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)

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 ~~(1 km or better)~~under the kilometer scale. Conventional ice flow models ~~may need significant computational resources because these models mainly execute Central Processing Units~~ mainly execute on central processing units (CPUs), which ~~lack massive parallelism capabilities and feature limited~~ feature limited parallel processing capabilities and peak memory bandwidth. ~~On the other side of the spectrum, Graphics Processing Units~~ 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 ~~at every grid point~~ can be performed concurrently by thousands of threads~~or parallel workers~~, processing most of the computational domain simultaneously. In this study, we combine ~~GPUs~~ a GPU-based approach with the pseudo-transient (PT) method, an accelerated iterative and matrix-free ~~solving approach~~solution strategy, and investigate its performance for finite elements and unstructured meshes ~~applied~~ with application to two-dimensional (2-D) models of real glaciers at a regional scale. For both Jakobshavn and Pine Island glacier models, the number of nonlinear PT iterations ~~to converge for~~ required to converge a given number of vertices N scales in the order of $\mathcal{O}(N^{1.2})$ or better. We ~~compared~~ further compare the performance of the PT CUDA C implementation with a standard finite-element CPU-based implementation using the ~~metric: price and power consumption to performance. The~~ price-to-performance metric. The price of a single Tesla V100 GPU is 1.5 times ~~the price of the~~ that of two Intel Xeon Gold 6140 ~~CPU processors. The power consumption of the PT CUDA C implementation was approximately one-seventh of the standard CPU implementation for the test cases chosen in this study~~CPUs. We expect a minimum speed-up of ~~>~~at least 1.5x to justify the Tesla V100 GPU price to performance. ~~We report the performance (or the~~ Our developments result in a GPU-based implementation that achieves this goal with a speed-up ~~) across glacier configurations for degrees of freedom (DoFs) tested to be >~~beyond 1.5~~on a Tesla V100~~x. This study ~~is~~ represents a first step toward leveraging GPU processing power~~for~~ , enabling more accurate polar ice discharge predictions. The insights gained ~~from this study~~ will benefit efforts to diminish spatial resolution constraints at ~~increased computing speed. The increased computing speed~~ higher computing

performance. The higher computing performance will allow running ensembles of ice-sheet flow simulations at the continental scale and ~~at high resolution, previously not possible, enabling~~ higher resolution, a previously challenging task. The advances will further enable quantification of model sensitivity to changes in ~~future~~ upcoming climate forcings. These findings will ~~be significantly beneficial for~~ significantly benefit process-oriented ~~and~~ 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 ~~remain poorly known~~ 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 ~~better~~ smaller (Larour et al., 2012; Aschwanden et al., 2021; Castleman et al., 2022).

Most numerical models ~~are designed for~~ use a solution strategy designed to target central processing units (CPU) ~~. The performance of CPUsis slowly leveling off~~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 ~~be available~~ become feasible at the continental scale (or ice-sheet scale)~~, especially for~~ . Specifically, complex flow models, such as Full-Stokes, ~~which~~ may remain challenging to employ beyond the regional scale. ~~On the other side of the spectrum~~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 ~~power has become an essential aspect of numerical computing~~ capabilities has become essential and has resulted in active development within scientific computing and high-performance computing (HPC) communities.

~~However, the~~ The traditional way of solving the ~~governing equations of~~ partial differential equations governing ice-sheet flow, ~~such as~~ employing, e.g., the finite-element analysis, ~~is not adapted to GPUs as they cannot handle large sparse matrices and linear solvers~~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 re-formulating the flow equations in the form of ~~a~~ pseudo-transient (PT) ~~model~~updates. The PT method augments ~~physically motivated time-dependent inertial termsto targeted~~ the time-independent ~~equations, which is the case of the ice sheet stress balance. These equations are explicitly and iteratively updated in~~ governing ice-sheet flow equation by physically motivated pseudo-time-dependent terms. The added pseudo-time τ ~~until we~~ 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 ~~, which provides a solution to the initial time-independent equations~~and, thus, the solution of the initial elliptic problem. The explicit ~~update~~ pseudo-time integration scheme eliminates the need for ~~an expensive stiffness matrix assembly, making it~~ the expensive direct-iterative type of solvers, making the proposed approach matrix-free and ~~suitable for parallel implementations and most operations would be identical for each grid point~~ attractive for various parallel computing approaches (Frankel, 1950; Poliakov et al., 1993; Kelley and Liao, 2013). Räss et al. (2020) introduced this method ~~and combined its implementation with GPUs~~ targeting specifically GPU computing to enable the development of high spatial resolution full Stokes ice-sheet flow ~~models, two-dimensional~~ solvers in two (2-D) and ~~three-dimensional~~ three-dimensions (3-D), respectively, on uniform grids (Räss et al., 2020). ~~Among the study's limitations are the proposed finite differences numerical scheme, the uniform structured grid, and the applications limited to simple idealized cases~~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 ~~extend~~ build upon work from previous studies (Räss et al., 2019, 2020) on the accelerated PT method ~~to finite elements on~~ 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 ~~apply it to real glaciers at a regional scale :~~ applied it to regional scale glaciers, Pine Island Glacier ~~, in west Antarctica,~~ and Jakobshavn Isbræ, in ~~western Greenland~~West Antarctica and Greenland, respectively. We compare the PT CUDA C implementation with ~~the~~ a more standard finite-element ~~CPU implementation available in~~ CPU-based implementation available within the Ice-sheet and Sea-level System Model (ISSM)~~using~~. Our comparison uses the same mesh, model equations, and boundary conditions. In Section 2, we present the mathematical reformulation of the 2D SSA momentum balance equations to incorporate the ~~PT method followed by its~~ 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 Sections 4 and 5, we illustrate the method's performance and conclude ~~with~~ on future research directions.

2 Methods

2.1 Mathematical formulation of 2D SSA model

We employ the SSA (MacAyeal, 1989)) formulation to solve the momentum balance ~~. The SSA equations in the matrix form read :~~ equation:

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

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

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

where B is ice rigidity, $\dot{\varepsilon}_e$ 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 $1e^5$ to address the singularity arising in regions of the computational domain where the strain-rate tends toward zero.

~~In terms of~~ As boundary conditions, we apply water pressure at the ice front Γ_σ, and non-homogeneous Dirichlet boundary conditions Γ_u on the other boundaries ~~Γ_u~~ (based on observed velocity).

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

The solution to ~~Eq.~~ the SSA ice flow problem is commonly achieved by discretizing Eq. (1) using the finite-element or finite-difference method. ~~These methods require to spatially discretize the domain and build a linear system of equations that is then solved using either a director iterative solver~~The discretized problem can be solved using a direct, direct-iterative, or iterative approach. Robust matrix-based direct-type of solvers exhibit significant scaling limitations restricting their applicability when considering high-resolution or three-dimensional configurations. Iterative solving approaches permit circumventing most ~~of the~~ scaling limitations. ~~The challenge is instead~~ However, they may encounter convergence issues for sub-optimally conditioned, stiff, or highly nonlinear problems, resulting in non-tractable growth of the iteration count. One challenge is thus 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 equations (1) by adding ~~a transient term. One can then use the transient term~~ the usually ignored transient term, which can be further used to integrate the equations in pseudo-time τ, seeking at an implicit solution once the steady state is reached~~at~~, i.e., when $\tau \to \infty$.

Building upon work from previous studies ~~(Omlin et al., 2018; Duretz et al., 2019; Räss et al., 2019)~~(Omlin et al., 2018; Duretz et al., 20, we reformulate the 2-D SSA steady-state momentum balance equations ~~to incorporate the usually ignored inertial terms:~~

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

~~allows us to turn the steady-state equations into transient diffusion of~~ into a transient diffusion-like formulation for flow velocities $v_{x,y}$ ~~. The velocity time~~ by incorporating the usually omitted time derivative.

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

The velocity-time derivatives represent physically motivated expressions we can further use to iteratively reach a ~~steady-state~~steady state, which provides the solution of the original time-independent equations. Since we are here only interested in the steady-

state, 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 we seek a solutionfor for which we are seeking a solution. We define the transient *pseudo* time step $\Delta\tau$ as a field variable, spatially variable, chosen to minimize the number of nonlinear PT iterations.

2.3 Pseudo-time stepping

We can here advance in numerical pseudo-time using a forward Euler pseudo-time stepping method. We choose our inertial term 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 recognises the diffusion of diffusive variables $v_{x,y}$ and the effective dynamic viscosity $4\mu/\rho$ as diffusion coefficient. Using the diffusion analogy analogy of a diffusive process, we can use this information to define the define a CFL-like (Courant-Friedrich-Lewy) stability criterion for the PT iterative scheme. The explicit CFL CFL-stable time step for viscous flow is given by:

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

where Δx is represents the grid spacing, μ_b is the numerical bulk ice viscosity and $n_{\dim} = 2.1, 4.1, 6.1$ in 1, 2 and 3D, 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 to diverge 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 ~~to~~ 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~~ wave-like pseudo-physics exhibits two main advantages: (i) the wave-like time step ~~limiter is~~ limit is a function of Δx instead of Δx^2, and (ii) it is possible to turn the wave equation into a damped wave equation. The latter permits finding optimal tuning ~~parameter~~ parameters to achieve optimal damping, resulting in fast convergence. Let's assume the following diffusion-like update step, reported here for the x direction only:

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

The above expression ~~would result~~ results in the following update rule:

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

where $\Delta\tau_\mathrm{D} \approx \Delta x^2/(4\mu/\rho)/4.1$ is the diffusion-like time step ~~limiter~~limit. This system can be ~~rewritten in wave-equation style:~~

$$A_x = A_x^{\mathrm{old}} + \frac{\Delta\tau_\mathrm{w}}{\rho} \left(\frac{\partial}{\partial x} 4 \mu \frac{\partial v_x}{\partial x} \right)$$

$$v_x = v_x^{\mathrm{old}} + \Delta\tau_\mathrm{w} A_x$$

separated into a residual assignment A_x and the velocity update v_x:

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

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

~~where $\Delta\tau_\mathrm{w} \approx \Delta x/\sqrt{4\mu/\rho}/2.1$ is the wave-like time step limiter where one recognises a term analogous to the numerical wave velocity . Moreover, the~~ Converting Eq. (11) ~~can be damped by adding friction γ to the current A_x:~~

$$A_x = A_x^{\mathrm{old}}(1-\gamma) + \frac{\Delta\tau_\mathrm{w}}{\rho} \left(\frac{\partial}{\partial x} 4 \mu \frac{\partial v_x}{\partial x} \right)$$

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

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

turns the system composed of Eqs (12) and (13) in 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:

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

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)~~Räss et al. (2019), Duretz et al. (2019), ~~Räss et al. (2020)~~ Räss et al. (2020) while an in-depth analysis is provided in Räss et al. (2022).

~~Glacier model configurations; observed surface velocities in m yr^{-1} interpolated on a uniform mesh. Panels **(a)** and **(b)** correspond to Jakobshavn Isbræand Pine Island Glacier respectively.~~

2.6 Weak formulation and finite-element discretization

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

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

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

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

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

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 ~~is:~~ reads:

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

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

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

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

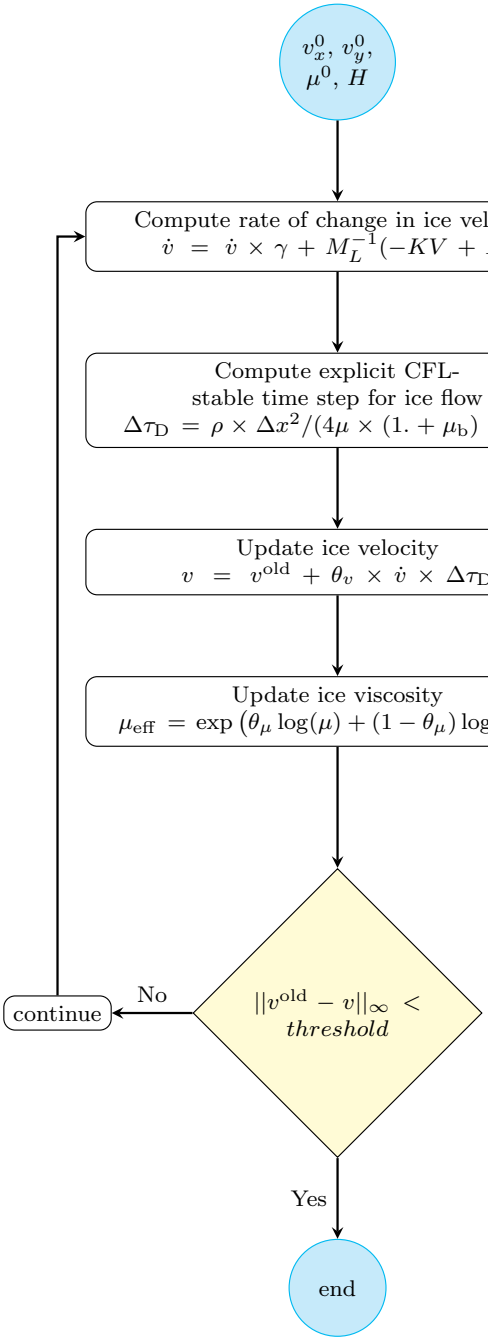

Figure 1. PT iterative algorithm for unstructured meshes applied to solve 2-D SSA momentum balance equations.

where $\mathbf{M}_L$ ~~a~~ stands for the lumped mass matrix permitting to avoid the resolution of a matrix system.

We hence have an explicit expression of the time derivative of the ice velocity for each vertex of the mesh:

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

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

where m_{Li} is the component number i along the diagonal of the lumped mass matrix M_L.

For every nonlinear PT iteration, we compute the rate of change in velocity $\dot{\mathbf{v}}$ and the explicit ~~CFL~~ CFL-stable time step $\Delta\tau$. We then deploy the reformulated 2D SSA momentum balance equations to update ice velocity $\mathbf{v}$ 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~~ Western Greenland, and Pine Island Glacier, in ~~west~~ 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 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).

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 $||v^{old} - v||_\infty < 10$ m yr^{-1}. The software solves the 2-D SSA momentum balance equations on a single GPU. We use ~~here~~ an NVIDIA Tesla V100 SXM2 GPU ~~featuring~~ with 16 gigabytes (GB) ~~onboard memory and an Ampere~~ of device RAM and an NVIDIA A100 SXM4 ~~featuring~~ with 80GB ~~onboard memory~~of device RAM. We compare the PT implementation's results on a Tesla V100 GPU ~~implementation~~ with ISSM's "standard" CPU implementation using a conjugate gradient (CG) iterative solver. ~~As a CPU, we~~ We used a 64-bit 18-core Intel Xeon Gold 6140 processor ~~with~~ for the CPU comparison, having 192 GB of RAM ~~per vertex. We executed~~ available. We perform multi-core MPI-parallelized ice-sheet flow simulations on two CPUs, all 36 cores enabled (Larour et al., 2012; Habbal et al., 2017). ~~We performed computations using~~ All simulations use double-precision arithmetic computations.

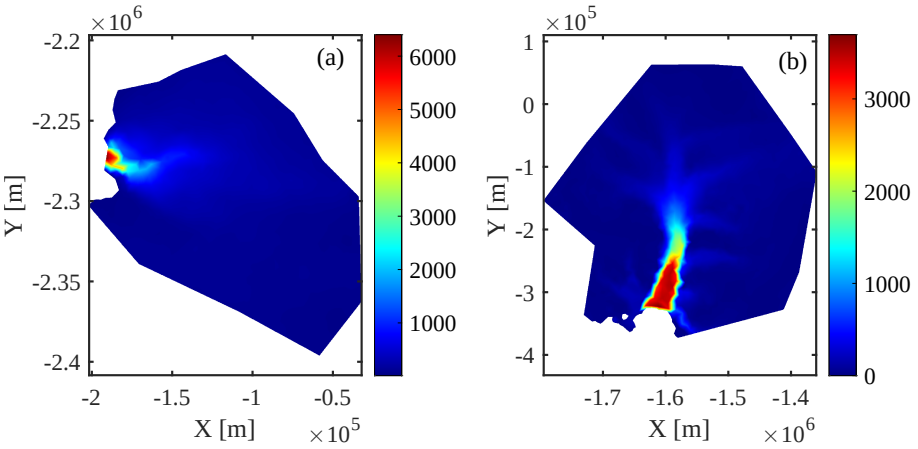

Figure 2. ~~Performance assessment of PT CUDA C implementation for unstructured meshes~~Glacier model configurations; observed surface velocities in m yr^{-1} interpolated on a uniform mesh. Panels **(a)** and **(b)** correspond to Jakobshavn Isbræ and Pine Island Glacier respectively.

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 ensure to exclude from timing all pre- and post-processing steps. We quantify the relative performance of the CPU and GPU implementations as the speed-up S, given by:

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

The PT method ~~applied to solve~~ employed to solve the nonlinear momentum balance equations ~~is~~ results in a memory-bound algorithm (Räss et al., 2020); therefore, the wall time depends on the memory throughput. ~~Hence,~~ In addition to speed-up, we employ the effective memory throughput metric to assess the performance of the PT CUDA C implementation developed in this study ~~, in addition to speed-up, we employ the effective memory throughput metric (Räss et al., 2020, 2022), defined as:~~ (Räss et al., 2020, 2022), which defines as:

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

where n_{n} represents the total number of vertices, n_{iter} a given number of PT iterations, n_{p} the arithmetic precision, $t_{\mathrm{n_{iter}}}$ the time taken to complete n_{iter} iterations and n_{IO} the minimal number of non-redundant memory accesses (read/write operations). The number of reads and write operations needed for this study would be 8~~;~~: 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 ~~for~~ 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 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 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 associated GitHub repository. We observed an exception at $\sim 3e^7$ degrees of freedom (DoFs) for the Pine Island glacier model; optimal ~~solver parameters~~ parameter values are unidentifiable. We will investigate further the convergence for the Pine Island glacier model at $\sim 3e^7$ DoFs~~in the following steps~~. 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).

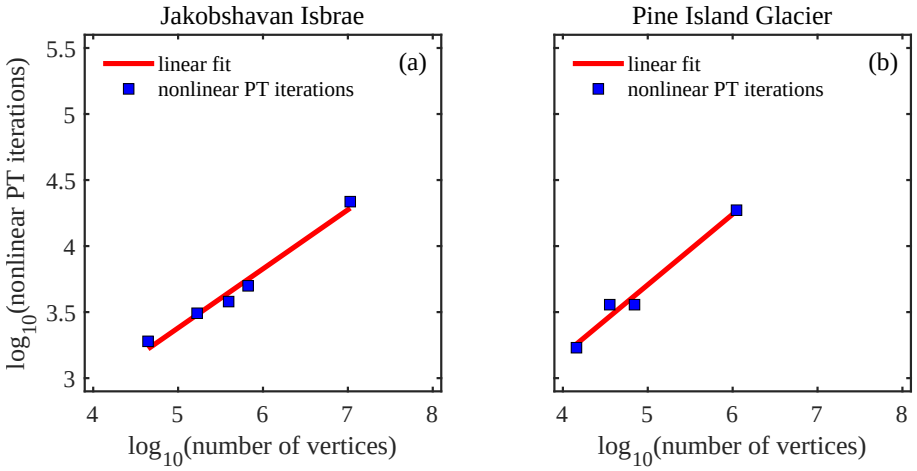

Figure 3. Performance assessment of the PT CUDA C implementation for unstructured meshes.

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

Jakobshavn IsbræDoFs	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		

~~To assess and compare the PT GPU implementation with the traditional matrix-based CPU implementation, we employed price and power consumption to performance as a~~ We further compare the performance of the PT CUDA C implementation

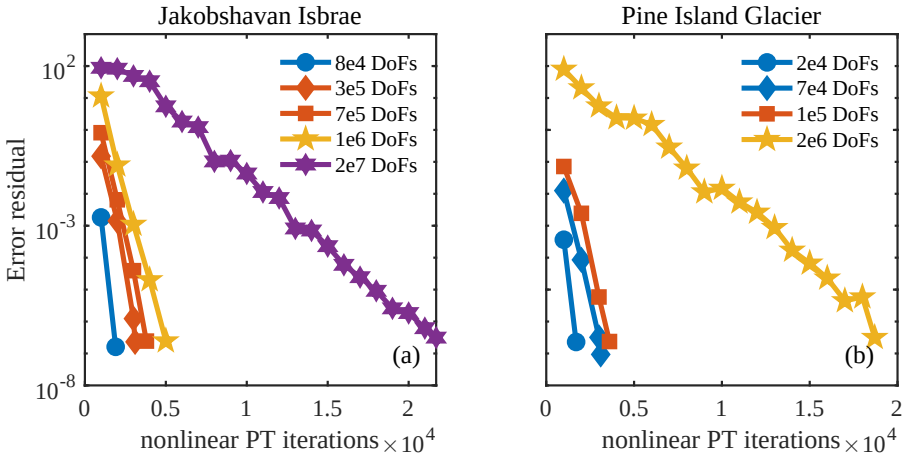

Figure 4. Residual error evolution of the PT CUDA C implementation for unstructured meshes.

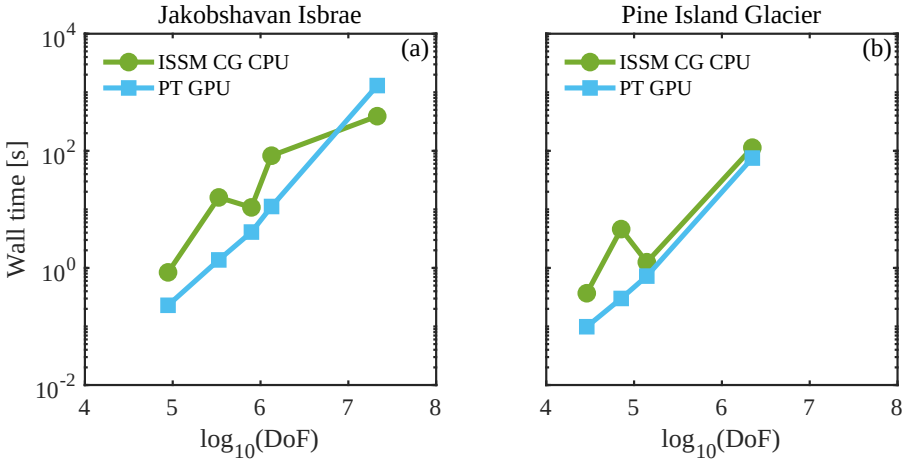

Figure 5. Performance comparison of the PT Tesla V100 implementation with the CPU implementation employing wall time (or computational time to reach convergence). Note that wall time does not include pre- and post-processing steps.

.

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 ~~the price of the~~ that of two Intel Xeon Gold 6140 CPU processors ~~chosen in this study. The power consumption of the PT GPU implementation was measured using NVIDIA System Management Interface. For the range of DoFs tested, the power usage for both glacier configurations to meet the stopping criterion was 38 ± 1 W. 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. We executed the CPU-based multi-core MPI-parallelized ice-sheet flow simulations on two CPUs, all 36 cores enabled, and we chose the power consumption to be 280~~

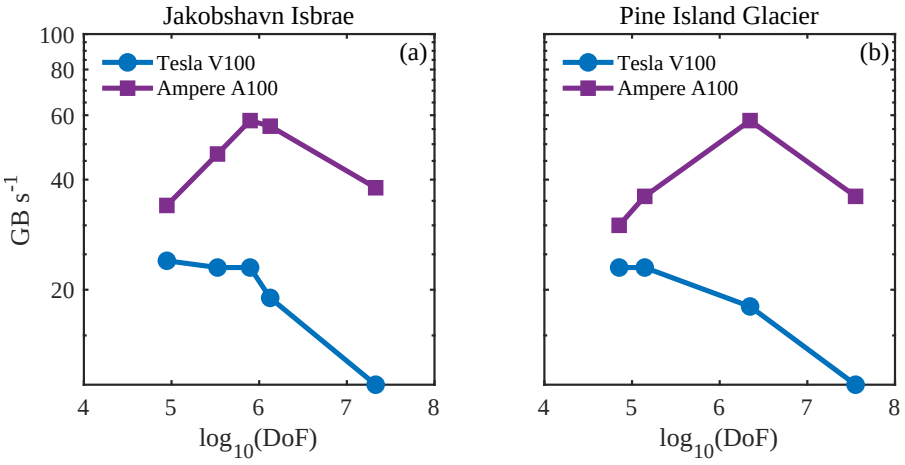

Figure 6. Performance assessment of the PT CUDA C implementation across GPU architectures employing effective memory throughput.

.

~~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.~~

~~As described in Section 3.3, we quantify the relative performance of the CPU and GPU implementations as the speed-up S~~[1].

We expect a minimum ~~speed-up of >~~speedup of at least 1.5x 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 $2e^7$ DoFs~~tested~~. Across glacier configurations, we ~~reported a speed-up~~ report a speedup of >1.5 on the Tesla V100 GPU. We ~~reported a speed-up~~ report a speedup of approximately $7\times$ at $\sim 1e^6$ DoFs for the Jakobshavn glacier model. This ~~high speed-up~~ larger speedup at $\sim 1e^6$ DoFs indicates PT GPU implementation's suitability to develop high spatial resolution ice-sheet flow models. We ~~reported~~ report an exception for the Jakobshavn glacier model at $2e^7$ DoFs, ~~speed-up~~ speedup of $0.28\times$. We suggest ~~that~~ readers compare the ~~speed-up~~ speedup results reported in this study ~~with other approaches to parallelization~~ (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. On the Tesla V100s, with the increase in DoFs, the profiling results indicated an increased utilization of~~ Section 3.3. The profiling results on the Tesla V100 GPU indicate an up to 85% increase in the device's available memory resources ~~, up to 85%, further confirming~~ 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.~~ Section 3.3. We report the effective memory throughput to DoFs for the PT CUDA C single GPU implementation (Fig. 6). We ~~observed~~ observe a significant drop in effective memory throughput on both GPU architectures at DoFs $> e^7$, which explains the drop in speed-up. We attribute the drop partly to the non-optimal global memory

[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

access patterns reported in the LITEX and L2 cache. We ~~identified data non-localities in~~ identify excessive non-local data access patterns in the ice stiffness and ~~strain rate computationsinvolving~~ strain-rate computations, which involve accessing element-to-vertex ~~connectivity~~ connectivities and vice versa~~as the primary source~~. 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.

~~The reported~~ We report a peak memory throughput for the ~~GPU hardware~~ NVIDIA Tesla V100 and ~~NVIDIA Ampere~~ A100 ~~was~~ GPUs of 785 GB s^{-1} and 1536 GB s^{-1}, respectively. The peak memory throughput ~~only reports the memory transfer speed~~ reflects the maximal memory transfer rates for performing memory ~~copy~~ 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 GB s^{-1} ~~for the Tesla V100~~ and 58 GB s^{-1} for the ~~Ampere~~ NVIDIA Tesla V100 and A100 ~~. Thus the PT CUDA C implementation reaches 3% and 4% of peak hardware value on the~~ 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~~and the Ampere A100, demonstrating the potential for increase in~~ . The measured memory throughput values reflect that we efficiently saturate the memory bandwidth. In contrast, the lower effective memory throughput ~~through better GPU resource exploitation. Future studies will include techniques to improve memory throughput and increase data locality~~values indicate that part of the memory accesses are redundant and could be further optimized.

Minimizing the memory footprint is critical ~~to the~~ when assessing the performance of memory-bounded ~~algorithm's performance , better~~ algorithms, further speed-ups, and increased ability to solve large-scale problems. Due to insufficient memory at $1e^8$ DoFs ~~with~~ for the Pine Island Glacier model configuration, we could neither execute the standard CPU ~~implementation on a~~ solver on four 18-core Intel Xeon Gold 6140 ~~processor with~~ processors and 3TB of RAM ~~per vertex~~ nor the PT GPU implementation on the Tesla V100 GPU architecture. However, we could ~~execute the PT GPU implementation~~ implement PT GPU on a single ~~Ampere~~ A100 SXM4 featuring 80GB ~~onboard memory, further confirming the need to keep~~ of device RAM. We could thus further confirm the necessity of keeping the memory footprint minimal for ~~high-spatial-resolution models~~ models targeting high spatial resolution.

In this ~~preliminary~~ study, we tested up to an estimated $2e^7$ DoFs needed to maintain a spatial resolution of $\sim$1 km or better in grounding line regions for Antarctic and Greenland-wide ice flow models. Future studies ~~would~~ may involve extending the PT CUDA C implementation from (i) regional scale to ice-sheet scale and (ii) 2-D SSA to 3-D Blatter-Pattyn ~~Higher-order~~ higher-order (HO) approximation. To extend the PT CUDA C implementation to ice-sheet scale ~~, we will~~ will require to carefully choose the damping parameter γ, nonlinear viscosity relaxation scalar θ_μ, and transient *pseudo* time step $\Delta\tau$~~carefully~~. The shared elliptical nature of the 2-D SSA and 3-D HO formulations and corresponding partial differential equations ~~(Gilbarg and Trudinger, 1977; Tezaur et al., 2015) confirms~~ 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 ~~increased computing speed for improved predictions of grounding line migration at the ice-sheet scale~~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 ~~is a primary bottleneck~~ 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~~, as mentioned above~~. Coupling the GPU-based ice velocity and pressure simulations ~~executed on the GPUs with~~ with CPU-based ice thickness and temperature simulations ~~executed on the CPUs~~ can provide an enhanced balance between speed and predictive performance.

~~The objective of this study was~~ This study aimed to investigate the PT CUDA C implementation for unstructured meshes and ~~application to~~ its application to the 2-D SSA ~~approximation~~model formulation. For both Jakobshavn ~~Isbræ~~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 $3e^7$ degrees of freedom (DoFs) for the Pine Island glacier model; optimal solver parameters are unidentifiable. We ~~assessed and compared~~ further compare the performance of the PT CUDA C implementation with a standard ~~CPU implementation using two metrics: price and power consumptionto performance. The single Tesla V100 GPU is 1.5 times the price of~~ finite-element CPU-based implementation using the ~~two~~ price-to-performance metric. We justify the GPU implementation in the price-to-performance metric for up to a 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 ~~CPU processors. The~~ processor, the thermal design power is 140 W[2]. We executed the CPU-based multi-core MPI-parallelized ice-sheet flow simulations on two CPUs, 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 ~~CUDA C~~ GPU implementation was approximately one-seventh of the ~~standard~~ traditional CPU implementation for the test cases chosen in this study. We ~~expect a minimum speed-up of $>$1.5 to justify the Tesla V100 GPU price to performance. We report the performance (or the speed-up) across glacier configurations for DoFs tested to be $>$1.5 on a Tesla V100. We identified an exception at $2e^7$ DoFs for the Jakobshavn Isbræmodel, speed-up of $\sim$0.28. We attribute this drop in the speed-up to non-optimal global memory access patterns reported in the LITEX and L2 cache. We are investigating techniques to reduce the mesh non-localities and allow optimal global memory accesses~~will investigate this further.

[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

~~The study is~~ This study represents a first step toward leveraging GPU processing power~~for~~ , enabling more accurate polar ice discharge predictions. The insights gained ~~from this study~~ will benefit efforts to diminish spatial resolution constraints at ~~increased computing speed. The increased computing speed~~ higher computing performance. The higher computing performance will allow running ensembles of ice-sheet flow simulations at the continental scale and ~~at high resolution, previously not possible, enabling~~ higher resolution, a previously challenging task. The advances will further enable quantification of model sensitivity to changes in ~~future~~ upcoming climate forcings. These findings will significantly benefit process-oriented ~~and~~ sea-level-projection studies over the coming decades.

Code 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), as are input data and scripts to run the model and produce the plots for all the simulations presented in this paper. The PT CUDA C implementation runs on a CUDA-capable GPU device.

Author contributions. **AS** developed PT CUDA C implementation, conducted the performance assessment tests described in the manuscript followed by data analysis, manuscript edition. **LR** provided guidance on the early stages of the mathematical reformulation of the 2D SSA model to incorporate the PT method and supported the PT CUDA C implementation, manuscript edition. **MM** reformulated the 2D 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 the writing of the manuscript.

Competing interests. Ludovic Räss is on the Geoscientific Model Development editorial board.

Acknowledgements. 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 the hardware support. We thank the NVIDIA solution architects Oded Green, Zoe Ryan, and Jonathan Dursi for their thoughtful ~~inputs~~input. LR thanks Ivan Utkin for helpful discussions and acknowledges the Laboratory of Hydraulics, Hydrology, and Glaciology (VAW) at ETH Zurich for computing access on the Superzack GPU server. CPU and GPU hardware architectures were made available through the first author's access to the University of North Dakota's Talon and NVIDIA's Curiosity clusters.

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, Nature Climate 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, Journal of Computational Physics, 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, Geophysical Journal International, 216, 365–379, https://doi.org/10.1093/gji/ggy434, 2019.

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

Gilbarg, D. and Trudinger, N. S.: Elliptic partial differential equations of second order, vol. 224, Springer, 1977.

Glen, J. W.: The creep of polycrystalline ice, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 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, Current climate change reports, 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), Geoscientific Model Development, 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, Journal of Advances in Modeling Earth Systems, 13, e2021MS002 717, 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, Proceedings of the National Academy of Sciences, 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, Journal of Glaciology, 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, Proceedings of the National Academy of Sciences, 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), Journal of Geophysical Research: Earth Surface, 117, 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, Journal of Geophysical Research: Solid Earth, 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, Geophysical research letters, 44, 11–051, 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, Nature Geoscience, 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, pp. 175–195, https://link.springer.com/chapter/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, Geophysical Journal International, 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), Geoscientific Model Development, 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, Geoscientific Model Development, 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, Proceedings of the National Academy of Sciences, 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, Geophysical Research Letters, 38, https://doi.org/10.1029/2011GL047109, 2011.

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, Geoscientific Model Development, 8, 1197–1220, https://doi.org/10.5194/gmd-8-1197-2015, 2015.