Stockholms universitet

Stabilizing Glacier Simulations: Coupling Free-Surface Stabilization Algorithm to the Positivity Preserving Surface Constraint in FeniCS.

Stabilisering av Glaciärsimuleringar: Implementering av Free-Surface Stabilization Algorithm och Positivitetsbevarande Begränsning av Ytan i FEniCS.

Nasiha Häfener

Handledare: Igor Tominec, Josefin Ahlkrona
Examinator: Anders Mörtberg

Inlämningsdatum: 2024-05-20

Kandidatuppsats i datalogi
Bachelor Thesis in Computer Science

Abstract

This thesis investigates enhancing glacier simulations, focusing on the utilization of the Free-Surface Stabilization Algorithm (FSSA). We derived that the FSSA stabilization term provides a improved coupling between the Stokes problem and the free-surface equation. Furthermore, to create a realistic model we incorporate a positivity preserving constraint on the glacier's surface, ensuring that the surface remains positioned atop the bedrock throughout the simulation. This is implemented in FEniCS using Scalable Nonlinear Equations Solvers (SNES). The experiments are executed on Löfgren's glacier, a two-dimensional glacier with positive accumulation. Testing the simulation with and without FSSA shows that errors occur where SNES is active, which reveals that the positive preserving constraint needs to be included to the FSSA formulation. To mitigate this, the thesis proposes two remedies to decrease these errors, but their implementation and evaluation are left for future work.

Sammanfattning

Denna uppsats undersöker förbättring av glaciärsimulationer, med fokus på användningen av Free-Surface Stabilization Algorithm (FSSA). Vi härledde att FSSA stabiliserings termen ger en förbättrad koppling mellan Stokes problemet och free-surface ekvationen. Dessutom, för att skapa en realistisk modell, införlivar vi en positivitetbevarande begränsning av glaciärens yta, för att säkerställa dess position ovanpå berget under hela simuleringen. Detta implementeras i FEniCS med hjälp av Scalable Nonlinear Equations Solvers (SNES). Experimenten görs på Löfgrens glaciär, en tvådimensionell glaciär med positiv ackumulering. När simuleringen körs med och utan FSSA uppkommer det fel i de områden där SNES är aktiv, vilket visar att den positivitetsbevarande begränsningen bör inkluderas i formuleringen av FSSA. För att förhindra dessa fel föreslår vi i denna studie två lösningar på att minska felen, men deras implementering och utvärdering lämnas till framtida arbete.

Contents

1 Introduction

Glaciers and ice sheets play a crucial role in the global climate system by helping to regulate Earth's climate through their reflective properties. When solar radiation interacts with ice and snow, a substantial portion is reflected back into space, which is an important factor in regulating the planet's energy balance [1].
However, the ongoing rise in temperature has triggered accelerated ice loss in polar regions, which is expected to rapidly contribute to rising global sea levels [7].
Moreover, the stability of permafrost, frozen soil prevalent in polar regions [2], is increasingly threatened by temperature changes [7]. It is also known that permafrost holds a vast amounts of carbon dioxide and methane, and there is evidence suggesting that these greenhouse gases are beginning to transfer into the atmosphere. This may lead to further exacerbate global warming [7].

Understanding the dynamics of ice sheets is crucial for predicting their future behavior and analyzing their impact on climate change. Ice sheets can be described as a non-Newtonian fluid [12], characterized by nonlinear viscosity.
The law of conservation of mass, momentum, and energy, embodied in the Stokes equations, serves as a foundational principle to describe the evolution of ice sheets. When coupled with the free-surface equation which governs surface evolution [12], these equations form a basis for ice sheet modeling. Utilizing such a higher-order physics model has been shown to increase the reliability of predicting future ice loss [13].

Despite advancements in modeling techniques, challenges persist, particularly regarding computational efficiency. There are a number of numerical instabilities that can compromise its accuracy and reliability. A restrictive time step constraint is a significant obstacle inherent to ice sheet modeling [12].
However, recent developments, such as the Free-Surface Equation Algorithm (FSSA) [12, 11] offer promising solutions for overcoming these limitations. FSSA is based on Reynolds transport theorem and enables an implicit time-stepping scheme, allowing for larger time step sizes and thus reducing computational time by a significant amount [11].

Another instability arises when the upper surface, denoted as $h(x, y, t)$, of the ice sheet approaches the lower surface called bedrock, $b(x, y)$. Here, x and y are coordinates below the surface, and t is time. Simulations may yield, unphysical outcomes, with the ice sheet surface appearing below the bedrock. To resolve this, we introduce a positive preserving constraint, $h(x, y, t) > b(x, y)$, to our numerical scheme.
This constraint guarantees that the upper surface stays above the lower surface throughout the simulation. We will implement this using FEniCS, which is a Python finite

element method library for finding solutions of partial differential equations (PDEs) [4]. To enforce the positivity constraint, we use the Scalable Nonlinear Equations Solver (SNES), which employs the Lagrange multiplier method [3].

Our study involves implementing the positivity constraint within simulations conducted on a two dimensional synthetic glacier named Löfgren's glacier [11]. We investigate the impact of this constraint by comparing simulations utilizing FSSA against those without.

In Section 2, the Stokes problem is defined together with the free-surface equation. Subsequently, we discretize these by the means of FEM and explain how the FSSA is included in the Stokes problem. Furthermore, we explain the SNES algorithm and how it utilizes the Lagrange multiplier method to implement the positivity preserving constraint. Then we define what regions the error would appear when using the FSSA while the SNES algorithm is active.

In Section 3 we define the Löfgren's glacier setup, see Section 3.1, which is used in the simulations. In the same Section 3 we manually determine a stable time step which we then use in the rest of the experiments. In these experiments we investigate if there are an error occurring when running the simulation using FSSA and not using FSSA, while SNES is active. In Section 4 we discuss two possible solutions to decrease the error but leave the implementation and evaluation of these to future research and the discussion regarding the result of this thesis is in Section 5.

2 Preliminaries and Methods

In this section, we provide an overview of the fundamental mathematics and numerical methods used in this thesis, along with the equations related to formulating the ice sheet simulation problem.

2.1 Definitions

Definition 2.1.1 (Hilbert space [8]) *A Hilbert space denoted as H, is a complete inner product space. It is a vector space equipped with inner product that induces a norm, making it complete with respect to the metric induced by the norm.*

Definition 2.1.2 (L^2-space [8]) *The space $L^2(\Omega)$ is a Hilbert space consisting of square-integrable functions, defined on a domain Ω, denoted as f, such that $\int_\Omega |f(x)|^2 dx < \infty$.*

Definition 2.1.3 (Inner product [16]) *On a Hilbert space V, an inner product of two continuous functions $f(x)$ and $g(x)$ defined on a domain $\Omega \subset \mathbb{R}^d$ is:*

$$(f, g)_\Omega = \int_\Omega f(x)g(x)dx.$$

If we instead consider the functions on a boundary $\partial\Omega \subset \mathbb{R}^{d-1}$, an inner product is defined as:

$$(f, g)_{\partial\Omega} = \oint_{\partial\Omega} f(x)g(x)dS,$$

Where $\oint_{\partial\Omega}$ denotes the appropriate surface integral over the boundary $\partial\Omega$, and dS represents the surface measure.

Definition 2.1.4 (Frobenius inner product [15]) *Given two matrices A and B, where their elements are denoted by a_{ij} and b_{ij} we define the Frobenius inner product as:*

$$\int_\Omega \boldsymbol{A} : \boldsymbol{B} d\Omega = (\boldsymbol{A}, \boldsymbol{B})_\Omega = \sum_{i=1}^{N}\sum_{j=1}^{N}(a_{ij}, b_{ij}).$$

Definition 2.1.5 (Sobolev space [8]) *A Sobolev space is a function space that contains functions and their derivatives up to a certain order. The function space $H^k(\Omega)$ consists of functions defined on a domain Ω such that their derivatives up to order k are square integrable on Ω.*

Definition 2.1.6 (Normal [5]) *The unit normal vector to a surface described by the function $h(x, y, t)$ can be computed using the gradient of h and then normalized to have a length of 1:*

$$\boldsymbol{n} = \frac{\nabla h(x, y, t)}{||\nabla h(x, y, t)||}.$$

Here, $\nabla h(x, y, t)$ denotes the gradient vector of $h(x, y, t)$ and

$$||\nabla h(x, y, t)|| = \sqrt{(\nabla_x h)^2 + (\nabla_y h)^2}.$$

2.2 Theorems

Theorem 2.2.1 (Orthonormal basis [6]) *Let $B = [e_1, ...e_n]$ be an orthonormal basis for a vector space $\boldsymbol{V}$. Then the i-th coordinate of a vector $\boldsymbol{v} \in \mathcal{V}$ with respect to B is given by*

$$b_i = \boldsymbol{v}_i \cdot \boldsymbol{e}_i.$$

Corollary 2.2.1.1 *This is a corollary of Theorem 2.2.1:*

$$\boldsymbol{v} = (\boldsymbol{v} \cdot \boldsymbol{e}_1)\boldsymbol{e}_1 + ... + (\boldsymbol{v} \cdot \boldsymbol{e}_n)\boldsymbol{e}_n.$$

Theorem 2.2.2 (Divergence theorem [10]) *Let $\Omega \subset \mathbb{R}^d$ is a domain with boundary $\partial\Omega \subset \mathbb{R}^{d-1}$ then the divergence theorem states:*

$$\int_\Omega \frac{\partial f}{\partial x_i} dx = \int_{\partial\Omega} f \cdot n_i dS,$$

for $i = 1, 2$ where n_i a normal with i components and $f \in C(\Omega)$.

Theorem 2.2.3 (Green's Formula [10]) *Let u and v be functions defined on a domain $\Omega \subset \mathbb{R}^d$, and let $\partial\Omega \subset \mathbb{R}^{d-1}$ denote the boundary of Ω. Then Green's formula states:*

$$\int_\Omega \frac{\partial u}{\partial x_i} v dx = \int_\Omega \frac{\partial (uv)}{\partial x_i} dx - \int_\Omega \frac{\partial v}{\partial x_i} u dx, \tag{2.1}$$

where $u, v \in C^0$.

Corollary 2.2.3.1 *By applying Theorem 2.2.2 to the first term on the left-hand side of (2.1), we obtain:*

$$\int_\Omega \frac{\partial u}{\partial x_i} v dx = \int_{\partial\Omega} uv \cdot n_i dS - \int_\Omega u \frac{\partial v}{\partial x_i} dx.$$

If $u = u_i$ represents the components of a vector field on Ω, then Green's formula can be expressed as:

$$\int_\Omega (\nabla u) v dx = \int_{\partial\Omega} (u \cdot n) v dS - \int_\Omega u \cdot (\nabla v) dx.$$

For $u = -\nabla u$ we obtain:

$$\int_\Omega -\Delta u v dx = -\int_{\partial\Omega} n \cdot \nabla u v dS + \int_\Omega \nabla u \cdot \nabla v dx.$$

Theorem 2.2.4 (Reynolds Transport Theorem [9]) *On a time-dependent domain $\Omega(t) \subset \mathbb{R}^d$ where $t \in \mathbb{R}^+$, we can define Reynolds transport theorem as follows:*

$$\frac{d}{dt}\int_{\Omega(t)} \phi d\Omega = \int_{\Omega(t)} \frac{d}{dt}\phi d\Omega + \int_{\partial\Omega(t)} (\boldsymbol{u} \cdot \boldsymbol{n})\phi d\Gamma.$$

Here, ϕ is a scalar-valued function and $\boldsymbol{u}$ represents the velocity at the boundary $\Gamma \subset \mathbb{R}^{d-1}$.

2.3 Numerical methods

In this section we introduce numerical methods used for computing ice sheet simulations.

2.3.1 Forward Euler

Forward Euler is a numerical technique that uses an explicit approach to approximate a solution to a differential equation [17]. Here, we consider an ordinary differential equation (ODE) that depends on only one independent variable. Consider an ODE:

$$\frac{dy}{dt} = f(t, y),$$

where $t \in \mathbb{R}^+$ and with some initial value $y(t_0) = y_0$. This method uses the slope of the function $f(t, y)$ to predict the value of y at the next time step $t_{k+1} = t_k + \Delta t$. We approximate the time-derivative as follows:

$$\frac{y_{k+1} - y_k}{\Delta t} = f(t_k, y_k) \Leftrightarrow$$

$$y_{k+1} = y_k + \Delta t f(t, y_k).$$

It's evident that for an explicit method, we calculate the function in a future time step using the current function value.

2.3.2 Backward Euler

Backward Euler is a numerical technique used to approximate a differential equation, employing an implicit approach [17]. Consider an ODE:

$$\frac{dy}{dt} = f(t, y),$$

with some initial value at t_0 given by $y(t_0) = y_0$. We can evaluate the time derivative as follows:

$$\frac{y_{k+1} - y_k}{\Delta t} = f(t_{k+1}, y_{k+1}) \Leftrightarrow$$

$$y_{k+1} = y_k \Delta t f(t_{k+1}, y_{k+1}).$$

For an implicit method, we calculate the next time step function value using both the current and next time steps.

When $f(t_{k+1}, y_{k+1})$ is a nonlinear function of y_{k+1}, we use an iterative technique such as Newton's or Picard's method. An implicit approach is often more stable with respect to the time step size.

2.3.3 Finite element method (FEM)

We aim to explain FEM using an easier example, focusing on Poisson's equation in two dimensions [10]:

$$-\Delta u = f, \quad on\ \Omega, \tag{2.2}$$

$$u = 0, \quad on\ \partial\Omega. \tag{2.3}$$

Where $\Omega \subset \mathbb{R}^d$ is an open and bounded domain. Here, $\Delta = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})$ denotes the Laplace operator in two dimensions and f is a given function in $L^2(\Omega)$, defined as in Definition 2.1.2. We define a function space V_0:

$$V_0 = \{v : ||\nabla v||_{L^2} + ||v||_{L^2} < \infty, v\big|_{\partial\Omega} = 0\}.$$

This means that both the L^2-norm of the function v and its gradient are finite, and the function is zero on the boundary.
To solve the PDEs (2.2)-(2.3), we first obtain the variational formulation by multiplying (2.2) with a test function v:

$$-\Delta u v = f v.$$

Integrating both sides over the domain Ω, we get:

$$(-\Delta u, v)_\Omega = (f, v)_\Omega.$$

Using Green's formula, see Theorem 2.2.3, and noting that $v = 0$ on the boundary $\partial\Omega$ we obtain:

$$(-\Delta u, v)_\Omega = (\nabla u, \nabla v)_\Omega - (n \cdot \nabla u, v)_{\partial\Omega} = (\nabla u, \nabla v)_\Omega.$$

Thus the variational formulation of (2.2) is to find $u \in V_0$ such that:

$$(\nabla u, \nabla v)_\Omega = (f, v)_\Omega, \quad \forall v \in V_0.$$

We then subdivide domain Ω into triangular elements (mesh) denoted by Ω_h. Let V_h be a space of continuous piecewise linear functions on Ω_h. We have $V_{h,0} \subset V_h$ defined as:

$$V_{h,0} = \{v \in V_h : v\big|_{\partial\Omega} = 0\}.$$

With this approximation, we have the finite element method for (2.2), denoted by $u_h \in V_{h,0}$ such that:

$$(\nabla u_h, \nabla v)_\Omega = (f, v)_\Omega, \quad u_h \in V_{h,0}. \tag{2.4}$$

To compute the finite element approximation u_h, we compute the hat functions φ_i and sum these over the domain. Let $\{\varphi_i\}_{i=1}^N$ be a basis for $V_{h,0}$. Let $v = \varphi_i$, then (2.4) can then be written as:

$$(\nabla u_h, \nabla \varphi_i)_\Omega = (f, \varphi_i)_\Omega, \quad i = 1, 2, .., N. \tag{2.5}$$

Since u_h is in V_h we can express it as:

$$u_h = \Sigma_{j=1}^N \xi_j \varphi_j,$$

with N unknown $\xi_j \in \mathbb{R}$. Adding this to (2.5), we get:

$$\Sigma_{j=1}^N \xi_j (\nabla \varphi_j, \nabla \varphi_i)_\Omega = (f, \varphi_i)_\Omega, \quad i = 1, 2, .., N.$$

This leads to the definition of:

$$\begin{aligned} A_{i,j} &= (\nabla \varphi_j, \nabla \varphi_i)_\Omega, \qquad i = 1, 2, .., N, \quad j = 1, 2, .., N, \\ b_i &= (f, \varphi_i)_\Omega, \qquad\qquad\qquad\quad i = 1, 2, .., N. \end{aligned}$$

We can express b_i as:

$$b_i = \Sigma_{j=1}^N A_{i,j} \xi_j, \quad i = 1, 2, .., N.$$

Giving us a linear system for an unknown vector ξ:

$$b = A\xi.$$

where A is the stiffness matrix and b is the load vector. Solving this linear system provides us with ξ and therefore u_h as well.

2.3.4 Newton's method

Newton's method is a numerical technique to find better approximations to the roots of a real-valued function $f(x)$ [8]. Initially we select a guess x_0 near the actual root of f. The closer this initial guess is to the actual root, the faster it will converge. The method employs an iterative formula, beginning with $x_n = x_0$. At each iteration n we compute a new approximation x_{n+1} using the formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, ..$$

This process continues iteratively, updating the value of x until the difference between successive approximations $|x_{n+1} - x_n|$ falls below a predetermined tolerance level. If convergence occurs, the sequence of approximations $x_0, .., x_n$ will converge to the actual root of the functions. The final value of x after iteration provides the approximation to the root of f. The Newton's method is used for solving the constraint nonlinear minimization problem formulated by the means of the Lagrange multiplier method In Secton 2.10. This is then used on (2.20).

2.3.5 Picard's method

The Picard's method is a numerical technique to approximate solutions to certain types of functional equations [8]. We start with a given functional equation of the form $y = g(x, y)$, where y is the unknown function and $g(x, y)$ is a given function of both x and y. Starting with an initial guess y_0 for the unknown function $y(x)$, the iterative process defines a sequence of functions $y_1(x), y_2(x), ...$ recursively:

$$y_{n+1}(x) = g(x, y_n(x)), \quad n = 0, 1, 2, ..$$

This sequence $y_0(x), y_1(x), ..$ converges to a fixed point $y(x)$ such that:

$$y(x) = g(x, y_n(x)).$$

The resulting function $y(x)$ serves as an approximation to the solution of the original functional equation. Picard's method is used when solving the Stokes problem, Equation (2.29) when simulating Löfgren's glacier, in Section 3.1.

In practice Newton's method generally converges more rapidly than Picard's method when the initial guess is closer to the root. However, Newton's method involves derivative computations, which may not always be feasible and can be computationally expensive. Therefore the selection between the two methods depends on the function's characteristics. For functions where computing derivatives is impractical or costly, Picard's method may be preferred despite its potentially slower convergence rate.

2.3.6 The Lagrange multiplier method

The Lagrange multiplier method is an analytical technique utilized to identify local maxima or minima of a function while adhering to an equality constraint [8]. Initially we define an objective function $f(x_1, .., x_n)$, representing the quantity we seek to

maximize or minimize. Subsequently, we introduce an equality constraint of the form $g(x_1, .., x_n) = 0$, where g is a function of the same variables as f. We construct the Lagrangian function $\mathcal{L}(x_1, .., x_n, \lambda)$ as follows:

$$\mathcal{L}(x_1, .., x_n, \lambda) = f(x_1, .., x_n) + \lambda g(x_1, .., x_n).$$

In the equation above we integrated the constraint into the objective function by multiplying with the Lagrange multiplier λ. Then we compute the partial derivatives for each variable and identify the stationary conditions. Setting the partial derivatives to zero we detect the critical points:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_i} &= 0, \quad i = 1, 2, .., n, \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= 0. \end{aligned}$$

We solve the system of equations generated by the stationarity conditions to find optimized values of $x_1, .., x_n$. The Lagrange multiplier method is utilized by SNES when we implement the constraint on the surface height explained in Section 2.10.

2.4 The Stokes equations

Ice sheets can be characterized as a slowly moving, gravity driven highly viscous fluid [11]. When considering the movement of fluid, two types of stresses play significant roles: internal stresses arising from changes in pressure as the fluid moves and viscous stresses, which result from the fluid's resistance to deformation. This leads to the Cauchy stress tensor, which describes the distribution of forces within a fluid:

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \mu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T). \tag{2.6}$$

Here, p represents pressure and $\boldsymbol{I}$ is the identity matrix. The coefficient μ is the viscosity of the fluid which depends on the strain-rate tensor [10]:

$$\boldsymbol{D}\boldsymbol{u} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T). \tag{2.7}$$

This describes how the fluid deforms at a specific point. The conservation of mass is expressed by the equation:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.8}$$

which means that the fluid is incompressible. We also have the conservation of momentum for a fluid, expressed as:

$$\rho \dot{\boldsymbol{u}} + \rho(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{f}. \tag{2.9}$$

Here, ρ represents the density of the fluid, and $\boldsymbol{f}$ is a given function [10]. If we compute the divergence of the stress tensor from (2.6). We further use (2.8) in the equation above to arrive to:

$$\nabla \cdot \boldsymbol{\sigma} = -\nabla p + \mu(\Delta \boldsymbol{u} + \nabla(\nabla \cdot \boldsymbol{u}) = -\nabla p + \mu \Delta \boldsymbol{u}.$$

Incorporating this into the momentum equation, (2.9) yields:

$$\begin{aligned} \rho \dot{\boldsymbol{u}} + \rho(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} &= -\nabla p + \mu \Delta \boldsymbol{u} + f \Leftrightarrow \\ \dot{\boldsymbol{u}} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} &= -\frac{\nabla p}{\rho} + \frac{\mu}{\rho}\Delta \boldsymbol{u} + f. \end{aligned}$$

This results in a set of nonlinear PDEs for velocity and pressure, known as the Navier-Stokes equations:

$$\begin{aligned} \dot{\boldsymbol{u}} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \frac{\nabla p}{\rho} - \frac{\mu}{\rho}\Delta \boldsymbol{u} &= \boldsymbol{f}, \quad \textbf{on } \Omega, \\ \nabla \cdot \boldsymbol{u} &= \boldsymbol{0}, \quad \textbf{on } \Omega. \end{aligned}$$

From the Navier-Stokes equations, we can derive the Stokes equations. The Stokes equations are advantageous for analysis as they are linear, when the viscosity function is linear, yet they still provide a reliable model for fluid flow [10]. Given the slow and predominantly parallel streamline flow of ice, we can neglect the term $(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}$ and since the flow over ice sheets is slow, we neglect the acceleration term $\dot{\boldsymbol{u}} = 0$. Thus we arrive at:

$$\begin{aligned} -\mu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f}, \quad \text{on } \Omega, \\ \nabla \cdot \boldsymbol{u} &= \boldsymbol{0}, \quad \text{on } \Omega. \end{aligned} \tag{2.10}$$

In this derivation of the Stokes equations, we assume linearity of the viscosity μ [10]. Instead, we aim to describe the Stokes equations specifically for ice, acknowledging the nonlinear nature of ice viscosity, which can be characterized by Glen's flow law [14]:

$$\mu(\boldsymbol{Du}) = \frac{1}{2}A^{-\frac{1}{n}}(\epsilon_{crit}^2 + \frac{1}{2}||\boldsymbol{Du}||_F^2)^{\frac{\frac{1}{n}-1}{2}}.$$

Here, $||()||_F$ is the Frobenius norm. The parameter n represents the flow law exponent, signifying the sensitivity of ice deformation to stress. Based on earlier research a reasonable choice for n is 3. Additionally, to ensure that the viscosity does not tend towards infinity as the strain rate approaches zero, a small regulation constant ϵ_{crit} is introduced [12]. Moreover, ice viscosity is influenced by temperature through the rate factor $A(T')$, which depends on the ice relative to the pressure melting point T' according to the Arrhenius equation:

$$A(T') = A_0 e^{-\frac{Q}{RT'}},$$

where A_0 is a pre-exponential factor, Q is the activation energy and R is the ideal gas constant, see Table 2.1. This values are considered reasonable for ice temperatures $T' \leq 263.15K$, when $n = 3$ [14]. With this we define the Cauchy stress tensor for ice:

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu(\boldsymbol{Du})\boldsymbol{Du}, \tag{2.11}$$

which incorporates the deviatoric stress tensor:

$$S(\boldsymbol{Du}) = 2\mu(\boldsymbol{Du})\boldsymbol{Du}, \tag{2.12}$$

dependent on the strain-rate tensor in (2.7). Consequently, with the nonlinear viscosity of ice, (2.10) becomes:

$$-\nabla \cdot S(\boldsymbol{Du}) + \nabla p = \boldsymbol{f}.$$

The force acting on the ice body is primarily due to the gravity g, exerted in a downward direction. The right-hand side of the equation should also incorporate the density ρ, as seen in the derivation of the Navier-Stokes equations. These values are available in Table 2.1. We define the right-hand side as $\boldsymbol{f} = (0, -\rho g)$ in $\mathbb{R}^2$ or $\boldsymbol{f} = (0, 0, -\rho g)$ in $\mathbb{R}^3$. Thus, we can define the Stokes equations for ice flow as [12]:

$$-\nabla \cdot 2\mu(\boldsymbol{Du})\boldsymbol{Du} + \nabla p = \boldsymbol{f}, \quad \text{on } \Omega, \tag{2.13}$$

$$\nabla \cdot \boldsymbol{u} = 0, \quad \text{on } \Omega. \tag{2.14}$$

Table 2.1: Parameters for computing the Stokes equations for ice:

A_0	$3.985 \times 10^{-13} s^{-1} Pa^{-3}$
Q	$60\ kJ\ mol^{-13}$
R	$8.314462\ JK^{-1}\ mol^{-1}$
A	$100 MPa^{-3} yr^{-1} \approx 3.2 Pa^{-3} s^{-1}$
ϵ^2_{crit}	$1.0 \times 10^{-11}\ m^2 s^{-2}$
g	$9.8\ m/s^2$
ρ	$910\ kg/m^3$

2.5 Glacier boundary conditions

The boundary of the glacier is denoted as $\partial\Omega$ comprising four distinct boundaries: $\Gamma_S, \Gamma_B, \Gamma_E$ and Γ_W, see Figure 2.1. The boundary conditions for the ice sheet are as follows:

$$\sigma \cdot \boldsymbol{n} = 0, \qquad \text{on } \Gamma_s, \tag{2.15}$$

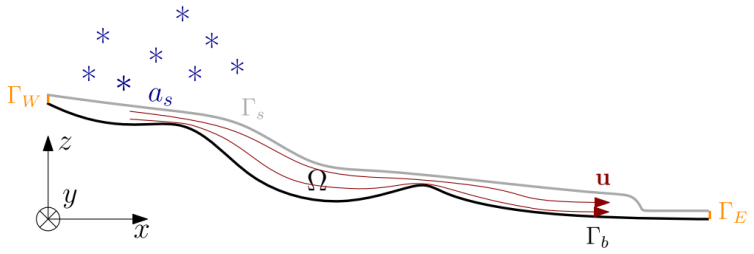
$$\boldsymbol{u} = 0, \qquad \text{on } \Gamma_e, \Gamma_w, \Gamma_b^f, \tag{2.16}$$

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0, \qquad \text{on } \Gamma_b^s, \tag{2.17}$$

$$\boldsymbol{t}_i \cdot (\sigma \cdot \boldsymbol{n}) = -\beta^2 \boldsymbol{u} \cdot \boldsymbol{t}_i, \qquad \text{on } \Gamma_b^s. \tag{2.18}$$

In (2.15), the stress tensor is defined as in (2.11). Here, $\boldsymbol{n}$ represent a unit normal pointing outward from the boundary. Equation (2.15) denotes a stress-free condition on the surface [11]. In (2.16), velocity is specified as zero on the east and west boundaries. At the bedrock, two different conditions are considered: either Γ_b^f, when the ice is frozen to the bedrock, or Γ_b^s, when the ice is sliding on top of the bedrock. Equation (2.16) indicates that when the ice is frozen the velocity is zero. For the sliding condition on top of the bedrock, the velocity is divided depending on the direction. Equation (2.17) specifies that the velocity is zero in the direction of the normal $\boldsymbol{n}$. Equation (2.18) introduces $\boldsymbol{t}_i$ representing the tangent spanning the plane where $i = [0, d-1]$, with d as the dimension and $d \in \{2, 3\}$. Equation (2.18) is derived from Weertman sliding law [18], where β is the drag coefficients of the ice. It establishes the relationship between the velocity in direction of the tangent and the stress tensor when the ice is sliding along the bedrock [11].

Figure 2.1: This image depicts a glacier, represented in a two-dimensional domain denoted as Ω. The boundaries of this domain are defined as follows: Γ_S represents the surface boundary, Γ_B indicates the bedrock boundary, while Γ_E and Γ_W denote the east and west boundaries respectively. The flow velocity of the glacier is described by the vector $\boldsymbol{u}$, illustrating the motion of the ice within the domain. Notably, at the surface, there is an accumulation rate denoted as a_s, representing the additional mass accumulation from falling snow [11].



2.6 Free-surface equation

To describe the evolution of the glacier surface, we employ the free-surface equation. In three dimensions ($\mathbb{R}^3$) the surface of the ice can be expressed as:

$$f(x, y, z, t) = h(x, y, t) - z,$$

where $h(x, y, t)$ denotes the surface elevation at position (x, y) and time t, and z represents the vertical coordinate. The governing transport equation for the alteration

in surface position is formulated as [14]:

$$\frac{\partial f}{\partial t} = -\boldsymbol{u} \cdot \nabla f. \tag{2.19}$$

Evaluating the right-hand side of (2.19) yields:

$$\frac{\partial f}{\partial t} = \frac{\partial (h(x,y,t) - z)}{\partial t} = \frac{\partial h}{\partial t},$$

which represents the rate of change of the surface elevation over time. On the right-hand side of (2.19), we have:

$$-\boldsymbol{u} \cdot \nabla f = -\boldsymbol{u} \cdot \nabla (h(x,y,t) - z) = -\boldsymbol{u} \cdot (\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}, -1).$$

Defining the horizontal velocity components as $u_\perp = (u_x, u_y)$ and the horizontal gradient of the surface elevation as $\nabla_\perp h = (\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y})$, we rewrite (2.19) as:

$$\frac{\partial h}{\partial t} = -\boldsymbol{u}_\perp \cdot \nabla_\perp h + u_z.$$

Incorporating the accumulation term a_s representing the vertical rate of mass accumulation on the surface, the free-surface equation becomes:

$$\frac{\partial h}{\partial t} = -\boldsymbol{u}_\perp \cdot \nabla_\perp h + u_z + a_s. \tag{2.20}$$

Equation (2.20) describes the velocity at the glaciers surface Γ_s, accounting for both horizontal and vertical motion components, as well as the accumulation of mass on the surface [12].

2.7 Discretization of free-surface equation

The free-surface equation characterizes the velocity at the glacier's surface and relies on the Stokes equations for ice, (2.13)-(2.14) [12]. Initially,we solve the Stokes equation to determine the velocity $\boldsymbol{u}$ at the surface. Subsequently, we utilize this velocity in the free-surface equation to compute a new height $h(x, y, t + \Delta t)$, thus defining the updated domain $\Omega(t + \Delta t)$. This procedure continues to update the domain in this way until the final time is reached.

While (2.20) can be discretized using the explicit Euler method, see Section 2.3.1, which offers simplicity, it introduces limitations on the time step size, thereby reducing its stability. Alternatively, employing an implicit time-stepping scheme involves solving $\boldsymbol{u}(t + \Delta t)$ using the unknown data at the subsequent time step. However, this necessitates solving the Stokes equations repeatedly within each time step, iterating between the domains. This iterative process adds an extra computational loop,

resulting in increased computational time. Consequently, despite offering stability advantages over the explicit method, the implicit scheme becomes inefficient due to the additional computational burden incurred.

Instead of employing an explicit or fully-implicit scheme, we opt for a semi-implicit approach to discretize the free-surface equation (2.20) [11]. This method combines Backward Euler, see Section 2.3.2, for the surface $h(x, y, t)$ and Forward Euler, see Section 2.3.1, for the velocity $\boldsymbol{u}$:

$$\frac{h^{k+1} - h^k}{\Delta t} + u_x^k \frac{\partial h^{k+1}}{\partial x} + u_y^k \frac{\partial h^{k+1}}{\partial y} = u_z^k + a_s^k.$$

We further rearrange the terms in the equation to obtain:

$$h^{k+1} + \Delta t(u_x^k \frac{\partial h^{k+1}}{\partial x} + u_y^k \frac{\partial h^{k+1}}{\partial y}) = h^k + \Delta t(u_z^k + a_s^k). \tag{2.21}$$

Then we employ FEM, similar to the approach outlined in Section 2.3.3, we introduce a test function $v \in V^s$ where:

$$V^s = \{H^1(\Omega^\perp) | v : ||v||_{L^2(\Omega^\perp)} + ||\nabla v||_{L^2(\Omega^\perp)} < \infty\},$$

where $\Omega^\perp = \{(x, y, 0) \in \Omega\}$. Multiplying (2.21) with the test function v and integrating over $\Omega^\perp$, we can formulate the weak formulation aiming to find $h^{k+1} \in V^s$ such that,

$$(h^{k+1}, v)_{\Omega^\perp} + \Delta t(u_x^k \frac{\partial h^{k+1}}{\partial x} + u_y^k \frac{\partial h^{k+1}}{\partial y}, v)_{\Omega^\perp} = (h^k, v)_{\Omega^\perp} + \Delta t(u_z^k + a_s^k, v)_{\Omega^\perp}, \quad \forall v \in V^s. \tag{2.22}$$

Following the approach demonstrated in Section 2.3.3, we constrain the test and trial space, denoted as V^s, to the subspace V_h^s. This subspace consists of piecewise-linear polynomials.

2.8 Finite element method for the Stokes equations

We begin by discretizing the Stokes equation using FEM, explained in Section 2.3.3. To facilitate this, we introduce the function spaces $\boldsymbol{V}$ and Q for velocity $\boldsymbol{u}$ and pressure p as follows:

$$\boldsymbol{V} = \{\boldsymbol{v} \in \boldsymbol{H}^1(\Omega) | \boldsymbol{v}|_{\partial\Omega \setminus \Gamma_s \cup \Gamma_b^s} = 0, \boldsymbol{v} \cdot \boldsymbol{n}|_{\Gamma_b^s} = 0\},$$
$$Q = \{q \in L^2(\Omega)\}.$$

Incorporating the boundary conditions in (2.15) where we specify that the velocity is zero except on the surface and when the ice is sliding on the bedrock. Also when the ice is sliding we defined Γ_{b^s} where we have that the velocity is zero in the direction of

the normal $\boldsymbol{n}$. We proceed to multiply our Stokes equations, (2.13)-(2.14) by the test functions $\boldsymbol{v} \in \boldsymbol{V}$ and $q \in Q$:

$$-\nabla \cdot (2\mu(\boldsymbol{Du})\boldsymbol{Du})\boldsymbol{v} + \nabla p\boldsymbol{v} = \boldsymbol{f}\boldsymbol{v}, \tag{2.23}$$

$$(\nabla \cdot \boldsymbol{u})q = 0. \tag{2.24}$$

Integrating (2.23) over the whole domain Ω:

$$-(\nabla \cdot 2\mu(\boldsymbol{Du})\boldsymbol{Du}, \boldsymbol{v})_\Omega + (\nabla p, \boldsymbol{v})_\Omega = (\boldsymbol{f}, \boldsymbol{v})_\Omega, \quad \forall \boldsymbol{v} \in \boldsymbol{V}.$$

We then apply Green's Formula, see Theorem 2.2.3 and derive:

$$(2\mu(\boldsymbol{Du})\boldsymbol{Du}, \nabla \boldsymbol{v})_\Omega - (\boldsymbol{n} \cdot 2\mu(\boldsymbol{Du})\boldsymbol{Du}, \boldsymbol{v})_{\partial\Omega} - (p, \nabla \cdot \boldsymbol{v})_\Omega + (\boldsymbol{n} \cdot \boldsymbol{v}, p)_{\partial\Omega} = (\boldsymbol{f}, \boldsymbol{v})_\Omega. \tag{2.25}$$

We previously defined the stress tensor in (2.11), and by integrating it over the boundary, we effectively derive the stress contribution. So that (2.25) instead can be written as:

$$(2\mu(\boldsymbol{Du})\boldsymbol{Du}, \nabla \boldsymbol{v})_\Omega - (p, \nabla \cdot \boldsymbol{v})_\Omega - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\partial\Omega} = (\boldsymbol{f}, \boldsymbol{v})_\Omega. \tag{2.26}$$

We then focus on the integral over the boundary $\partial\Omega$, separating it into different boundaries:

$$\begin{aligned} -(\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\partial\Omega} &= -(\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_s} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_e} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_w} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b} \\ &= -(0, \boldsymbol{v})_{\Gamma_s} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, 0)_{\Gamma_e} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, 0)_{\Gamma_w} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b} \\ &= -(\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b}. \end{aligned}$$

On the surface boundary, $\boldsymbol{\sigma} \cdot \boldsymbol{n} = 0$ defined in (2.15). Also we defined that $\boldsymbol{v} = 0$ for the east and west boundaries earlier. For the sliding bedrock we further refine the calculation when the ice is frozen and when the ice is sliding:

$$\begin{aligned} -(\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b} &= -(\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b^f} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b^s} \\ &= -(\boldsymbol{\sigma} \cdot \boldsymbol{n}, 0)_{\Gamma_b^f} - (\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b^s} \\ &= -(\boldsymbol{\sigma} \cdot \boldsymbol{n}, \boldsymbol{v})_{\Gamma_b^s}. \end{aligned} \tag{2.27}$$

For the frozen bedrock Γ_b^f, we also defined $\boldsymbol{v} = 0$. So from the integral over the boundaries we only have to consider the bedrock when the ice is sliding. Using Theorem 2.2.1 of orthonormal basis, we can write the stress tensor in three dimensions:

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = ((\boldsymbol{\sigma} \cdot \boldsymbol{n}) \cdot \boldsymbol{n})\boldsymbol{n} + ((\boldsymbol{\sigma} \cdot \boldsymbol{n}) \cdot \boldsymbol{t_1})\boldsymbol{t_1} + ((\boldsymbol{\sigma} \cdot \boldsymbol{n}) \cdot \boldsymbol{t_2})\boldsymbol{t_2}.$$

If we use this in (2.27), we use the fact that we defined $\boldsymbol{v}\cdot\boldsymbol{n}=0$ and we also use (2.18) so that:

$$\begin{aligned}
-(\boldsymbol{\sigma}\cdot\boldsymbol{n},\boldsymbol{v})_{\Gamma_b^s} &= -((((\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\boldsymbol{n})\boldsymbol{n}+((\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\boldsymbol{t}_1)\boldsymbol{t}_1+((\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\boldsymbol{t}_2)\boldsymbol{t}_2),\boldsymbol{v})_{\Gamma_b^s}\\
&= -((\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\boldsymbol{n},\boldsymbol{n}\cdot\boldsymbol{v})_{\Gamma_b^s}-((\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\boldsymbol{t}_1,\boldsymbol{t}_1\cdot\boldsymbol{v})_{\Gamma_b^s}-((\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\boldsymbol{t}_2,\boldsymbol{t}_2\cdot\boldsymbol{v})_{\Gamma_b^s}\\
&= -((\boldsymbol{\sigma}\cdot\boldsymbol{n})\cdot\boldsymbol{n},0)_{\Gamma_b^s}-(-\beta^2(\boldsymbol{t}_1\cdot\boldsymbol{u}),\boldsymbol{t}_1\cdot\boldsymbol{v})_{\Gamma_b^s}-(-\beta^2(\boldsymbol{t}_2\cdot\boldsymbol{u}),\boldsymbol{t}_2\cdot\boldsymbol{v})_{\Gamma_b^s}\\
&= (\beta^2(\boldsymbol{t}_1\cdot\boldsymbol{u}),\boldsymbol{t}_1\cdot\boldsymbol{v})_{\Gamma_b^s}+(\beta^2(\boldsymbol{t}_2\cdot\boldsymbol{u}),\boldsymbol{t}_2\cdot\boldsymbol{v})_{\Gamma_b^s}\\
&= (\beta^2((\boldsymbol{t}_1\cdot\boldsymbol{u})\boldsymbol{t}_1+(\boldsymbol{t}_2\cdot\boldsymbol{u})\boldsymbol{t}_2),\boldsymbol{v})_{\Gamma_b^s}.
\end{aligned} \tag{2.28}$$

We use Theorem 2.2.1 to write an orthonormal basis for $\boldsymbol{u}$:

$$\boldsymbol{u}=(\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{n}+((\boldsymbol{t}_1\cdot\boldsymbol{u})\boldsymbol{t}_1+(\boldsymbol{t}_2\cdot\boldsymbol{u})\boldsymbol{t}_2),$$

where we defined $\boldsymbol{u}\cdot\boldsymbol{n}=0$ on Γ_b^s in (2.17) so that:

$$\boldsymbol{u}=((\boldsymbol{t}_1\cdot\boldsymbol{u})\boldsymbol{t}_1+(\boldsymbol{t}_2\cdot\boldsymbol{u})\boldsymbol{t}_2).$$

From this we have that (2.28) instead can be written as:

$$(\beta^2((\boldsymbol{t}_1\cdot\boldsymbol{u})\boldsymbol{t}_1+(\boldsymbol{t}_2\cdot\boldsymbol{u})\boldsymbol{t}_2),\boldsymbol{v})_{\Gamma_b^s}=(\beta^2\boldsymbol{u},\boldsymbol{v})_{\Gamma_b^s}.$$

We can now conclude that the integral over the boundary is equal to the equation above. We rewrite (2.26) as:

$$(2\mu(\boldsymbol{Du})\boldsymbol{Du},\nabla\boldsymbol{v})_\Omega-(p,\nabla\cdot\boldsymbol{v})_\Omega+(\beta^2\boldsymbol{u},\boldsymbol{v})_{\Gamma_b}=(\boldsymbol{f},\boldsymbol{v})_\Omega,\quad\forall\boldsymbol{v}\in\boldsymbol{V}.$$

Then we also integrate (2.24) over the whole domain Ω:

$$(\nabla\cdot\boldsymbol{u},q)_\Omega=0,\quad\forall q\in Q.$$

We can now add the two equations above together to make it simpler and we subtract the second equation because we want to have a symmetric matrix when we derive the linear system [12]. The weak formulation is then, to find $\boldsymbol{u}\in\boldsymbol{V}$ and $p\in Q$ such that:

$$\begin{aligned}
(2\mu(\boldsymbol{Du})\boldsymbol{Du},\nabla\boldsymbol{v})_\Omega-(p,\nabla\cdot\boldsymbol{v})_\Omega-(\nabla\cdot\boldsymbol{u},q)_\Omega+(\beta^2\boldsymbol{u},\boldsymbol{v})_{\Gamma_b}&=(\boldsymbol{f},\boldsymbol{v})_\Omega,\\
\forall\boldsymbol{v}\in\boldsymbol{V}&\text{and }\forall q\in Q.
\end{aligned}$$

We now subdivide Ω into triangles and let this be a mesh $\Omega_h=\cup_{i=1}^N\mathcal{K}_i$, where $\mathcal{K}_i$ is a mesh element. We choose $\boldsymbol{V}_h$ to be a space of piecewise qudratic polynomials over Ω_h, and Q_h a space of piecewise linear polynomials over Ω_h. The final Stokes problem formulation is: find $\boldsymbol{u_h}\in\boldsymbol{V}_h$ and $p_h\in Q_h$ such that:

$$\begin{aligned}
(2\mu(\boldsymbol{Du}_h)\boldsymbol{Du}_h,\nabla\boldsymbol{v})_\Omega-(p_h,\nabla\cdot\boldsymbol{v})_\Omega-(\nabla\cdot\boldsymbol{u}_h,q)_\Omega+(\beta^2\boldsymbol{u}_h,\boldsymbol{v})_{\Gamma_b}&=(\boldsymbol{f},\boldsymbol{v})_\Omega,\\
\forall\boldsymbol{v}\in\boldsymbol{V}_h&\text{ and }\forall q\in Q_h.
\end{aligned} \tag{2.29}$$

Then we let $\{\boldsymbol{\varphi}_i\}_{i=1}^n$ be a set of vector valued basis functions for $\boldsymbol{V}_h$ and $\{\chi_i\}_{i=1}^m$ be a scalar basis functions for Q_h. We also have $\boldsymbol{\xi}$ and ω containing the vectors of unknown degree so that:

$$\boldsymbol{u}_h = \sum_{j=1}^{n} \boldsymbol{\xi}_j \boldsymbol{\varphi}_j$$
$$p_h = \sum_{j=1}^{m} \omega_j \chi_j.$$

We can now write the (2.29) so that:

$$\sum_{j=1}^{n} \boldsymbol{\xi}_j (2\mu(\nabla\boldsymbol{\varphi}_j)\nabla\boldsymbol{\varphi}_j, \nabla\boldsymbol{\varphi}_i)_\Omega - \sum_{j=1}^{m} \omega_j (\chi_j, \nabla\cdot\boldsymbol{\varphi}_i)_\Omega - \sum_{j=1}^{n} \boldsymbol{\xi}_j (\nabla\cdot\boldsymbol{\varphi}_j, \chi_i) +$$
$$\sum_{j=1}^{n} \boldsymbol{\xi}_j (\beta^2\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i)_{\Gamma_b} = -(f, \boldsymbol{\varphi}_i)_\Omega, \quad \forall\boldsymbol{\varphi}_i \in \boldsymbol{V}_h \text{ and } \forall\chi_i \in Q_h.$$

From this we get a matrix system simarly as in the Poisson example in Section 2.3.3. The different is that we have a stiffness matrix A and a divergence matrix B. We form a blockstructured system [10]:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \xi \\ \omega \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

where we have that,

$$\begin{aligned} A_{ij} &= (2\mu(\nabla\boldsymbol{\varphi}_j)\nabla\boldsymbol{\varphi}_j, \nabla\boldsymbol{\varphi}_i)_\Omega + (\beta^2\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j)_{\Gamma_b}, \\ B_{ij} &= (\chi_j, \nabla\cdot\boldsymbol{\varphi}_i)_\Omega, \\ B^T_{ij} &= B_{ji} = (\nabla\cdot\boldsymbol{\varphi}_j, \chi_i)_\Omega, \\ b_i &= (\boldsymbol{f}, \boldsymbol{\varphi}_i)_\Omega. \end{aligned}$$

2.9 Free-Surface Stabilization Algorithm (FSSA)

In dealing with a time-dependent domain Ω, we employ Reynolds transport theorem, see Theorem 2.2.4, to handle the evolution of quantities within the domain. The theorem states [9]:

$$\frac{d}{dt}\int_{\Omega(t)} \phi d\Omega = \int_{\Omega(t)} \frac{d}{dt}\phi d\Omega + \int_{\partial\Omega(t)} (\boldsymbol{u}\cdot\boldsymbol{n})\phi d\Gamma.$$

Here ϕ is any scalar quantity, let's choose $\phi = (0, 0, -\rho g)\cdot\boldsymbol{v} = \boldsymbol{f}\cdot\boldsymbol{v}$, substituting this yields:

$$\frac{d}{dt}\int_{\Omega(t)} \boldsymbol{f}\cdot\boldsymbol{v} d\Omega = \int_{\Omega(t)} \frac{d}{dt}\boldsymbol{f}\cdot\boldsymbol{v} d\Omega + \int_{\partial\Omega(t)} (\boldsymbol{u}\cdot\boldsymbol{n})(\boldsymbol{f}\cdot\boldsymbol{v}) d\Gamma.$$

As $\boldsymbol{f} \cdot \boldsymbol{v}$ doesn't depend on time, the first term on the right-hand side equal zero:

$$\frac{d}{dt}\int_{\Omega(t)} \boldsymbol{f} \cdot \boldsymbol{v} d\Omega = \int_{\partial\Omega(t)} (\boldsymbol{u} \cdot \boldsymbol{n})(\boldsymbol{f} \cdot \boldsymbol{v}) d\Gamma.$$

This yields:

$$\frac{d}{dt}(\boldsymbol{f}, \boldsymbol{v})_{\Omega(t)} = (\boldsymbol{u} \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma}. \tag{2.30}$$

We approximate the time derivative using Forward Euler, in Section 2.3.1:

$$\frac{d}{dt}(\boldsymbol{f}, \boldsymbol{v})_{\Omega(t)} = \frac{(\boldsymbol{f}, \boldsymbol{v})_{\Omega^{k+1}} - (\boldsymbol{f}, \boldsymbol{v})_{\Omega^k}}{\Delta t}.$$

Equating this with the previous expression (2.30), we obtain:

$$\frac{(\boldsymbol{f}, \boldsymbol{v})_{\Omega^{k+1}} - (\boldsymbol{f}, \boldsymbol{v})_{\Omega^k}}{\Delta t} = (\boldsymbol{u} \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma^k}.$$

This leads to [12]:

$$(\boldsymbol{f}, \boldsymbol{v})_{\Omega^{k+1}} = (\boldsymbol{f}, \boldsymbol{v})_{\Omega^k} + \Delta t(\boldsymbol{u} \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma^k}. \tag{2.31}$$

We can now relate the last term to the free-surface equation. A normal to the surface, see Definition 2.1.6, is determined by the surface equation $h(x, y, t)$:

$$\boldsymbol{n} = \frac{(\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}, -1)}{\sqrt{(\frac{\partial h}{\partial x})^2 + (\frac{\partial h}{\partial y})^2 + (-1)^2}}. \tag{2.32}$$

The last term in (2.31) is an integral over the surface, where the surface interval is $[-L, L]$:

$$\Delta t(\boldsymbol{u} \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma_s^k} = \Delta t \int_{-L}^{L} (\boldsymbol{u} \cdot \boldsymbol{n})(\boldsymbol{f} \cdot \boldsymbol{v}) d\Gamma_s. \tag{2.33}$$

The arc length over Γ_s is:

$$d\Gamma_s = \sqrt{(\frac{\partial h}{\partial x})^2 + +(\frac{\partial h}{\partial y})^2 + (-1)^2} dt, \tag{2.34}$$

where t represents a parameter along the surface boundary. If we include the normal in (2.32) and the surface boundary in (2.34), we can express the integral in (2.33) as:

$$\begin{aligned}
&\Delta t \int_{-L}^{L} (\boldsymbol{u} \cdot \boldsymbol{n})(\boldsymbol{f} \cdot \boldsymbol{v}) d\Gamma_s = \\
&= \Delta t \int_{-L}^{L} (\boldsymbol{u} \cdot \frac{(\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}, -1)}{\sqrt{(\frac{\partial h}{\partial x})^2 + (\frac{\partial h}{\partial y})^2 + (-1)^2}})(\boldsymbol{f} \cdot \boldsymbol{v}) \sqrt{(\frac{\partial h}{\partial x})^2 + (\frac{\partial h}{\partial y})^2 + (-1)^2} dt \\
&= \Delta t \int_{-L}^{L} (\boldsymbol{u} \cdot (\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}, -1))(\boldsymbol{f} \cdot \boldsymbol{v}) dt = \int_{-L}^{L} \Delta t(u_x \frac{\partial h}{\partial x} + u_y \frac{\partial h}{\partial y} - u_z)(\boldsymbol{f} \cdot \boldsymbol{v}) dt.
\end{aligned} \tag{2.35}$$

Upon observation, term:

$$\Delta t(u_x \frac{\partial h}{\partial x} + u_y \frac{\partial h}{\partial y} - u_z),$$

matches the right-hand-side of the free-surface equation (2.20) but discretized by means of Forward Euler method, see Section 2.3.1. To fully couple the terms in (2.35) to the free-surface equation we also add the mass accumulation term a_s from (2.20) to (2.31):

$$(\boldsymbol{f}, \boldsymbol{v})_{\Omega^{k+1}} = (\boldsymbol{f}, \boldsymbol{v})_{\Omega^k} + \Delta t((\boldsymbol{u} + a_s \boldsymbol{z}) \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma_s^k}. \tag{2.36}$$

As previously discussed in Section 2.7 the implicit method is unfortunately not feasible. Instead we utilize the Free-Surface Stabilization Algorithm (FSSA), which leverages the Reynolds transport theorem to mimic an implicit time-stepping scheme without the extra cost of iterating [11]. Using (2.36), we derived from Reynolds transport theorem for (2.29), we can compute the velocity and pressure of the next time step while still integrate over the current domain. We estimate $(\tilde{\boldsymbol{u}}^{k+1}, \tilde{p}^{k+1}) \in \boldsymbol{V} \times Q$ such that,

$$(2\mu(\nabla \tilde{\boldsymbol{u}}^{k+1})\nabla \tilde{\boldsymbol{u}}^{k+1}, \nabla \boldsymbol{v})_{\Omega^k} - (\tilde{p}^{k+1}, \nabla \cdot \boldsymbol{v})_{\Omega^k} - (\nabla \cdot \tilde{\boldsymbol{u}}^{k+1}, q)_{\Omega^k} + (\beta^2 \tilde{\boldsymbol{u}}^{k+1}, \boldsymbol{v})_{\Gamma_b^k} =$$

$$(\boldsymbol{f}, \boldsymbol{v})_{\Omega^k} + \Delta t\theta((\tilde{\boldsymbol{u}}^{k+1} + a_s^k) \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma_s^k}.$$

Separating the last term for more clarity:

$$(2\mu(\nabla \tilde{\boldsymbol{u}}^{k+1})\nabla \tilde{\boldsymbol{u}}^{k+1}, \nabla \boldsymbol{v})_{\Omega^k} - (\tilde{p}^{k+1}, \nabla \cdot \boldsymbol{v})_{\Omega^k} - (\nabla \cdot \tilde{\boldsymbol{u}}^{k+1}, q)_{\Omega^k} + (\beta^2 \tilde{\boldsymbol{u}}^{k+1}, \boldsymbol{v})_{\Gamma_b^k} -$$

$$\Delta t\theta((\boldsymbol{u}^{k+1} \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma_s^k} = (\boldsymbol{f}, \boldsymbol{v})_{\Omega^k} + \Delta t\theta(a_s^k \cdot \boldsymbol{n}, \boldsymbol{f} \cdot \boldsymbol{v})_{\Gamma_s^k}, \quad \forall \boldsymbol{v} \in \boldsymbol{V} \text{ and } \forall q \in Q. \tag{2.37}$$

This equation incorporates a parameter $\theta \in [0, 1]$ to control the level of implicitness in the function. When $\theta = 0$ the function is fully explicit with respect to $\boldsymbol{u}$ and when $\theta = 1$ it becomes semi-implicit [12].

2.10 Scalable Nonlinear Equations Solvers (SNES)

SNES is a toolkit in FEniCS that employs Newton's method, see Section 2.3.4 to solve nonlinear equations numerically. We are using this to implement the positivity preserving constraint, that the surface height should always stay above the bedrock height. For a surface $h(x, y, t)$ and the bedrock $b(x, y)$ we want to implement that $h(x, y, t) > b(x, y)$. To do this we will be looking at the free-surface equation (2.22):

$$(h^{k+1}, v)_{\Gamma_s^k} + \Delta t(u_x^k \frac{\partial h^{k+1}}{\partial x} + u_y^k \frac{\partial h^{k+1}}{\partial y}, v)_{\Gamma_s^k} = (h^k, v)_{\Gamma_s^k} + \Delta t(u_z^k + a_s^k, v)_{\Gamma_s^k} \Leftrightarrow$$

$$a(h^{k+1}, v) = L(v), \quad \forall v \in V^s.$$

In every iteration we want to find h^{k+1}, that follows the constraint $h^{k+1} > b$. To implement this, SNES uses the Lagrange multiplier method, see Section 2.3.6. The objective function that we want to minimize is:

$$f(h^{k+1}) = a(h^{k+1}, h^{k+1}) - L(h^{k+1}).$$

We formulated the constraint as $h^{k+1} - b - s^2 = 0$ where s is a slack term so that our constraint function becomes,

$$g(h^{k+1}, s) = h^{k+1} - b - s^2.$$

We can formulate the Lagrangian formation:

$$\mathcal{L}(h^{k+1}, s, \lambda) = f(h^{k+1}) - \lambda g(h^{k+1}, s).$$

The minimization problem is:

$$\min_{h^{k+1}, s, \lambda} \mathcal{L}(h^{k+1}, s, \lambda) \tag{2.38}$$

We then obtain the constrained h^{k+1} by solving the system of equations [3]:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial h^{k+1}} &= 0, \\ \frac{\partial \mathcal{L}}{\partial s} &= 0, \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= 0. \end{aligned}$$

2.11 Relation between the SNES constrained free-surface equation and the FSSA stabilization term

In this thesis, we use FEM to discretize the Stokes equations resulting in (2.29). Through the incorporation of the FSSA term (2.36), we enable larger simulation time steps, denoted in (2.37).

Moreover, in Section 2.9 we demonstrated that the FSSA term itself, expressed in (2.36), establishes a coupling between the Stokes problem (2.29) and the free-surface equation, (2.20). The FSSA term entails the left-hand side of the free-surface equation as depicted in (2.35).

However, this connection is limited to the original free-surface equation. The SNES algorithm, outlined in 2.10 essentially modifies the free-surface equation (2.20) such that its solution obeys the constraint, ensuring that $h(x, y, t) > b(x, y)$, where $h(x, y, t)$ represents the ice sheet surface height, and $b(x, y)$ denotes the ice sheet bedrock height.

As an illustration, we take this into account by reformulating the free-surface equation (2.20) as:

$$\frac{\partial h}{\partial t} = -\boldsymbol{u}_{\perp} \cdot \nabla_{\perp} h + u_z + a_s + \text{SNES term}. \tag{2.39}$$

Here, we introduce an additional term called SNES that, after the equation is solved, effectively raises the surface height, ensuring its position is over the bedrock.

Upon examining the derived connection between the original free-surface equation (2.20) and the FSSA stabilization term (2.36), it becomes apparent that the SNES term, present on the right-hand-side of (2.39), is absent. This makes it evident that further investigation concerning the integration of the SNES term within the context of this thesis is needed.

2.12 Regions over the ice sheet surface where SNES is active

We have the positivity preserving constraint $h^{k+1} \geq b^k + \delta$, where δ is the glaciers initial ice thickness. We approximate $h^{k+1} = h^k + \Delta t(\boldsymbol{u}^k \cdot \boldsymbol{n}^k)$, where $\Delta t(\boldsymbol{u}^k \cdot \boldsymbol{n}^k)$ is the displacement of the surface into normal direction. We use this in the positivity preserving constraint to arrive at:

$$h^k + \Delta t(\boldsymbol{u^k} \cdot \boldsymbol{n}^k) \geq b^k + \delta.$$

This will be satisfied when:

$$\boldsymbol{u}^k \cdot \boldsymbol{n}^k \geq \frac{1}{\Delta t}(b^k - h^k + \delta). \tag{2.40}$$

Thus, the approximate regions where the positivity preserving constraint is not satisfied are given as a set below:

$$\tilde{\alpha} = \{x \in \Gamma_s \,|\, \boldsymbol{u}^k \cdot \boldsymbol{n}^k < \frac{1}{\Delta t}(b^k - h^k + \delta)\}, \tag{2.41}$$

where $\tilde{\alpha}$ represent the regions on the surface Γ_s where the SNES algorithm is active.

2.13 The goals of the thesis

In this thesis we examine simulation errors arising due to the fact that the SNES term in (2.39), is not taken into account when integrating the FSSA stabilization term (2.36) into the Stokes problem (2.29). We make the examination by a series of numerical experiments. Furthermore, we develop a strategy to eliminate or decrease those errors.

3 Numerical experiments

In Section 3.1 we define the glacier we conduct the simulations on. In Section 3.2 we conduct simulation to find a stable time step for when we are not using the FSSA. Then we will conduct four experiments to investigate the connection between the surface error and the activation of the positivity preserving constraint when the FSSA stabilization term is added to the Stokes problem. In Section 3.3 we determine if any error appear at all and compare the relative spatial error distribution on a linear scale with that on a logarithm scale to determine the most effective visualization method.
In Section 3.4 we show the errors throughout the simulation, observing how they fluctuates when accumulation is active.
In Setion 3.5, we extend the total simulation duration to observe the changes in error when the accumulated snow is melted.
Finally, in Section 3.6, we closely examine the velocity field, comparing the errors of two closely spaced times when the glacier experiences significant mass loss.

3.1 Löfgren's glacier (Perlin)

Here we describe a simulation of a glacier geometry in two dimensions from (Lofgren, 2023) [11]. At the initial time, denoted as $t = 0$, the glacier consists of a thin ice layer, which progressively thickens due to positive accumulation. Figure 3.1 illustrates the temporal evolution over an extended period as the glacier accumulates snow. The accumulation process, represented by a non-negative function, is described by a linearly decaying function:

$$a_s(x, y = 0) = \begin{cases} 2, & \text{if } r(x,y) \leq 0.25, \\ 2(1 - \frac{r(x,y)-0.25}{0.75}), & 0.25 \leq r(x,y) \leq 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Here, $r(x, y)$ is an anisotropic distance function, with (x_c, y_c) denoting the center, defined as :

$$r(x,y) = \sqrt{\left(\frac{x - x_c}{a}\right)^2 + \left(\frac{y - y_c}{b}\right)^2}.$$

The parameters $a = 750$ and $b = 1500$ represent the distance from the center to the vertices along the x-axis and y-axis, respectively. In our experiments we use a_s as defined in (3.1), when $t < T_{melt}$. Then when $t \geq T_{melt}$ we use $a_s = -1$. Here, T_{melt} is the time where we switch from accumulation mode to melting mode.

The boundary conditions at Γ_W, Γ_E are set according to (2.16). On the surface Γ_s the free-surface equation, as per Equation (2.20), is applied. For the bedrock (2.16)-(2.18) are employed depending on whether the glacier is frozen to the bedrock or not. In (2.18), the drag coefficient β^2 is defined as:

$$\beta^2(x) = \beta^2_{\text{min}} + \frac{\beta^2_{\text{max}} - \beta^2_{\text{min}}}{1 + e^{(\frac{x-\mu}{\sigma})}}.$$

Here, β_{min} and β_{max} denotes the minimum and maximum values of β, respectively. The parameters σ and μ are as in Table 3.1. Also A_0 is the same value that are used in Table 2.1. The simulation utilizes the Picard iteration method as described in Section 2.3.5 due to its independence from a precise initial guess, especially useful when the function's derivative might be non-existent or computationally expensive to compute. The mesh resolution is set to $(N_x, N_z) = (200, 5)$, where N_x and N_z represent the number of layers in the horizontal and vertical directions, respectively. In our computational framework, we include the variable θ as outlined in (2.37). This variable enables us to selectively apply the FSSA.

Table 3.1: Variables for Perlin glacier

β^2_{min}	0.01 MPa yr m^{-1}
β^2_{max}	1000 MPa yr m^{-1}
σ	200 m
μ	3000 m

3.2 Stable time step

In this section we establish the stability threshold for the time step size required to simulate the Perlin glacier model in absence of FSSA. This entails conducting simulations without FSSA and analyze the behavior of the velocity component u_x as the time step size, denoted as Δt, is increasing.

Velocity plots are generated for each time step during the simulation iterations to enable a comprehensive comparison. Figure 3.2 illustrates the velocity distribution at the simulation's end time $T = 100$ for four distinct step sizes. Our observations reveal that when the step size exceeds $\Delta t = 5$, oscillations emerge on the surface, signifying instability.

Additionally, we explore the surface plots represented in Figure 3.3, for the corresponding time steps, which further confirm instability for larger values than $\Delta t = 5$. Based on our analysis, we conclude that $\Delta t = 5$ is exhibiting stable behavior. We thus decide to use $\Delta t = 3$ for our experimental simulations.

3.3 Experiment 1: Surface error at the end of the simulation

Here, we conduct simulations both with and without FSSA. We use a stable time step size $\Delta t = 3$. The simulation ran until the end time $T = 100$, with melting beginning at $T_{melt} = 80$. In all cases the SNES constraint is turned on. Our aim is to investigate if the FSSA and the constraint by SNES entails errors defined as:

$$e(x_i, t) = \frac{|h_{FSSA}(x_i, t) - h_{NOFSSA}(x_i, t)|}{(\text{max i})|h_{NOFSSA}(x_i, t)|}, \quad i = 1, 2, .., N.$$

Here, h_{FSSA} represents the surface height when utilizing FSSA, and h_{NOFSSA} represents the surface height when not using FSSA, serving as the reference and true height. Additionally, e is calculated at a specific point x_i and a specific time t. In this experiment we measure the error at the end time, $t_{end} = 100$. The logarithmic scale helps with magnifying relatively small errors, which might otherwise be overshadowed.

In Figure 3.4, we depict surface and velocity plots at the simulation's end time, $T = 100$ for both vertical and horizontal velocity vectors, with melting initiated at $T_{melt} = 80$. Additionally in the same figure, we present two plots illustrating the error of the surface height at the end time. Initially, we observed significant error in only one specific area when plotting the error in linear scale , necessitating a logarithmic scale to enhance visualization of the error. We can see in Figure 3.4, that a logarithmic scale gives a fairer result, therefore we will use this in all further experiments.

Upon analyzing the plots, we observed that the majority of the error was approximately 10^{-16}, rendering it effectively negligible. However, a significant error at 10^{-2} was evident in the region extending from the glacier's start to $x = 2000$, as can be seen in the plot from Figure 3.4. The point where the error begins corresponds to the area where the glacier have accumulated a substantial mass, which starts to thin out at around $x = 1000$.

This suggests that the error may be attributed to the significant snow accumulation in the area, as in Section 2.12. Snow accumulating on the first thin surface likely triggers higher velocities, potentially necessitating the activation of the SNES algorithm. Additionally, the observation of high velocities in both vertical and horizontal components within the domain $x = 1000$ and $x = 2000$, where the surface height is relatively small further supports the need for SNES activation in this specific region.

3.4 Experiment 2: Surface error as a function of time

In the second experiment, we have the same setup as in the first experiment but we now look at the error as a function of time. In Figure 3.5 we display the velocity component u_x in the first four plots and u_z in the last four at different times T. Figure

3.6 visualizes the respective relative error on a logarithmic scale for each of these times.

Similar to the first experiment, a distinct error pattern emerges in Figure 3.6, persisting from $x = 0$ til $x = 2000$, consistetly present across all figure. This error pattern correlates with the domain where the glacier accumulates a significant amount of snow. As in Section 2.13, we hypothesized that the error would be prominent initially when the surface is thin and begins to accumulate snow. Furthermore, we observed that the error does not increase over time, suggesting that it is more pronounced at the beginning and then stabilizes as the surface accumulates mass, becoming less sensitive to SNES.

Moreover, as the error can be observed across the glacier's surface, a comparison of the two last plots in Figure 3.6, reveals a decrease in error between $T = 60 - 90$. However, considering that melting starts at $T_{melt} = 80$, and a significant portion of the mass has melted by $T = 90$, this reduction in error can also be attributed to the reduction in mass. As in Section 2.13, this error may be caused by the active SNES term, suggesting that it manifests where mass accumulation occurs.

3.5 Experiment 3: Surface error as a function of time for a longer simulation

In the third experiment we simulate the Perlin glacier using a time step size of $\Delta t = 3$ and an extended end time $T = 500$, with melting initiating at $T_{melt} = 240$.
Figure 3.7 displays surface and velocity plots for the u_x components at six different time steps, while Figure 3.8 depicts the surface and velocity plots for the u_z components at the corresponding six time steps. Additionally, Figure 3.9 presents the surface error on a logarithmic scale for these six distinct times.

Compared to the first and second experiments here we see that the glacier not only accumulates snow from the start til point $x = 2000$, it now gather mass up to the point $x = 3000$ and then further til $x = 4000$.
Here, we can also see from the surface plots that it also melts all of the accumulated snow when we appear at the end time $T = 500$. Therefore, we now have a simulation where we can see the glacier is adding on mass on a larger domain then before and also melting it back to the starting thickness.

If we investigate the error plot in the first three plots in Figure 3.9 we can see that similar to the first and second experiment we have the largest error at the place where we accumulate the most snow. It can also be said that the doesn't seem to grow explained in the second experiment this is probably due to the fact that the error is large when the glacier begins to accumulate snow and then when it has gathered mass the downward normal velocity is not so large to make it sensitive to the constraint.

In Figure 3.9 at $T = 240$ we have a lot of errors through out the surface, but when the melting start at $T = 240$ we see in the next plot at $T = 300$ a lot of this accumulated snow have melted only remaining the larger error.
As in the second experiment we can connect this to the accumulated mass is affecting the error.
Then in the fifth plot in Figure 3.9 where $T = 240$ we can see that there is almost none of the errors left. If we compare this to the Figure 3.7 we can see that this correspond to the last part where there is mass. Then in the last of the error plots we can see that all of the snow has melted and the error is gone which further confirmed the correlation of SNES being active when the glacier collect mass.

3.6 Experiment 4: Surface error and the change of velocities between two time points

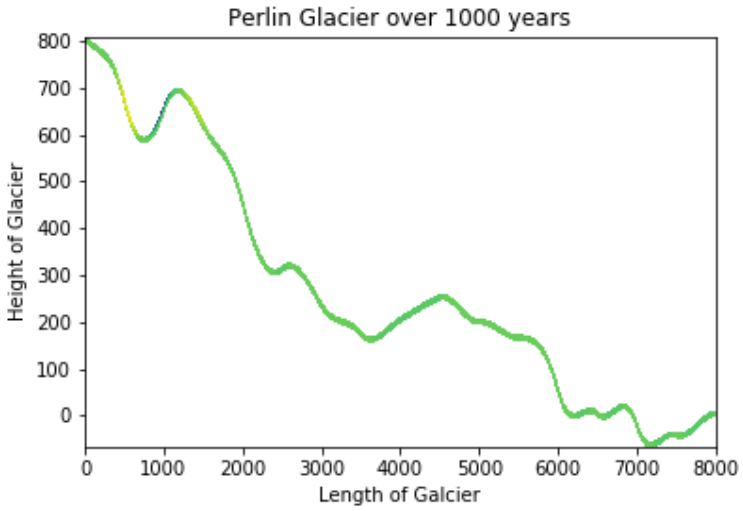
Similar to the third experiment we compute a simulation where the end time is $T = 500$, with time step size $\Delta t = 3$ and we have that melting starts at $T = 240$. Here, we want to look closer at the velocity of the vertical and horizontal component at the times, $T = 240$ and $T = 300$.

In Figure 3.10 the first two plots we first see the surface and bedrock at times $T = 240$ and $T = 300$ making it obvious that the glacier has lost parts of the glacier due to the melting. The next four plots are of the velocity the first two represents the velocity component of u_x at $T = 240$ respectively $T = 300$. The next two is of the velocity components u_z at time $T = 240$ and $T = 300$. Then we included the error plots to compare the error at the times $T = 240$ and $T = 300$.

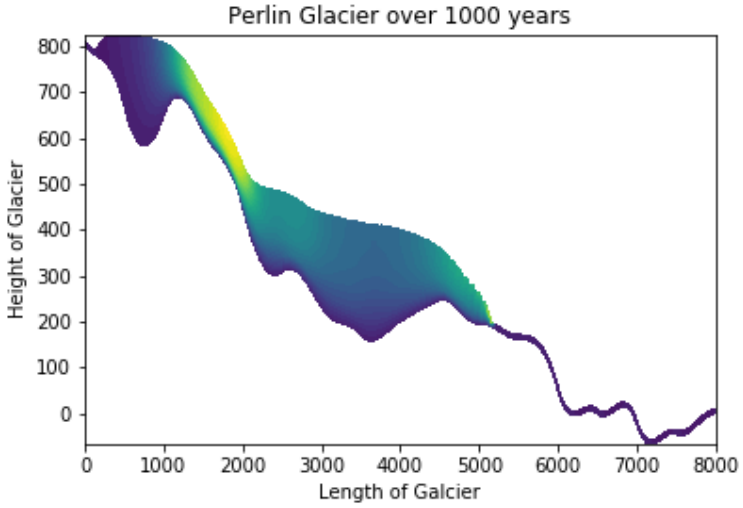
Looking at the plots in Figure 3.10, where $T = 240$, the velocity u_x is at the highest value at point $x = 2000$. We also see that the u_z component is large at this time, but in a negative direction. If we compare this to the error plot when $T = 240$, we here have a large error at (10^{-2}) and looking at the surface we see that at point $x = 2000$ the glacier has accumulated a bit of snow.

If we then look at Figure 3.10, where $T = 300$, we see that the velocity components have decreased significantly both in horizontal and vertical direction at the point $x = 2000$. The surface plots also show that the ice at this point has melted, going back to the original thickness. At the same time the error around $x = 2000$ became of size 10^{-16}. We observe closer that the velocity and the thickness of the ice affect the error in the regions where SNES should be active this supports the hypothesis that this is the cause of the error.

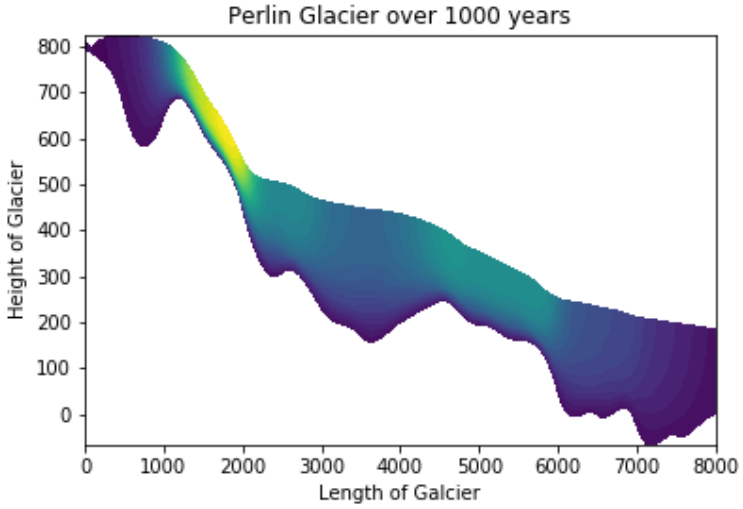
The conclusion to make is as in Section 2.12 that when we have a proportionally large velocity compared to the thickness of the ice this creates an error due to the fact that SNES is active but the FSSA does not take this into account.



$T = 0$.

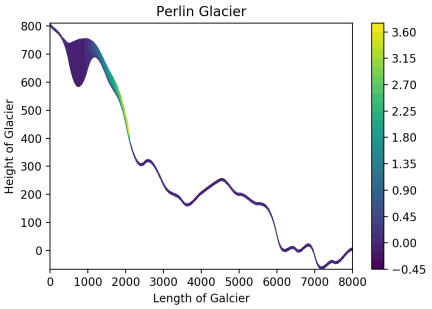


$T = 500$.

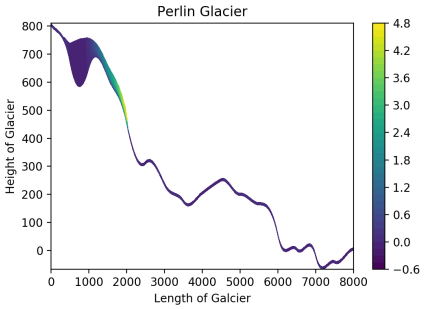


$T = 1000$.

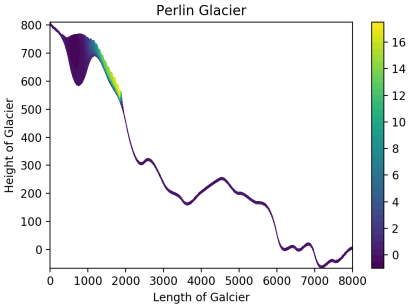
Figure 3.1: The surface and velocity component u_x (meter/years) of the Perlin glacier. The glaciers length is $L_x = 8000\ m$ and the height is $L_z = 800\ m$. We can see that the accumulation function is adding mass through the years with end time $T = 1000$ years.



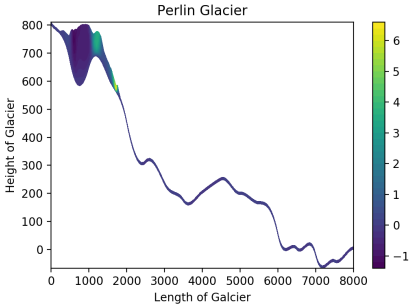
$\Delta t = 3.$



$\Delta t = 5.$

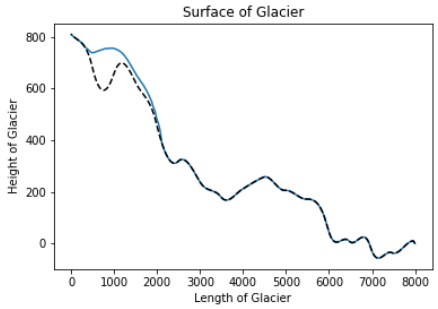


$\Delta t = 13.$

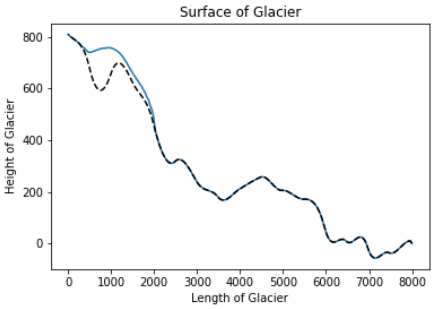


$\Delta t = 25.$

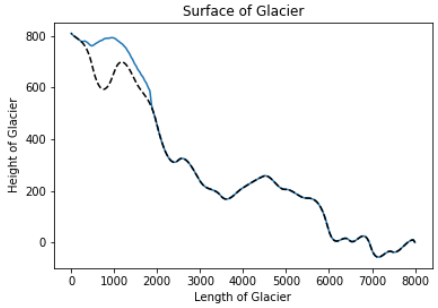
Figure 3.2: The Perlin glacier simulated with various time step sizes, denoted as Δt, depicted at the end time $T = 100$, all conducted without utilizing FSSA. This analysis aims to show how stability varies with increasing the time step size.



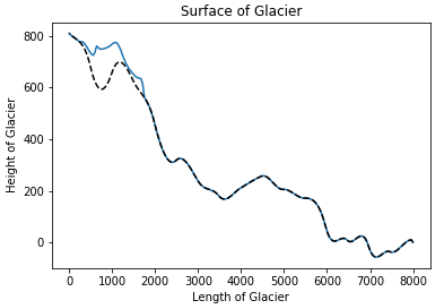
$\Delta t = 3$.



$\Delta t = 5$.

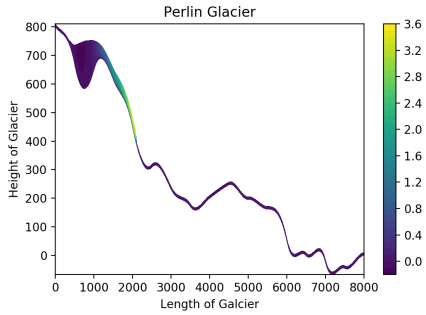


$\Delta t = 13$.

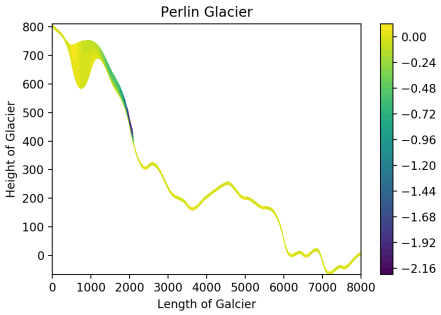


$\Delta t = 25$.

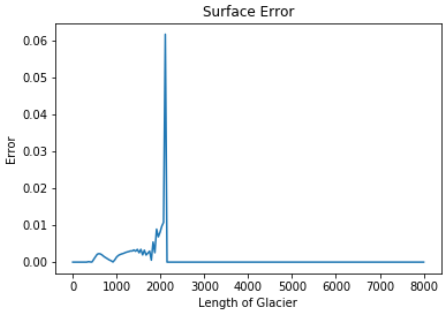
Figure 3.3: The simulation of the Perlin glacier surface during different time step sizes, denoted as Δt, depicted at $T = 100$ without the utilization of FSSA. This examination aims to highlight the differences in stability as the time step size is increased.



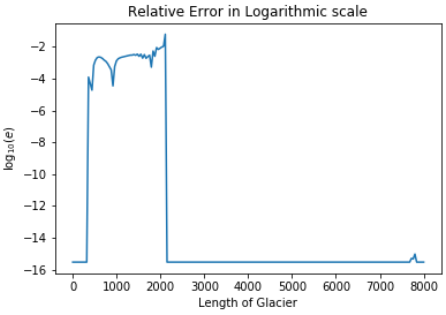
u_x, $T = 100$.



u_z, $T = 100$.

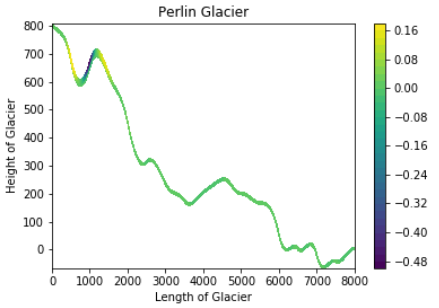


Error in linear scale, $T = 100$.

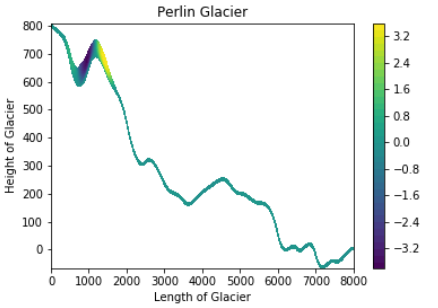


Relative error in logarithm scale, $T = 100$.

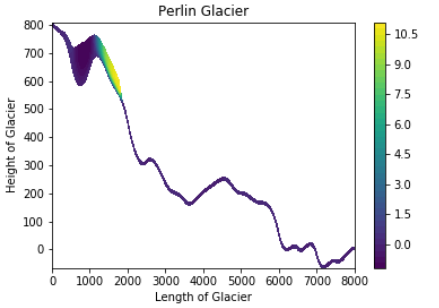
Figure 3.4: Experiment 1: The Perlin glacier's surface and velocity when simulating with a time step size of $\Delta t = 3$. Depicted at the end time $T = 100$ and melting starts at the time $T_{melt} = 80$. The relative error of the surface height is depicted in linear scale and in logarithm scale.



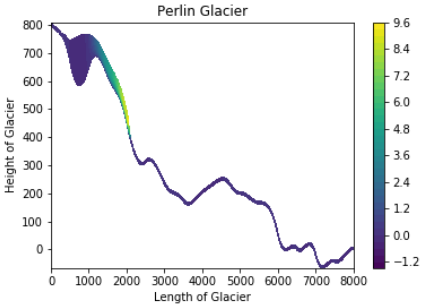
u_x, $T = 9$.



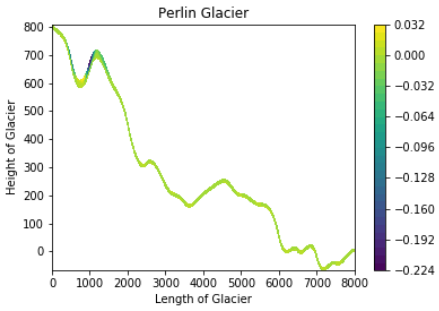
u_x, $T = 30$.



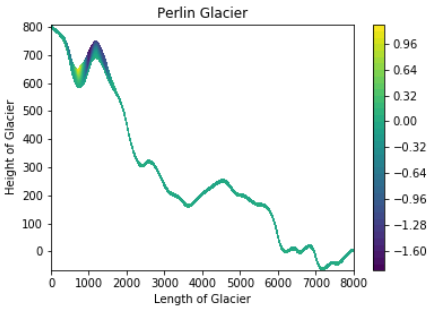
u_x, $T = 60$.



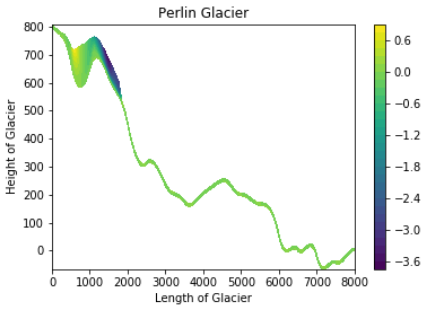
u_x, $T = 90$.



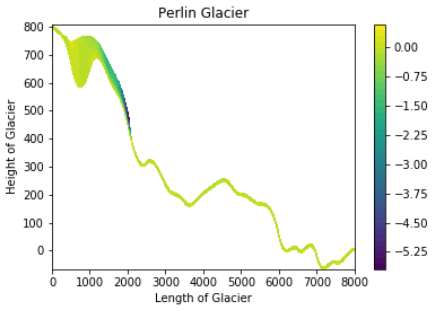
u_z, $T = 9$.



u_z, $T = 30$.

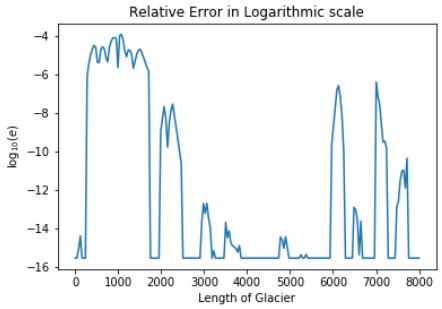


u_z, $T = 60$.

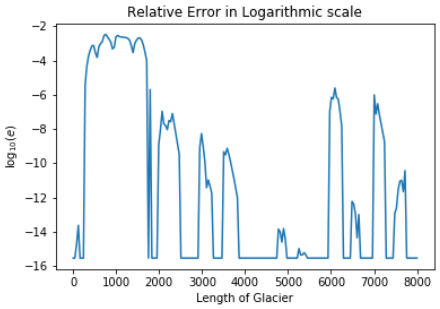


u_z, $T = 90$.

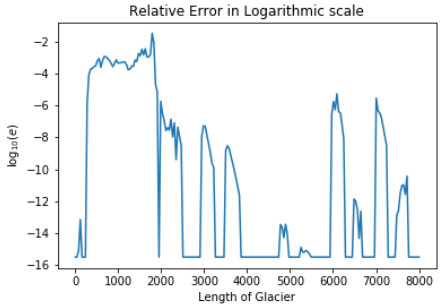
Figure 3.5: Experiment 2: The surface and velocity of the Perlin glacier, including u_x and u_z, are examined at four distinct time points T. With a time step size of $\Delta t = 3$ and melting of the glacier begins at the time $T_{melt} = 80$.



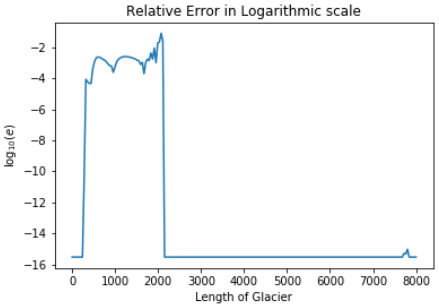
Error, $T = 9$.



Error, $T = 30$.

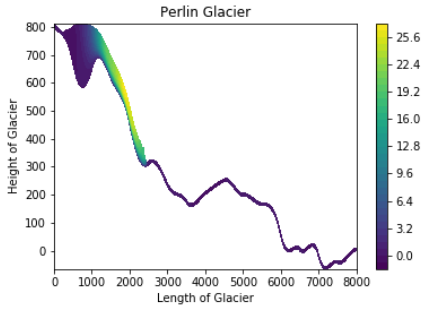


Error, $T = 60$.

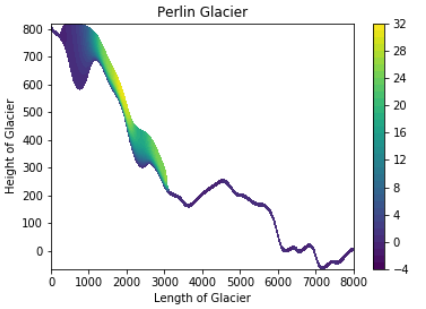


Error, $T = 90$.

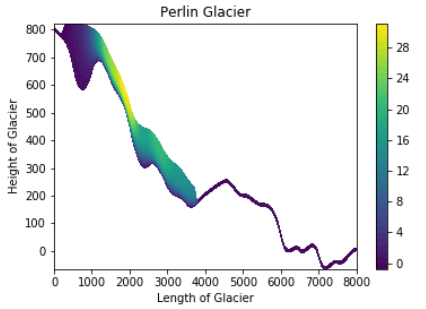
Figure 3.6: Experiment 2: The relative error in logarithm scale of the Perlin glacier's surface is evaluated at four time points T. The simulation extend until the end time $T = 100$. With a time step size of $\Delta t = 3$ and melting of the glacier begins at the time $T_{melt} = 80$.



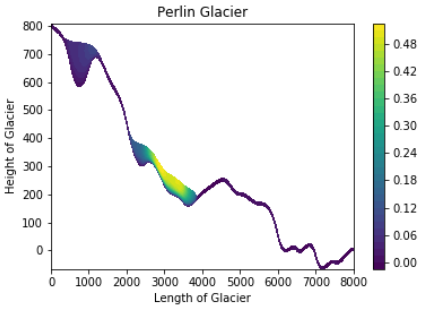
u_x, $T = 120$.



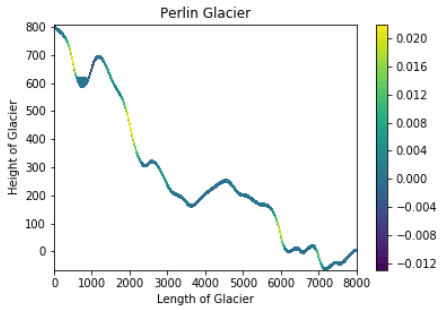
u_x, $T = 180$



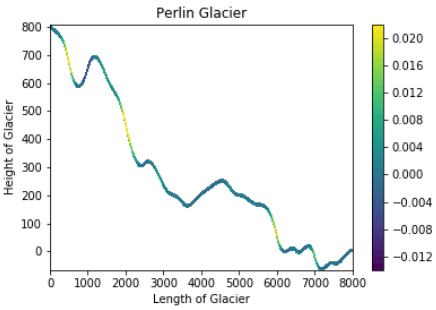
u_x, $T = 240$



u_x, $T = 300$

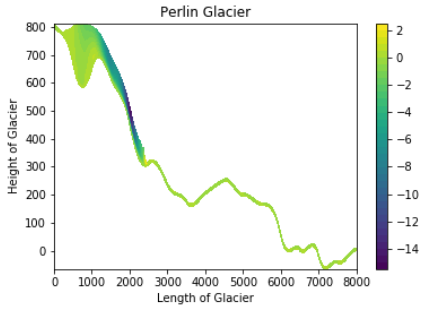


u_x, $T = 420$

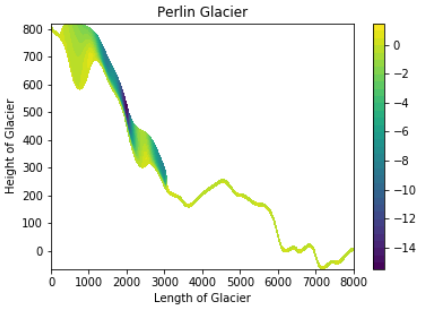


u_x, $T = 480$

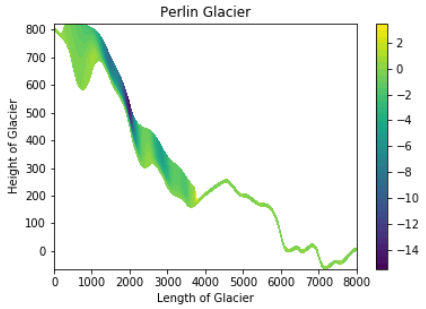
Figure 3.7: Experiment 3: The surface and velocity (u_x) of the Perlin glacier are examined at six distinct time points T. With a time step size of $\Delta t = 3$ and melting of the glacier begins at $T_{melt} = 240$.



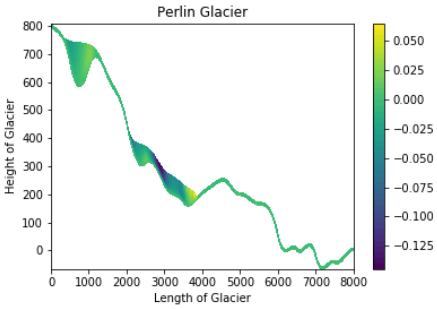
u_z, $T = 120$



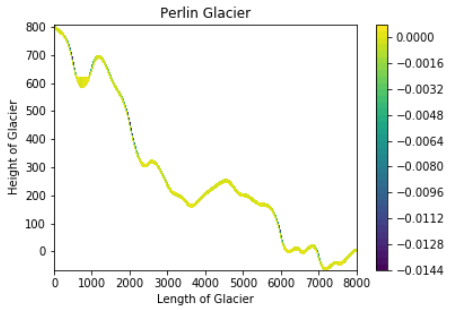
u_z, $T = 180$



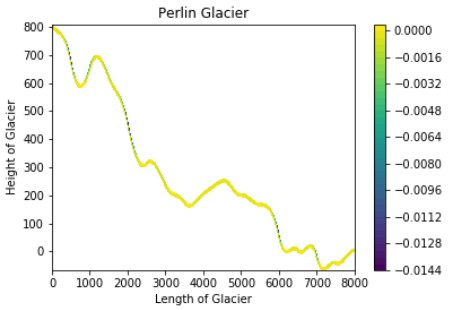
u_z, $T = 240$



u_z, $T = 300$

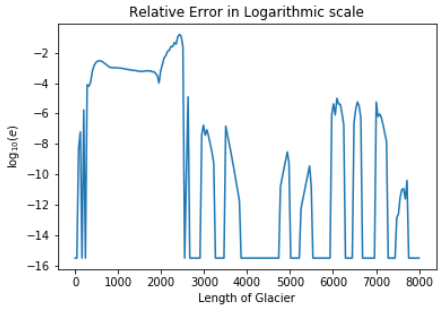


u_z, $T = 420$

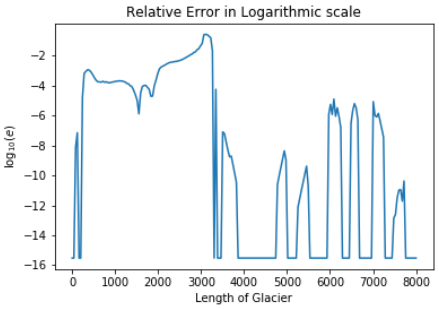


u_z, $T = 480$

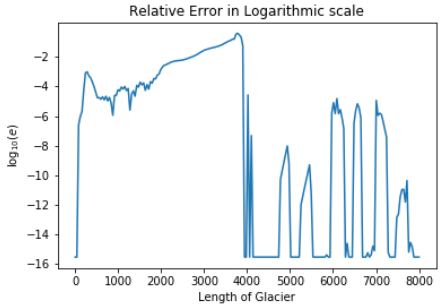
Figure 3.8: Experiment 3: The surface and velocity (u_z) of the Perlin glacier are examined at six distinct time points T. With a time step size of $\Delta t = 3$ and melting of the glacier begins at the time $T_{melt} = 240$.



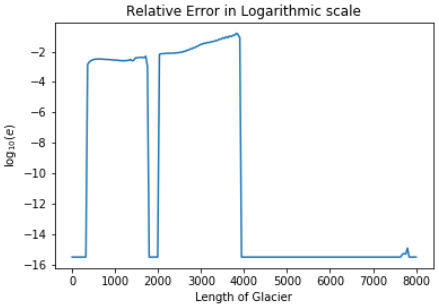
Error, $T = 120$.



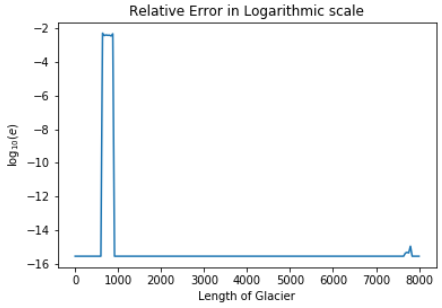
Error, $T = 180$.



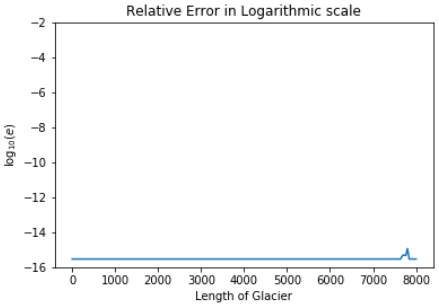
Error, $T = 240$.



Error, $T = 300$.

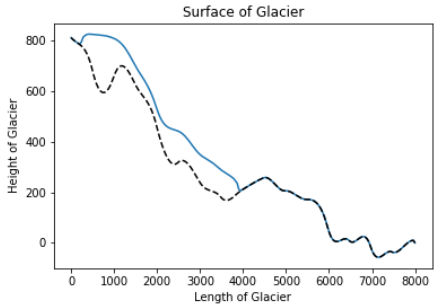


Error, $T = 420$.

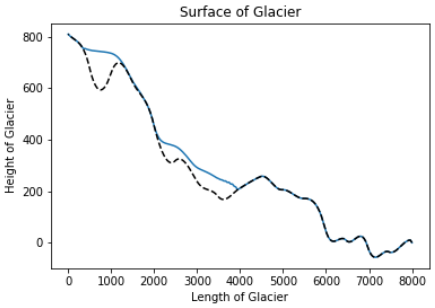


Error, $T = 480$.

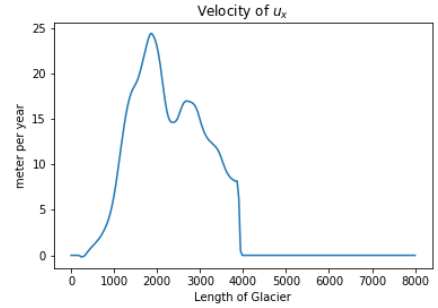
Figure 3.9: Experiment 3: The relative error in logarithmic scale of the Perlin glacier's surface is evaluated at six time points T. The simulation extend until the end time $T = 500$. With a time step size of $\Delta t = 3$ and melting of the glacier begins at the time $T_{melt} = 240$.



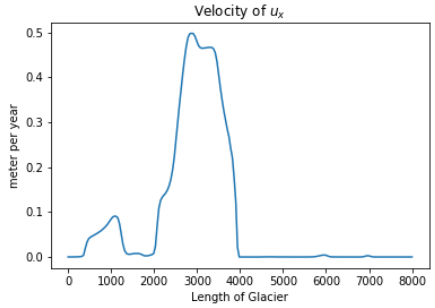
Surface, $T = 240$.



Surface, $T = 300$.

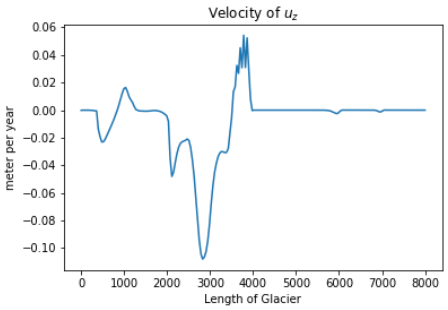


u_x, $T = 240$.

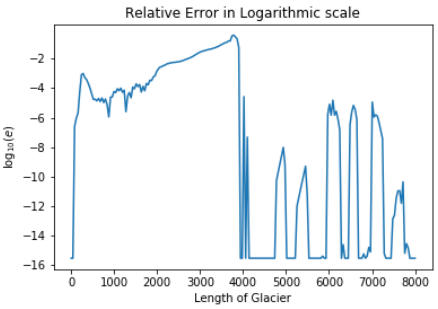


u_x, $T = 300$.

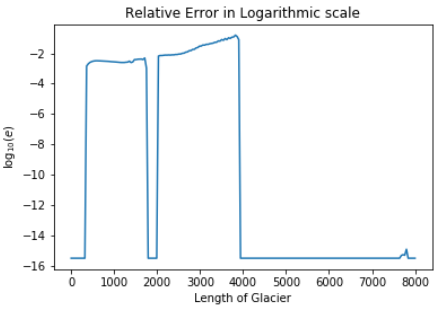
u_z, $T = 240$.



u_z, $T = 300$.



Error, $T = 240$.



Error, $T = 300$.

Figure 3.10: Experiment 4: Comparing the simulations error,the velocity components u_x and u_z and the surface height, at the time points $T = 240$ and $T = 300$.

4 Two candidate remedies for enhancing the FSSA stabilization term with the surface positivity preserving constraint

In the Section 3 we found the presence of the error when the SNES algorithm is active and the FSSA stabilization term is added to the Stokes problem. Here we propose two candidate remedies to augment the FSSA stabilization term with an approximate positivity preserving constraint on the velocity field.

The first remedy is to solve the Stokes problem (2.37), with:

$$(\boldsymbol{u}^k \cdot \boldsymbol{n}^k) \geq \frac{1}{\Delta t}(b^k - h^k + \delta), \quad \Gamma_s,$$

added as a requirement. Implementing this by employing the SNES algorithm is cumbersome, as imposing an inequality constraint over the domain boundary is not a natural part of the code interface.

The second remedy is to compute the regions where the SNES algorithm is active by using $\tilde{\alpha}$ as given in (2.41) and then impose the condition:

$$(\boldsymbol{u}^k \cdot \boldsymbol{n}^k) = \frac{1}{\Delta t}(b^k - h^k + \delta), \quad \text{on } \tilde{\alpha} \subset \Gamma_s.$$

This condition would be easier to implement in the FEniCS library then the first remedy. In this thesis we have not attempted to use the two proposed remedies. We leave that to the research community as future work.

5 Final remarks

The central focus of this thesis addresses numerical instabilities encountered in simulations of ice sheet dynamics, specifically when the ice sheet surface height approaches the underlying bedrock height.

Previous studies have demonstrated that incorporating the FSSA stabilization term (2.36) with the Stokes problem discussed in (2.29) effectively increases the time step size, when the velocities computed from the Stokes problem are used in the free-surface equation.

In Section 2.11 we showed that the FSSA stabilization term is supposed to take the action of the SNES algorithm into account. In Section 3, we numerically showed that the SNES positivity preserving algorithm combined with the FSSA, gives rise to an error.

In Section 4 we then propose two candidate remedies for including a positivity preserving constraint to the FSSA stabilization term. We leave their implementation and evaluation for future work.

Other future work also entails investigations over a more realistic three-dimensional ice sheet domain, as well as a study of the effect of the mesh size on the error.

The significance of this study is a contribution to enhancing the FSSA, which in turn improves the reliability of predictions concerning glacier dynamics, crucial for assessing their impact on climate change.

Bibliography

[1] Climate and ice. `https://scied.ucar.edu/learning-zone/climate-change-impacts/climate-and-ice`. (Accessed: 2024-02-26).

[2] Parts of the cryosphere. `https://scied.ucar.edu/learning-zone/earth-system/parts-cryosphere`. (Accessed: 2024-02-26).

[3] SNES: Nonlinear Solvers. `https://petsc.org/main/manual/snes/`. (Accessed: 2024-04-30).

[4] The FEniCS computing platform. `https://fenicsproject.org/`. (Accessed: 2024-04-30).

[5] Jerrold E. Marsden Anthony Tromba. *"Vector Calculus"*. Freeman, 2016.

[6] Judi McDonald David Lay, Steven Lay. *"Linear Algebra and Its Applications, Global Edition "*. PEARSON EDUCATION LIMITED, 2021.

[7] V. Masson-Delmotte P. Zhai M. Tignor E. Poloczanska K. Mintenbeck A. Alegría M. Nicolai A. Okem J. Petzold B. Rama N.M. Weyer (eds.) H.-O. Portner, D.C. Roberts. *IPCC, 2019: IPCC Special Report on the Ocean and Cryosphere in a Changing Climate*. 2019. (Accessed: 2024-02-26).

[8] Kent-Andre Mardal Hans Petter Langtangen1. *"Introduction to Numerical Methods for Variational Problems"*. Springer Nature Switzerland AG, 2016.

[9] Volker John. *"Methods For Incompressible Flow Problems"*. Springer, 2016.

[10] Mats G Larson and Fredrik Bengtzon. *"The Finite Element Method: Theory, Implementation and Practice"*. Springer-Verlag Berlin and Heidelberg GmbH Co. K, 2013.

[11] A. Löfgren, T. Zwinger, P. Råback, C. Helanow, and J. Ahlkrona. Increasing numerical stability of mountain valley glacier simulations: implementation and testing of free-surface stabilization in elmer/ice. *EGUsphere*, 2023:1–22, 2023.

[12] André Löfgren, Josefin Ahlkrona, and Christian Helanow. "increasing stable time-step sizes of the free-surface problem arising in ice-sheet simulations". *Journal of Computational Physics: X*, 16:100114, 2022.

[13] Frank Pattyn, Catherine Ritz, Edward Hanna, Xylar Asay-Davis, Rob DeConto, Gaël Durand, Lionel Favier, Xavier Fettweis, Heiko Goelzer, Nicholas R Golledge, et al. The greenland and antarctic ice sheets under 1.5 c global warming. *Nature climate change*, 8(12):1053–1061, 2018.

[14] Heinz Blatter Ralf Greve. *"Dynamics of Ice Sheets and Glaciers"*. Springer Berlin, Heidelberg, 2009.

[15] Charles R. Johnson Roger A. Horn. *"Matrix Analysis"*. Cambridge University Press; 2nd edition, 2012.

[16] Walter Rudin. *"Functional analysis"*. McGraw-Hill Science/Engineering/Math; 2nd edition, 1991.

[17] Felix Kwok Walter Gander, Martin J. Gander. *"Scientific Computing - An Introduction using Maple and MATLAB"*. Springer Cham, 29 April 2014.

[18] Johannes Weertman. On the sliding of glaciers. *Journal of glaciology*, 3(21):33–38, 1957.

Datalogi
www.math.su.se

Beräkningsmatematik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm