parallel Ice Sheet Model v1.2: User's Manual, www.pism-docs.org, 2020.

Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.

Kleiner, T., Rückamp, M., Bondzio, J. H., and Humbert, A.: Enthalpy benchmark experiments for numerical ice sheet models, The Cryosphere, 9, 217–228, 2015.

Larour, E., Seroussi, H., Morlighem, M., and Rignot, E.: Continental scale, high order, high spatial resolution, ice sheet modeling using the Ice Sheet System Model (ISSM), Journal of Geophysical Research: Earth Surface, 117, https://doi.org/10.1029/2011JF002140, 2012.

Lipscomb, W. H., Price, S. F., Hoffman, M. J., Leguy, G. R., Bennett, A. R., Bradley, S. L., Evans, K. J., Fyke, J. G., Kennedy, J. H., Perego, M., et al.: Description and evaluation of the community ice sheet model (CISM) v2. 1, Geoscientific Model De-

velopment, 12, 387–424, https://doi.org/10.5194/gmd-12-387-2019, 2019.

Maussion, F., Butenko, A., Champollion, N., Dusch, M., Eis, J., Fourteau, K., Gregor, P., Jarosch, A. H., Landmann, J. M., Oesterle, F., et al.: The Open Global Glacier Model (OGGM) v1. 1, Geoscientific Model Development, 12, 909–931, https://doi.org/10.5194/gmd-12-909-2019, 2019.

Millan, R., Mouginot, J., Rabatel, A., and Morlighem, M.: Ice velocity and thickness of the world's glaciers, Nature Geoscience, 15, 124–129, https://doi.org/10.1038/s41561-021-00885-z, 2022.

Paterson, W. S. B.: The Physics of Glaciers, Pergamon, New York, third edn., 1994.

Pattyn, F.: The paradigm shift in Antarctic ice sheet modelling, Nature communications, 9, 1–3, https://doi.org/10.1038/s41467-018-05003-z, 2018.

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

Rounce, D. R., Hock, R., and Shean, D. E.: Glacier mass change in High Mountain Asia through 2100 using the open-source python glacier evolution model (PyGEM), Frontiers in Earth Science, 7, 331, https://doi.org/10.3389/feart.2019.00331, 2020.

Schoof, C. and Hewitt, I.: Ice-Sheet Dynamics, Annual Review of Fluid Mechanics, 45, 217–239, https://doi.org/10.1146/annurev-fluid-011212-140632, 2013.

Seguinot, J., Ivy-Ochs, S., Jouvet, G., Huss, M., Funk, M., and Preusser, F.: Modelling last glacial cycle ice dynamics in the Alps, The Cryosphere, 12, 3265–3285, 2018.

Višnjević, V., Herman, F., and Prasicek, G.: Climatic patterns over the European Alps during the LGM derived from inversion of the paleo-ice extent, Earth and Planetary Science Letters, 538, 116 185, https://doi.org/10.1016/j.epsl.2020.116185, 2020.

Wang, Y., Zhang, T., Xiao, C., Ren, J., and Wang, Y.: A two-dimensional, higher-order, enthalpy-based thermomechanical ice flow model for mountain glaciers and its benchmark experiments, Computers & geosciences, 141, 104 526, https://doi.org/10.1016/j.cageo.2020.104526, 2020.

Winkelmann, R., Martin, M. A., Haseloff, M., Albrecht, T., Bueler, E., Khroulev, C., and Levermann, A.: The Potsdam Parallel Ice Sheet Model (PISM-PIK) – Part 1: Model description, The Cryosphere, 5, 715–726, https://doi.org/10.5194/tc-5-715-2011, 2011.

Zekollari, H., Huss, M., Farinotti, D., and Lhermitte, S.: Ice-Dynamical Glacier Evolution Modeling—A Review, Reviews of Geophysics, 60, e2021RG000 754, https://doi.org/10.1029/2021RG000754, 2022.