Graphics-processing-unit-accelerated ice flow solver for unstructured meshes using the Shallow-Shelf Approximation (FastIceFlo v1.0.1)

Geosci. Model Dev., 17, 899–909, 2024
https://doi.org/10.5194/gmd-17-899-2024

Geoscientific Model Development
Open Access

Graphics-processing-unit-accelerated ice flow solver for unstructured meshes using the Shallow-Shelf Approximation (FastIceFlo v1.0.1)

Anjali Sandip[1], **Ludovic Räss**[2,3,a], **and Mathieu Morlighem**[4]

[1]Department of Mechanical Engineering, University of North Dakota, North Dakota, USA
[2]Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Zurich, Switzerland
[3]Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland
[4]Department of Earth Sciences, Dartmouth College, New Hampshire, USA
[a]now at: Swiss Geocomputing Centre, Faculty of Geosciences and Environment,
University of Lausanne, Lausanne, Switzerland

Correspondence: Anjali Sandip (anjali.sandip@und.edu)

Received: 3 March 2023 – Discussion started: 10 May 2023
Revised: 23 November 2023 – Accepted: 19 December 2023 – Published: 2 February 2024

Abstract. Ice-sheet flow models capable of accurately projecting their future mass balance constitute tools to improve flood risk assessment and assist sea-level rise mitigation associated with enhanced ice discharge. Some processes that need to be captured, such as grounding-line migration, require high spatial resolution (under the kilometer scale). Conventional ice flow models mainly execute on central processing units (CPUs), which feature limited parallel processing capabilities and peak memory bandwidth. This may hinder model scalability and result in long run times, requiring significant computational resources. As an alternative, graphics processing units (GPUs) are ideally suited for high spatial resolution, as the calculations can be performed concurrently by thousands of threads, processing most of the computational domain simultaneously. In this study, we combine a GPU-based approach with the pseudo-transient (PT) method, an accelerated iterative and matrix-free solution strategy, and investigate its performance for finite elements and unstructured meshes with application to two-dimensional (2-D) models of real glaciers at a regional scale. For both the Jakobshavn and Pine Island glacier models, the number of nonlinear PT iterations required to converge a given number of vertices (N) scales in the order of $\mathcal{O}(N^{1.2})$ or better. We further compare the performance of the PT CUDA C implementation with a standard finite-element CPU-based implementation using the price-to-performance metric. The price of a single Tesla V100 GPU is 1.5 times that of two Intel Xeon Gold 6140 CPUs. We expect a minimum speedup of at least 1.5 times to justify the Tesla V100 GPU price to performance. Our developments result in a GPU-based implementation that achieves this goal with a speedup beyond 1.5 times. This study represents a first step toward leveraging GPU processing power, enabling more accurate polar ice discharge predictions. The insights gained will benefit efforts to diminish spatial resolution constraints at higher computing performance. The higher computing performance will allow for ensembles of ice-sheet flow simulations to be run at the continental scale and higher resolution, a previously challenging task. The advances will further enable the quantification of model sensitivity to changes in upcoming climate forcings. These findings will significantly benefit process-oriented sea-level-projection studies over the coming decades.

1 Introduction

Global mean sea level is rising at an average rate of 3.7 mm yr^{-1}, posing a significant threat to coastal communities and global ecosystems (Hinkel et al., 2014; Kopp et al., 2016). The increase in ice discharge from the Greenland and Antarctic ice sheets significantly contributes to sea-level rise. However, their dynamic response to climate change remains a fundamental uncertainty in future projection (Rietbroek et al., 2016; Chen et al., 2017; IPCC, 2021). While much progress has been made over the last decades, several critical physical processes, such as calving and ice-sheet basal sliding, remain poorly understood (Pattyn and Morlighem, 2020). Existing computational resources limit the spatial resolution and simulation time on which continental-scale ice-sheet models can run. Some processes, such as grounding-line migration or ice front dynamics, require spatial resolutions in the order of 1 km or smaller (Larour et al., 2012; Aschwanden et al., 2021; Castleman et al., 2022).

Most numerical models use a solution strategy designed to target central processing units (CPUs) and shared memory parallelization. CPUs' parallel processing capabilities, peak memory bandwidth, and power consumption remain limiting factors. It remains to be seen whether high-resolution modeling will become feasible at the continental scale (or ice-sheet scale). Specifically, complex flow models, such as full-Stokes models, may remain challenging to employ beyond the regional scale. Trying to overcome the technical limitations tied to CPU-based computing, graphics processing units (GPUs) feature interesting capabilities and have been booming over the past decade (Brædstrup et al., 2014; Häfner et al., 2021). Developing algorithms and solvers to leverage GPU computing capabilities has become essential and has resulted in active development within scientific computing and high-performance computing (HPC) communities.

The traditional way of solving the partial differential equations governing ice-sheet flow, employing, e.g., finite-element analysis, may represent a challenge to leverage GPU acceleration efficiently. Handling unstructured grid geometries and having global-to-local indexing patterns may significantly hinder efficient memory transfers and optimal bandwidth utilization. Räss et al. (2020) proposed an alternative approach by reformulating the flow equations in the form of pseudo-transient (PT) updates. The PT method augments the time-independent governing ice-sheet flow equation by physically motivated pseudo-time-dependent terms. The added pseudo-time τ terms turn the initial time-independent elliptic equations into a parabolic form, allowing for an explicit iterative pseudo-time integration to reach a steady state and, thus, the solution of the initial elliptic problem. The explicit pseudo-time integration scheme eliminates the need for the expensive direct–iterative type of solvers, making the proposed approach matrix-free and attractive for various parallel computing approaches (Frankel, 1950; Poliakov et al., 1993; Kelley and Liao, 2013). Räss et al. (2020) introduced this method specifically targeting GPU computing to enable the development of high-spatial-resolution full-Stokes ice-sheet flow solvers in two dimensions (2-D) and three dimensions (3-D), respectively, on uniform grids (Räss et al., 2020). The approach unveils a promising solution strategy, but the finite-difference discretization on uniform and structured grids and the idealized test cases represent actual limitations.

Here, we build upon work from previous studies (Räss et al., 2019, 2020) on the accelerated PT method for finite-difference discretization on uniform, structured grids and extend it to finite-element discretization and unstructured meshes. We developed a CUDA C implementation of the PT depth-integrated Shallow-Shelf Approximation (SSA) and applied it to regional-scale glaciers, Pine Island Glacier and Jakobshavn Isbræ, in West Antarctica and Greenland, respectively. We compare the PT CUDA C implementation with a more standard finite-element CPU-based implementation available within the Ice-sheet and Sea-level System Model (ISSM). Our comparison uses the same mesh, model equations, and boundary conditions. In Sect. 2, we present the mathematical reformulation of the 2-D SSA momentum balance equations to incorporate the additional pseudo-transient terms needed for the PT method. We provide the weak formulation and discuss the spatial discretization. Section 3 describes the numerical experiments conducted, chosen glacier model configurations, hardware implementation, and performance assessment metrics. In Sects. 4 and 5, we illustrate the method's performance and conclude on future research directions.

2 Methods

2.1 Mathematical formulation of the 2-D SSA model

We employ the SSA (MacAyeal, 1989) formulation to solve the momentum balance equation:

$$\nabla \cdot (2H\mu\dot{\boldsymbol{\varepsilon}}_{\mathrm{SSA}}) = \rho g H \nabla \boldsymbol{s} + \alpha^{2}\boldsymbol{v} , \tag{1}$$

where the 2-D SSA strain rate $\dot{\boldsymbol{\varepsilon}}_{\mathrm{SSA}}$ is defined as

$$\dot{\boldsymbol{\varepsilon}}_{\mathrm{SSA}} = \begin{pmatrix} 2\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} & \frac{\partial v_x}{\partial y} \\ \frac{\partial v_x}{\partial y} & 2\frac{\partial v_y}{\partial y} + \frac{\partial v_x}{\partial x} \end{pmatrix} . \tag{2}$$

The terms v_x and v_y represent the respective x and y ice velocity components, H is the ice thickness, ρ is the ice density, g is the gravitational acceleration, s is the glacier's upper surface z coordinate, and $\alpha^2\boldsymbol{v}$ is the basal friction term. The ice viscosity μ follows Glen's flow law (Glen, 1955):

$$\mu = \frac{B}{2\,\dot{\varepsilon}_{\mathrm{e}}^{(n-1)/n}} , \tag{3}$$

where B is ice rigidity, $\dot{\varepsilon}_{\mathrm{e}}$ is the effective strain rate, and $n = 3$ is Glen's power-law exponent. We regularize the strain-rate-dependent viscosity formulation in the numerical implementation by capping it at 1×10^5 to address the singularity arising in regions of the computational domain where the strain rate tends toward zero.

As boundary conditions, we apply water pressure at the ice front Γ_σ and nonhomogeneous Dirichlet boundary conditions Γ_u on the other boundaries (based on observed velocity).

2.2 Mathematical reformulation of the 2-D SSA model to incorporate the PT method

The solution to the SSA ice flow problem is commonly achieved by discretizing Eq. (1) using the finite-element or finite-difference method. The discretized problem can be solved using a direct, direct–iterative, or iterative approach. Robust matrix-based direct-type solvers exhibit significant scaling limitations restricting their applicability when considering high-resolution or 3-D configurations. Iterative solving approaches allow one to circumvent most scaling limitations. However, they may encounter convergence issues for suboptimally conditioned, stiff, or highly nonlinear problems, resulting in the non-tractable growth of the iteration count. Thus, one challenge is to prevent the iteration count from growing exponentially. We propose the accelerated pseudo-transient (PT) method as an alternative approach. The method augments the steady-state viscous flow (Eq. 1) by adding the usually ignored transient term, which can be further used to integrate the equations in pseudo-time τ, seeking an implicit solution once the steady state is reached, i.e., when $\tau \rightarrow \infty$.

Building upon work from previous studies (Omlin et al., 2018; Duretz et al., 2019; Räss et al., 2019, 2020, 2022), we reformulate the 2-D SSA steady-state momentum balance equations into a transient diffusion-like formulation for flow velocities $v_{x,y}$ by incorporating the usually omitted time derivative.

$$\nabla \cdot (2H\mu\dot{\boldsymbol{\varepsilon}}_{\text{SSA}}) - \rho g H \nabla s - \alpha^2 \boldsymbol{v} = \rho H \frac{\partial \boldsymbol{v}}{\partial \tau} \tag{4}$$

The velocity–time derivatives represent physically motivated expressions that we can further use to iteratively reach a steady state, which provides the solution of the original time-independent equations. As we are only interested in the steady-state here, transient processes evolve in numerical time or pseudo-time τ:

$$\rho H \frac{\partial v_x}{\partial \tau} = R_x \, ,$$
$$\rho H \frac{\partial v_y}{\partial \tau} = R_y \, , \tag{5}$$

where R_x and R_y correspond to the right-hand-side expressions of Eq. (1) and define the residuals of the original SSA equations for which we are seeking a solution. We define the transient *pseudo*-time step $\Delta\tau$ as a field variable (that is spatially variable) chosen to minimize the number of nonlinear PT iterations.

2.3 Pseudo-time-stepping method

Here, we advance in numerical pseudo-time using a forward Euler pseudo-time-stepping method. We choose our time derivative by approximating the transient diffusive system for both v_x and v_y:

$$\rho H \frac{\partial v_x}{\partial \tau} = \frac{\partial}{\partial x}\left(4H\mu\frac{\partial v_x}{\partial x}\right) ,$$
$$\rho H \frac{\partial v_y}{\partial \tau} = \frac{\partial}{\partial y}\left(4H\mu\frac{\partial v_y}{\partial y}\right) , \tag{6}$$

where one recognizes the diffusive variables $v_{x,y}$ and the effective dynamic viscosity $4\mu/\rho$ as a diffusion coefficient. Using the analogy of a diffusive process, we can define a CFL (Courant–Friedrichs–Lewy)-like stability criterion for the PT iterative scheme. The explicit CFL-stability-based time step for viscous flow is given by the following:

$$\Delta\tau_{\max} = \rho \frac{\Delta x^2}{4\mu(1+\mu_{\text{b}}) \times n_{\text{dim}}} \, , \tag{7}$$

where Δx represents the grid spacing, μ_{b} is the numerical bulk ice viscosity, and $n_{\text{dim}} = 2.1, 4.1$, and 6.1 in 1-D, 2-D, and 3-D, respectively.

2.4 Viscosity continuation

We implement a continuation on the nonlinear strain-rate-dependent effective viscosity μ_{eff} to avoid the iterative solution process diverging, as strain-rate values may not satisfy the momentum balance at the beginning of the iterative process and may, thus, be far from equilibrium. At every pseudo-time step, the effective viscosity μ_{eff} is updated in the logarithmic space:

$$\mu_{\text{eff}} = \exp\left(\theta_\mu \log(\mu) + (1-\theta_\mu)\log(\mu_{\text{eff}}^{\text{old}}\right) \, , \tag{8}$$

where the scalar $10^{-2} < \theta_\mu < 1$ is selected such that we provide sufficient time to relax the nonlinear viscosity at the start of the pseudo-iterative loop.

2.5 Acceleration owing to damping

The major limitation of this simple first-order, or Picard-type, iterative approach resides in the poor iteration count scaling with increased numerical resolution. The number of iterations needed to converge for a given problem for N number of grid points involved in the computation scales in the order of $\mathcal{O}(N^2)$.

To address this limitation, we consider a second-order method, referred to as the second-order Richardson method, as introduced by Frankel (1950). This approach allows us to aggressively reduce the number of iterations to the number of grid points, making the method scale as $\approx \mathcal{O}(N^{1.2})$. Optimal scaling can be achieved by realizing that the PT framework's diffusion type of updates readily provided can be divided into two wave-like update steps. Transitioning from diffusion to wave-like pseudo-physics exhibits two main advantages: (i) the wave-like time step limit is a function of Δx instead of Δx^2; (ii) it is possible to turn the wave equation into a damped wave equation. The latter permits one to find optimal tuning parameters to achieve optimal damping, resulting in fast convergence. Let us assume the following diffusion-like update step, reported here for the x direction only:

$$\rho H \frac{\partial v_x}{\partial \tau} = \frac{\partial}{\partial x} 4H\mu \frac{\partial v_x}{\partial x} \ . \tag{9}$$

The above expression results in the following update rule:

$$v_x = v_x^{\text{old}} + \frac{\Delta\tau_{\text{D}}}{\rho H}\left(\frac{\partial}{\partial x} 4H\mu \frac{\partial v_x}{\partial x}\right) , \tag{10}$$

where $\Delta\tau_{\text{D}} \approx \Delta x^2/(4\mu/\rho)/4.1$ is the diffusion-like time step limit. This system can be separated into a residual assignment A_x and the velocity update v_x:

$$A_x = \frac{1}{\rho H}\left(\frac{\partial}{\partial x} 4H\mu \frac{\partial v_x}{\partial x}\right) , \tag{11}$$

$$v_x = v_x^{\text{old}} + \Delta\tau_{\text{D}} A_x . \tag{12}$$

Converting Eq. (11) into an update rule using a step size of $(1-\gamma)$,

$$A_x = A_x^{\text{old}}(1-\gamma) + \frac{1}{\rho H}\left(\frac{\partial}{\partial x} 4H\mu \frac{\partial v_x}{\partial x}\right) , \tag{13}$$

turns the system composed of Eqs. (12) and (13) into a damped wave equation similar to what was suggested by Frankel (1950). Ideal convergence can be reached upon selecting the appropriate damping parameter γ. To maintain solution stability, we include relaxation θ_v:

$$v_x = v_x^{\text{old}} + \theta_v \Delta\tau_{\text{D}} A_x , \tag{14}$$

where $0 < \gamma < 1$ and $0 < \theta_v < 1$.

Alternative and complementary details about the PT acceleration can be found in Räss et al. (2019), Duretz et al. (2019), and Räss et al. (2020), while an in-depth analysis is provided in Räss et al. (2022).

2.6 Weak formulation and finite-element discretization

Using the PT method, the equations to solve are as referenced in Eq. (4):

$$\rho H \frac{\partial \boldsymbol{v}}{\partial \tau} = \nabla \cdot 2H\mu\dot{\boldsymbol{\varepsilon}}_{\text{SSA}} - \rho g H \nabla s - \alpha^2 \boldsymbol{v} \ . \tag{15}$$

The weak form of the equation, assuming homogeneous Dirichlet conditions along all model boundaries for simplicity, reads: $\forall \boldsymbol{w} \in \mathcal{H}^1(\Omega)$,

$$\int_\Omega \rho H \frac{\partial \boldsymbol{v}}{\partial \tau} \cdot \mathbf{w} \mathrm{d}\Omega + \int_\Omega 2H\mu\dot{\boldsymbol{\varepsilon}}_{\text{SSA}} \cdot \nabla \boldsymbol{w} \ \mathrm{d}\Omega = \int_\Omega -\rho g H \nabla s \cdot \boldsymbol{w} - \alpha^2 \boldsymbol{v} \cdot \boldsymbol{w} \ \mathrm{d}\Omega , \tag{16}$$

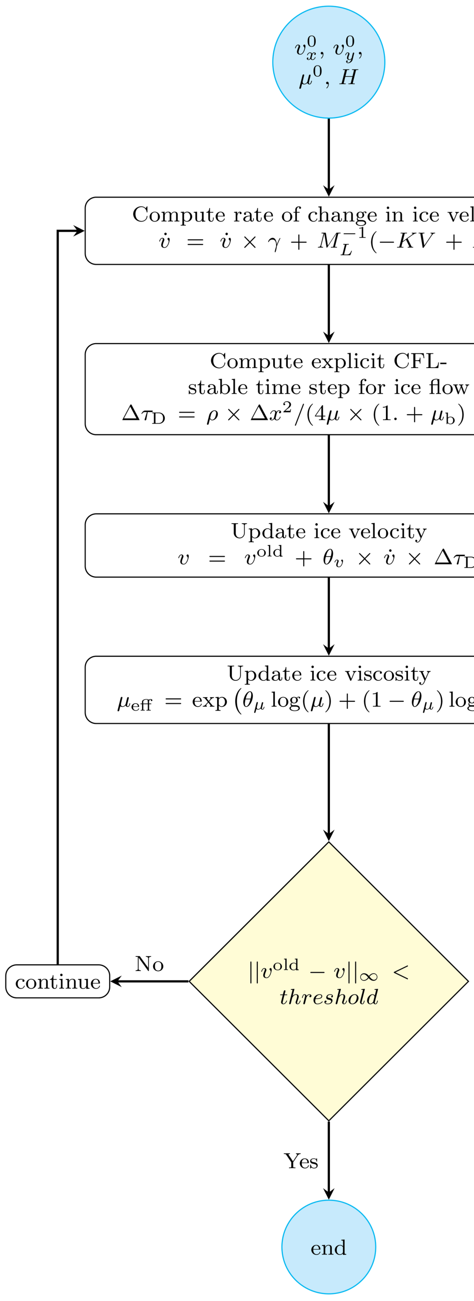
v_x^0, v_y^0, μ^0, H
Compute rate of change in ice velocity
$\dot{v} \ = \ \dot{v} \times \gamma + M_L^{-1}(-KV + F)$
Compute explicit CFL-stable time step for ice flow
$\Delta\tau_{\text{D}} \ = \ \rho \times \Delta x^2/(4\mu \times (1. + \mu_{\text{b}}) \times 4.1)$
Update ice velocity
$v \ = \ v^{\text{old}} + \theta_v \times \dot{v} \times \Delta\tau_{\text{D}}$
Update ice viscosity
$\mu_{\text{eff}} \ = \exp\left(\theta_\mu \log(\mu) + (1-\theta_\mu)\log(\mu_{\text{eff}}^{\text{old}})\right)$
$\|v^{\text{old}} - v\|_\infty <$ *threshold*
No
continue
Yes
end

Figure 1. PT iterative algorithm for unstructured meshes applied to solve 2-D SSA momentum balance equations.

where $\mathcal{H}^1(\Omega)$ is the space of square-integrable functions whose first derivatives are also square integrable.

Once discretized using the finite-element method, the matrix system to solve reads:

$$\mathbf{M}\dot{\boldsymbol{V}} + \mathbf{K}\boldsymbol{V} = \boldsymbol{F} , \tag{17}$$

where $\boldsymbol{M}$ is the mass matrix, $\mathbf{K}$ is the stiffness matrix, $\boldsymbol{F}$ is the right-hand-side or load vector, and $\boldsymbol{V}$ is the vector of ice velocity.

We can compute $\dot{\mathbf{V}}$ by solving

$$\dot{\mathbf{V}} \simeq \mathbf{M}_{\mathrm{L}}^{-1}\left(-\mathbf{K}\mathbf{V}+\boldsymbol{F}\right) , \tag{18}$$

where $\mathbf{M}_{\mathrm{L}}$ stands for the lumped mass matrix that permits one to avoid the resolution of a matrix system.

Hence, we have an explicit expression of the time derivative of the ice velocity for each vertex of the mesh:

$$\begin{aligned}\dot{v}_{xi} =& \frac{1}{\rho H m_{\mathrm{L}i}}\left(-\int_\Omega \left(4H\mu\frac{\partial v_x}{\partial x}+2H\mu\frac{\partial v_y}{\partial y}\right)\frac{\partial \varphi_i}{\partial x}\right.\\ &+\left(H\mu\frac{\partial v_x}{\partial y}+H\mu\frac{\partial v_y}{\partial x}\right)\frac{\partial \varphi_i}{\partial y}\mathrm{d}\Omega\\ &\left.+\int_\Omega -\rho g H\frac{\partial s}{\partial x}\varphi_i-\alpha^2 v_x\varphi_i\,\mathrm{d}\Omega\right) ,\end{aligned} \tag{19}$$

$$\begin{aligned}\dot{v}_{yi} =& \frac{1}{\rho H m_{\mathrm{L}i}}\left(-\int_\Omega \left(4H\mu\frac{\partial v_y}{\partial y}+2H\mu\frac{\partial v_x}{\partial x}\right)\frac{\partial \varphi_i}{\partial y}\right.\\ &+\left(H\mu\frac{\partial v_x}{\partial y}+H\mu\frac{\partial v_y}{\partial x}\right)\frac{\partial \varphi_i}{\partial x}\mathrm{d}\Omega\\ &\left.+\int_\Omega -\rho g H\frac{\partial s}{\partial y}\varphi_i-\alpha^2 v_y\varphi_i\mathrm{d}\Omega\right) ,\end{aligned} \tag{20}$$

where $m_{\mathrm{L}i}$ is the component number i along the diagonal of the lumped mass matrix $\mathbf{M}_{\mathrm{L}}$.

For every nonlinear PT iteration, we compute the rate of change in velocity $\dot{\boldsymbol{v}}$ and the explicit CFL-stability-based time step $\Delta\tau$. We then deploy the reformulated 2-D SSA momentum balance equations to update ice velocity $\boldsymbol{v}$ followed by ice viscosity μ_{eff}. We iterate in pseudo-time until the stopping criterion is met (Fig. 1).

3 Numerical experiments

3.1 Glacier model configurations

To test the performance of the PT method beyond simple idealized geometries, we apply it to two regional-scale glaciers: Jakobshavn Isbræ, in Western Greenland, and Pine Island Glacier, in West Antarctica (Fig. 2). For Jakobshavn Isbræ, we rely on BedMachine Greenland v4 (Morlighem et al., 2017) and also invert for basal friction to infer the basal boundary conditions. Note that the inversion is run on the Ice-sheet and Sea-level System Model (ISSM), using a standard approach (Larour et al., 2012). For Pine Island Glacier, we initialize the ice geometry using BedMachine Antarctica v2 (Morlighem et al., 2020) and infer the friction coefficient using surface velocities derived from satellite interferometry (Rignot et al., 2011).

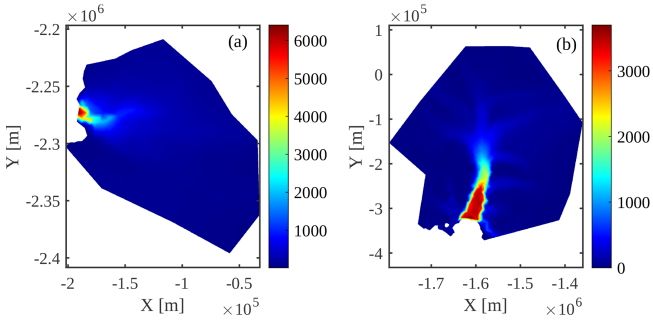

Figure 2. Glacier model configurations: observed surface velocities (in $\mathrm{m\,yr^{-1}}$) interpolated on a uniform mesh. Panels **(a)** and **(b)** correspond to Jakobshavn Isbræ and Pine Island Glacier, respectively.

3.2 Hardware implementation

We developed a CUDA C implementation to solve the SSA equations using the PT approach on unstructured meshes. We choose a stopping criterion of $||v^{\mathrm{old}}-v||_\infty < 10\,\mathrm{m\,yr^{-1}}$. The software solves the 2-D SSA momentum balance equations on a single GPU. We use an NVIDIA Tesla V100 SXM2 GPU with 16 GB (gigabytes) of device RAM and an NVIDIA A100 SXM4 with 80 GB of device RAM. We compare the PT implementation's results on a Tesla V100 GPU with ISSM's "standard" CPU implementation using a conjugate gradient (CG) iterative solver. We used a 64-bit 18-core Intel Xeon Gold 6140 processor for the CPU comparison, with 192 GB of RAM available. We perform multicore Message Passing Interface (MPI)-parallelized ice-sheet flow simulations on two CPUs, with all 36 cores enabled (Larour et al., 2012; Habbal et al., 2017). All simulations use double-precision arithmetic computations.

3.3 Performance assessment metrics

To investigate the PT CUDA C implementation for unstructured meshes, we report the number of vertices (or grid size) and the corresponding number of nonlinear PT iterations needed to meet the stopping criterion. We employ the computational time required to reach convergence as a proxy to assess and compare the performance of the PT CUDA C with the ISSM CG CPU implementation. We make sure to exclude all pre- and post-processing steps from the timing. We quantify the relative performance of the CPU and GPU implementations as the speedup (S), given by the following:

$$S=\frac{t_{\mathrm{CPU}}}{t_{\mathrm{GPU}}} . \tag{21}$$

The PT method employed to solve the nonlinear momentum balance equations results in a memory-bound algorithm (Räss et al., 2020); therefore, the wall time depends on the memory throughput. In addition to the speedup, we employ the effective memory throughput metric to assess the performance of the PT CUDA C implementation developed in this

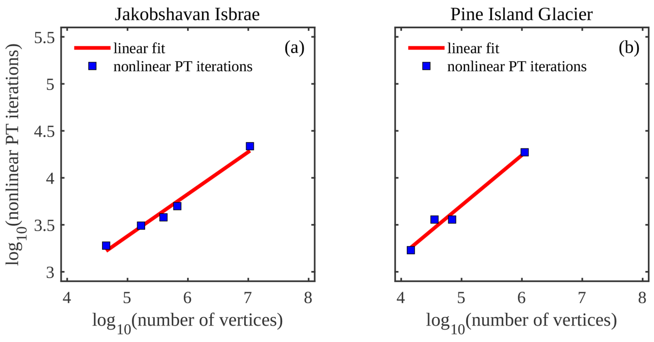

Figure 3. Performance assessment of the PT CUDA C implementation for unstructured meshes.

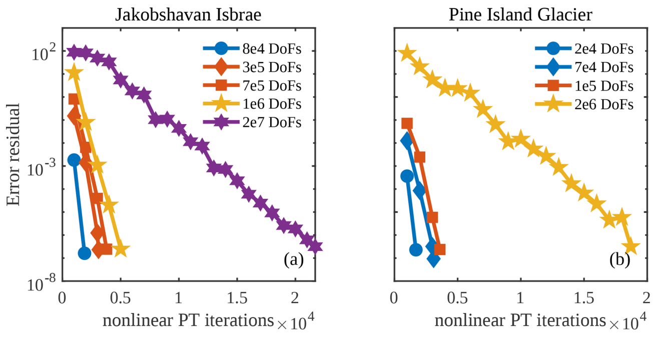

Figure 4. Residual error evolution of the PT CUDA C implementation for unstructured meshes.

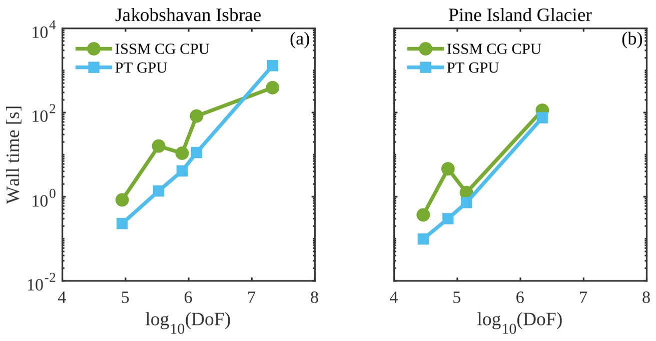

Figure 5. Performance comparison of the PT Tesla V100 implementation with the CPU implementation employing wall time (or computational time to reach convergence). Note that wall time does not include pre- and post-processing steps.

study (Räss et al., 2020, 2022), which is defined as follows:

$$T_{\mathrm{eff}} = \frac{n_{\mathrm{n}}\, n_{\mathrm{iter}}\, n_{\mathrm{IO}}\, n_{\mathrm{p}}}{1024^3\, t_{n_{\mathrm{iter}}}}\,, \tag{22}$$

where n_{n} represents the total number of vertices, n_{iter} is a given number of PT iterations, n_{p} is the arithmetic precision, $t_{n_{\mathrm{iter}}}$ is the time taken to complete n_{iter} iterations, and n_{IO} is the minimal number of nonredundant memory accesses (read and write operations). The number of read and write operations needed for this study would be eight: update v_x, v_y, and nonlinear viscosity arrays for every PT iteration, in addition to reading the basal friction coefficient and the masks.

4 Results and discussion

To investigate the performance of the PT CUDA C implementation on unstructured meshes, we report the number of vertices (or grid size) and the corresponding number of nonlinear PT iterations needed to meet the stopping criterion (Fig. 3). For both the Jakobshavn and Pine Island glacier models, the number of nonlinear PT iterations required to converge for a given number of vertices (N) scales in the order of $\approx \mathcal{O}(N^{1.2})$ or better. We chose the damping parameter γ, nonlinear viscosity relaxation scalar θ_μ, and transient *pseudo*-time step $\Delta\tau$ to maintain the linear scaling described above; optimal parameter values are listed in the Appendix (Table A1). We observed an exception at $\sim 3\times10^7$ degrees of freedom (DoFs) for the Pine Island Glacier model; optimal parameter values are unidentifiable. We will further investigate the convergence for the Pine Island Glacier model at $\sim 3\times10^7$ DoFs. Among the two glacier models chosen in this study, for a given number of vertices (N), Jakobshavn Isbræ resulted in faster convergence rates, which we attribute to differences in scale and bed topography and the nonlinearity of the problem (Fig. 4).

We further compare the performance of the PT CUDA C implementation with a standard finite-element CPU-based implementation using the price-to-performance metric. The price of a single Tesla V100 GPU is 1.5 times that of two Intel Xeon Gold 6140 CPU processors. [1] We expect a minimum speedup of at least 1.5 times to justify the Tesla V100 GPU price to performance. We recorded the computational time to reach convergence for the ISSM CG CPU and the PT GPU solver implementations (Fig. 5) for up to 2×10^7 DoFs. Across glacier configurations, we report a speedup of >1.5 times on the Tesla V100 GPU. We report a speedup of approximately 7 times at $\sim 1\times10^6$ DoFs for the Jakobshavn glacier model. This larger speedup at $\sim 1\times10^6$ DoFs indicates the PT GPU implementation's suitability to develop high-spatial-resolution ice-sheet flow models. We report an exception for the Jakobshavn glacier model at 2×10^7 DoFs (speedup of 0.28 times). We suggest that readers compare the speedup results reported in this study (Table 1) with other parallelization strategies.

The PT method applied to solve nonlinear momentum balance equations is a memory-bound algorithm, as described in Sect. 3.3. The profiling results on the Tesla V100 GPU indicate an up to 85 % increase in the device's available mem-

[1] Intel Xeon Gold 6140 processor specification sheet: https://ark.intel.com/content/www/us/en/ark/products/120485/intel-xeon-gold-6140-processor-24-75m-cache-2-30-ghz.html (last access: 31 December 2023).

Table 1. Performance comparison of the PT Tesla V100 implementation with the CPU implementation employing speedup S.

Jakobshavn Isbræ DoFs	Speedup S	Pine Island Glacier DoFs	Speedup S
88 458	3.6	28 920	3.73
329 362	11.68	71 292	15.30
787 542	2.64	139 578	1.73
1 335 458	7.37	2 221 410	1.5
21 328 514	0.299		

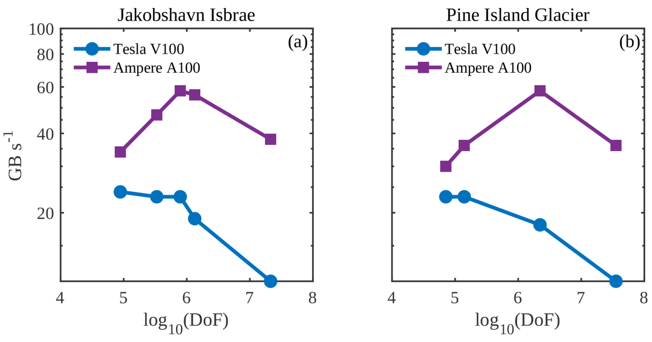

Figure 6. Performance assessment of the PT CUDA C implementation across GPU architectures employing effective memory throughput.

ory resources utilization with the increase in DoFs. This further confirms the memory-bounded nature of the implementation. To assess the performance of the memory-bound PT CUDA C implementation, we employ the effective memory throughput metric defined in Sect. 3.3. We report the effective memory throughput to DoFs for the PT CUDA C single-GPU implementation (Fig. 6). We observe a significant drop in effective memory throughput on both GPU architectures at DoFs $> \times 10^7$, which explains the drop in the speedup. We attribute the drop partly to the nonoptimal global memory access patterns reported in the L1TEX and L2 cache. We identify excessive nonlocal data access patterns in the ice stiffness and strain-rate computations, which involve accessing element-to-vertex connectivities and vice versa. For optimal or fully coalesced global memory access patterns, the threads in a warp must access the same relative address. We are investigating techniques to reduce the mesh non-localities and allow for coalesced global accesses.

We report a peak memory throughput for the NVIDIA Tesla V100 and A100 GPUs of 785 and 1536 GB s^{-1}, respectively. The peak memory throughput reflects the maximal memory transfer rates for performing memory copy-only operations. It represents the hardware performance limit in a memory-bound regime. Across glacier model configurations for the DoFs chosen in this study, the PT CUDA C implementation achieves a maximum of 23 and 58 GB s^{-1} for the NVIDIA Tesla V100 and A100 GPUs, respectively. The measured memory throughput is in the order of 500 GB s^{-1}, as reported by the NVIDIA Nsight Compute profiling tool 2022.2 on the NVIDIA Tesla V100. The measured memory throughput values reflect that we efficiently saturate the memory bandwidth. In contrast, the lower effective memory throughput values indicate that some of the memory accesses are redundant and could be further optimized.

Minimizing the memory footprint is critical when assessing the performance of memory-bounded algorithms, further speedups, and increased ability to solve large-scale problems. Due to insufficient memory at 1×10^8 DoFs for the Pine Island Glacier model configuration, we could neither execute the standard CPU solver on four 18-core Intel Xeon Gold 6140 processors and 3 TB of RAM nor the PT GPU implementation on the Tesla V100 GPU architecture. However, we could implement PT GPU on a single A100 SXM4 featuring 80 GB of device RAM. Thus, we could further confirm the necessity to keep the memory footprint minimal for models targeting high spatial resolution.

In this study, we tested up to an estimated 2×10^7 DoFs needed to maintain a spatial resolution of ~ 1 km or better in grounding-line regions for Antarctic and Greenland-wide ice flow models. Future studies may involve extending the PT CUDA C implementation from (i) the regional scale to the ice-sheet scale and from (ii) a 2-D SSA to a 3-D Blatter–Pattyn higher-order (HO) approximation. Extending the PT CUDA C implementation to the ice-sheet scale will require the user to carefully choose the damping parameter γ, nonlinear viscosity relaxation scalar θ_μ, and transient *pseudo*-time step $\Delta\tau$. The shared elliptical nature of the 2-D SSA and 3-D HO formulations and corresponding partial differential-equation-based models (Gilbarg and Trudinger, 1977; Tezaur et al., 2015) suggests the PT method's ability to solve the 3-D HO momentum balance applied to unstructured meshes. The overarching goal is to diminish spatial resolution constraints at higher computing performance to improve predictions of ice-sheet evolution.

5 Conclusions

Recent studies have implemented techniques that keep computational resources manageable at the ice-sheet scale while increasing the spatial resolution dynamically in areas where the grounding lines migrate during prognostic simulations (Cornford et al., 2013; Goelzer et al., 2017). In terms of computer memory footprint and execution time, the computational cost associated with solving the momentum balance equations to predict the ice velocity and pressure represents one of the primary bottlenecks (Jouvet et al., 2022). This preliminary study introduces a PT solver, applied to unstructured meshes, that leverages the GPU computing power to alleviate this bottleneck. Coupling the GPU-based ice velocity and pressure simulations with CPU-based ice thickness and temperature simulations can provide an enhanced balance between speed and predictive performance.

This study aimed to investigate the PT CUDA C implementation for unstructured meshes and its application to the 2-D SSA model formulation. For both the Jakobshavn and Pine Island glacier models, the number of nonlinear PT iterations required to converge for a given number of vertices (N) scales in the order of $\approx \mathcal{O}(N^{1.2})$ or better. We observed an exception at 3×10^7 degrees of freedom (DoFs) for the Pine Island Glacier model; optimal solver parameters are unidentifiable. We further compare the performance of the PT CUDA C implementation with a standard finite-element CPU-based implementation using the price-to-performance metric. We justify the GPU implementation in the price-to-performance metric for up to million-grid-point spatial resolutions.

In addition to the price-to-performance metric, we preliminary investigated the power consumption. The power consumption of the PT GPU implementation was measured using the NVIDIA System Management Interface 460.32.03. For the range of DoFs tested, the power usage for both glacier configurations to meet the stopping criterion was 38 ± 1 W. The power consumption measurement for the CPU implementation was taken from the hardware specification sheet: thermal design power. For a 64-bit 18-core Intel Xeon Gold 6140 processor, the thermal design power is 140 W.[2] We executed the CPU-based multicore MPI-parallelized ice-sheet flow simulations on two CPUs, with all 36 cores enabled, and we chose the power consumption to be 280 W. This is a first-order estimate. Thus, the power consumption of the PT GPU implementation was approximately one-seventh of the traditional CPU implementation for the test cases chosen in this study. We will investigate this further.

This study represents a first step toward leveraging GPU processing power, enabling more accurate polar ice discharge predictions. The insights gained will benefit efforts to diminish spatial resolution constraints at higher computing performance. The higher computing performance will allow users to run ensembles of ice-sheet flow simulations at the continental scale and higher resolution, a previously challenging task. The advances will further enable the quantification of model sensitivity to changes in upcoming climate forcings. These findings will significantly benefit process-oriented sea-level-projection studies over the coming decades.

[2]Intel Xeon Gold 6140 processor specification sheet: https://ark.intel.com/content/www/us/en/ark/products/120485/intel-xeon-gold-6140-processor-24-75m-cache-2-30-ghz.html (last access: 31 December 2023).

Appendix A

Table A1. Optimal combination of damping parameter γ, nonlinear viscosity relaxation scalar θ_μ, and relaxation θ_v to maintain the linear scaling and solution stability for the glacier model configurations and DoFs listed.

Jakobshavn Isbræ DoFs	γ	θ_v	θ_μ	Pine Island Glacier DoFs	γ	θ_v	θ_μ
88 458	0.98	0.99	3×10^{-2}	28 920	0.98	0.6	1×10^{-1}
329 362	0.987	0.98	7×10^{-2}	71 292	0.99	0.49	8×10^{-2}
787 542	0.99	0.99	1×10^{-1}	139 578	0.991	0.99	2×10^{-2}
1 335 458	0.992	0.999	1×10^{-1}	2 221 410	0.998	0.995	1×10^{-2}
21 328 514	0.998	0.999	1×10^{-1}				

Code and data availability. The current version of FastIceFlo is available for download from GitHub at https://github.com/AnjaliSandip/FastIceFlo (last access: 18 September 2023) under the MIT license. The exact version of the model used to produce the results used in this paper is archived on Zenodo (https://doi.org/10.5281/zenodo.8356351, Sandip et al., 2023) along with the input data and scripts to run the model and produce the plots for all of the simulations presented in this paper. The PT CUDA C implementation runs on a CUDA-capable GPU device. The research data are presented in the paper.

Author contributions. AS developed the PT CUDA C implementation, conducted the performance assessment tests described in the paper, and was responsible for data analysis. LR provided guidance on the early stages of the mathematical reformulation of the 2-D SSA model to incorporate the PT method and supported the PT CUDA C implementation. MM reformulated the 2-D SSA model to incorporate the PT method, developed the weak formulation, and wrote the first versions of the code in MATLAB and then C. All authors participated in writing the manuscript.

Competing interests. At least one of the (co-)authors is a member of the editorial board of *Geoscientific Model Development*. The peer-review process was guided by an independent editor, and the authors also have no other competing interests to declare.

Acknowledgements. We thank Dan Martin and the anonymous reviewer for their valuable feedback that enhanced the study. We acknowledge the University of North Dakota Computational Research Center for computing resources on the Talon cluster and are grateful to David Apostal and Aaron Bergstrom for technical support. We thank the NVIDIA Applied Research Accelerator Program for hardware support. We are also grateful to the NVIDIA solution architects, Oded Green, Zoe Ryan, and Jonathan Dursi, for their thoughtful input. Ludovic Räss thanks Ivan Utkin for helpful discussions and acknowledges the Laboratory of Hydraulics, Hydrology, and Glaciology (VAW) at ETH Zurich for computing access to the Superzack GPU server. CPU and GPU hardware architectures were made available through the first author's access to the Talon (University of North Dakota) and Curiosity (NVIDIA) clusters.

Review statement. This paper was edited by Philippe Huybrechts and reviewed by Daniel Martin and one anonymous referee.

References

Aschwanden, A., Bartholomaus, T. C., Brinkerhoff, D. J., and Truffer, M.: Brief communication: A roadmap towards credible projections of ice sheet contribution to sea level, The Cryosphere, 15, 5705–5715, https://doi.org/10.5194/tc-15-5705-2021, 2021.

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

Castleman, B. A., Schlegel, N.-J., Caron, L., Larour, E., and Khazendar, A.: Derivation of bedrock topography measurement requirements for the reduction of uncertainty in ice-sheet model projections of Thwaites Glacier, The Cryosphere, 16, 761–778, https://doi.org/10.5194/tc-16-761-2022, 2022.

Chen, X., Zhang, X., Church, J. A., Watson, C. S., King, M. A., Monselesan, D., Legresy, B., and Harig, C.: The increasing rate of global mean sea-level rise during 1993–2014, Nat. Clim. Change, 7, 492–495, https://doi.org/10.1038/nclimate3325, 2017.

Cornford, S. L., Martin, D. F., Graves, D. T., Ranken, D. F., Le Brocq, A. M., Gladstone, R. M., Payne, A. J., Ng, E. G., and Lipscomb, W. H.: Adaptive mesh, finite volume modeling of marine ice sheets, J. Comput. Phys., 232, 529–549, https://doi.org/10.1016/j.jcp.2012.08.037, 2013.

Duretz, T., Räss, L., Podladchikov, Y., and Schmalholz, S.: Resolving thermomechanical coupling in two and three dimensions: spontaneous strain localization owing to shear heating, Geophys. J. Int., 216, 365–379, https://doi.org/10.1093/gji/ggy434, 2019.

Frankel, S. P.: Convergence rates of iterative treatments of partial differential equations, Math. Comput., 4, 65–75, 1950.

Gilbarg, D. and Trudinger, N. S.: Elliptic partial differential equations of second order, vol. 224, Springer, https://doi.org/10.1007/978-3-642-61798-0, 1977.

Glen, J. W.: The creep of polycrystalline ice, P. Roy. Soc. Lond.-Ser. A, 228, 519–538, https://doi.org/10.1098/rspa.1955.0066, 1955.

Goelzer, H., Robinson, A., Seroussi, H., and Van De Wal, R. S.: Recent progress in Greenland ice sheet modelling, Curr. Clim. Change Rep., 3, 291–302, https://doi.org/10.1007/s40641-017-0073-y, 2017.

Habbal, F., Larour, E., Morlighem, M., Seroussi, H., Borstad, C. P., and Rignot, E.: Optimal numerical solvers for transient simulations of ice flow using the Ice Sheet System Model (ISSM versions 4.2.5 and 4.11), Geosci. Model Dev., 10, 155–168, https://doi.org/10.5194/gmd-10-155-2017, 2017.

Häfner, D., Nuterman, R., and Jochum, M.: Fast, cheap, and turbulent – Global ocean modeling with GPU acceleration in python, J. Adv. Model. Earth Sy., 13, e2021MS002717, https://doi.org/10.1029/2021MS002717, 2021.

Hinkel, J., Lincke, D., Vafeidis, A. T., Perrette, M., Nicholls, R. J., Tol, R. S., Marzeion, B., Fettweis, X., Ionescu, C., and Levermann, A.: Coastal flood damage and adaptation costs under 21st century sea-level rise, P. Natl. Acad. Sci. USA, 111, 3292–3297, https://doi.org/10.1073/pnas.1222469111, 2014.

IPCC: Climate Change 2021 The Physical Science Basis, Working Group 1 (WG1) Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change, Tech. rep., Intergovernmental Panel on Climate Change, 2021.

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, J. Glaciol., 68, 651–664, https://doi.org/10.1017/jog.2021.120, 2022.

Kelley, C. and Liao, L.-Z.: Explicit pseudo-transient continuation, Computing, 15, 18, 2013.

Kopp, R. E., Kemp, A. C., Bittermann, K., Horton, B. P., Donnelly, J. P., Gehrels, W. R., Hay, C. C., Mitrovica, J. X., Morrow, E. D., and Rahmstorf, S.: Temperature-driven global sea-level variability in the Common Era, P. Natl. Acad. Sci. USA, 113, E1434–E1441, https://doi.org/10.1073/pnas.1517056113, 2016.

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), J. Geophys. Res.-Earth, 117, 22, https://doi.org/10.1029/2011JF002140, 2012.

MacAyeal, D. R.: Large-scale ice flow over a viscous basal sediment: Theory and application to ice stream B, Antarctica, J. Geophys. Res.-Sol. Ea., 94, 4071–4087, 1989.

Morlighem, M., Williams, C. N., Rignot, E., An, L., Arndt, J. E., Bamber, J. L., Catania, G., Chauché, N., Dowdeswell, J. A., Dorschel, B., Fenty, I., Hogan, K., Howat, I., Hubbard, A., Jakobsson, M., Jordan, T. M., Kjeldsen, K. K., Millan, R., Mayer, L., Mouginot, J., Noel, B. P. Y., O'Cofaigh, C., Palmer, S., Rysgaard, S., Seroussi, H., Siegert, M. J., Slabon, P., Straneo, F., Van Den Broeke, M. R., Weinrebe, W., Wood, M., and Zinglersen, K. B.: BedMachine v3: Complete bed topography and ocean bathymetry mapping of Greenland from multibeam echo sounding combined with mass conservation, Geophys. Res. Lett., 44, 11–51, https://doi.org/10.1002/2017GL074954, 2017.

Morlighem, M., Rignot, E., Binder, T., Blankenship, D., Drews, R., Eagles, G., Eisen, O., Ferraccioli, F., Forsberg, R., Fretwell, P., Goel, V., Greenbaum, J. S., Gudmundsson, H., Guo, J., Helm, V., Hofstede, C., Howat, I., Humbert, A., Jokat, W., Karlsson, N., Lee, W. S., Matsuoka, K., Millan, R., Mouginot, J., Paden, J., Pattyn, F., Roberts, J., Rosier, S., Ruppel, A., Seroussi, H., Smith, E. C., Steinhage, D., Sun, B., Van Den Broeke, M. R., Van Ommen, T. D., Wessem, M. V., and Young, D. A.: Deep glacial troughs and stabilizing ridges unveiled beneath the margins of the Antarctic ice sheet, Nat. Geosci., 13, 132–137, https://doi.org/10.1038/s41561-019-0510-8, 2020.

Omlin, S., Räss, L., and Podladchikov, Y. Y.: Simulation of three-dimensional viscoelastic deformation coupled to porous fluid flow, Tectonophysics, 746, 695–701, https://doi.org/10.1016/j.tecto.2017.08.012, 2018.

Pattyn, F. and Morlighem, M.: The uncertain future of the Antarctic Ice Sheet, Science, 367, 1331–1335, https://doi.org/10.1126/science.aaz5487, 2020.

Poliakov, A. N., Cundall, P. A., Podladchikov, Y. Y., and Lyakhovsky, V. A.: An explicit inertial method for the simulation of viscoelastic flow: an evaluation of elastic effects on diapiric flow in two- and three- layers models, Flow and creep in the solar system: observations, modeling and theory, in: Flow and Creep in the Solar System: Observations, Modeling and Theory, edited by: Stone, D. B. and Runcorn, S. K., NATO ASI Series, vol. 391, Springer, Dordrecht, 175–195, https://doi.org/10.1007/978-94-015-8206-3_12, 1993.

Räss, L., Duretz, T., and Podladchikov, Y.: Resolving hydromechanical coupling in two and three dimensions: spontaneous channelling of porous fluids owing to decompaction weakening, Geophys. J. Int., 218, 1591–1616, https://doi.org/10.1093/gji/ggz239, 2019.

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), Geosci. Model Dev., 13, 955–976, https://doi.org/10.5194/gmd-13-955-2020, 2020.

Räss, L., Utkin, I., Duretz, T., Omlin, S., and Podladchikov, Y. Y.: Assessing the robustness and scalability of the accelerated pseudo-transient method, Geosci. Model Dev., 15, 5757–5786, https://doi.org/10.5194/gmd-15-5757-2022, 2022.

Rietbroek, R., Brunnabend, S.-E., Kusche, J., Schröter, J., and Dahle, C.: Revisiting the contemporary sea-level budget on global and regional scales, P. Natl. Acad. Sci. USA, 113, 1504–1509, https://doi.org/10.1073/pnas.1519132113, 2016.

Rignot, E., Mouginot, J., and Scheuchl, B.: Antarctic grounding line mapping from differential satellite radar interferometry, Geophys. Res. Lett., 38, 49, https://doi.org/10.1029/2011GL047109, 2011.

Sandip, A., Morlighem, M., and Räss, L.: AnjaliSandip/-FastIceFlo: FastIceFlo v1.0.1 (v1.0.1), Zenodo [code], https://doi.org/10.5281/zenodo.8356351, 2023.

Tezaur, I. K., Perego, M., Salinger, A. G., Tuminaro, R. S., and Price, S. F.: Albany/FELIX: a parallel, scalable and robust, finite element, first-order Stokes approximation ice sheet solver built for advanced analysis, Geosci. Model Dev., 8, 1197–1220, https://doi.org/10.5194/gmd-8-1197-2015, 2015.