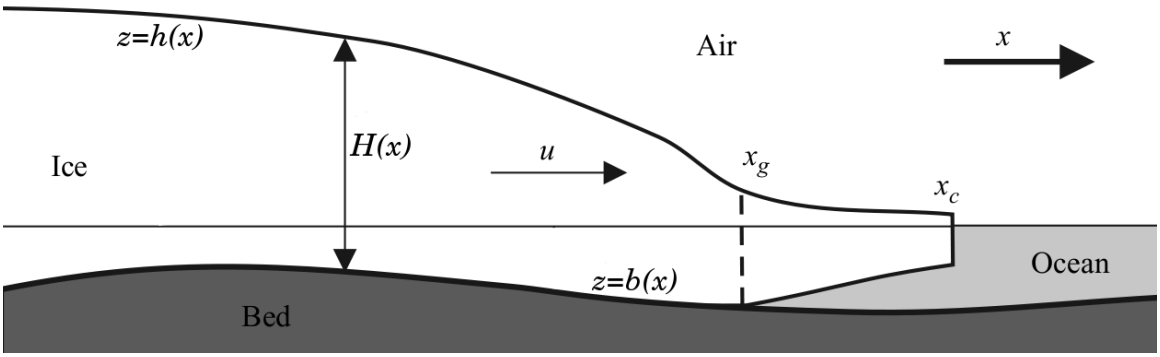
Numerical modelling of ice sheets, streams, and shelves

Ed Bueler

Karthaus, September 2012



Illustrates the notation used in this chapter. Figure modified from [?].

1.1 Introduction

Numerical models can actually help you, the reader, understand the behavior of particular ice flows. Their most common use may be to help you ask: When I put together my incomplete understanding of glacier processes into a mathematical model, does the combination behave as I expect? Or why does it mis-behave? Though numerical models rarely generate new theoretical understandings—a human brain seems to be needed to get new insights from observations—they can demonstrate flaws in our understanding of glacier processes, and they can illuminate how these processes combine to give overall behavior.

Numerical models should be built with care. The worst outcome is to spend time explaining, through physical argumentation and using observational evidence, numerical model behavior that is an artifact of poor computer programming or numerical analysis. The reader of this chapter may be surprised that a continuum model, and not a code, seems to be our focus much of the time. All numerical codes produce numbers, but we want numbers that actually come from our continuum model. We analyse numerical implementations to see if they match the continuum model and its solutions.

This chapter has a limited scope:

> *shallow approximations of ice flow.*

It adopts a constructive approach:

> *example numerical codes that actually work.*

Within our scope are the shallow ice approximation (SIA) in two horizontal dimensions (2D), the shallow shelf approximation (SSA) in 1D, and the mass continuity and surface kinematical equations. We recall the Stokes model, but we do not address its numerical solution. Our numerical ideas include finite difference schemes, solving algebraic systems from stress balances, and the verification of codes using exact solutions.

Our notation is common in the glaciological literature (Table 1). Coordinates are t, x, y, z, with z perpendicular to the geoid and positive-upward. If these coordinates appear as subscripts then they denote partial derivatives: $u_x = \partial u/\partial x$. Tensor notation uses subscripts from the list $\{1, 2, 3, i, j\}$, so that, for example, τ_{ij} and τ_{13} are entries of the deviatoric stress tensor.

This chapter is based on sixteen Matlab codes, each about one-half page, of which only five are printed in this text. All have been tested in Matlab and Octave. They are distributed in `.zip` form at

> `http://www.dms.uaf.edu/~bueler/karthaus/`

When displayed in these notes these codes have their comments stripped for compactness and clarity. The electronic versions have generous comments and help files.

1.2 Ice flow equations

My first goal in these notes is to get to an equation for which I can say:

> *by numerically solving this equation, we have a usable model for an ice sheet.*

Table 1: Notation used in this chapter.

variable	description	SI units
A	$A = A(T)$ = ice softness in the Glen flow law	$\text{Pa}^{-n}\,\text{s}^{-1}$
B	ice hardness; $B = A^{-1/n}$	$\text{Pa}\,\text{s}^{1/n}$
b	bedrock elevation	m
∇	(spatial) gradient	m^{-1}
$\nabla\cdot$	(spatial) divergence	m^{-1}
$\mathbf{g}$	gravity	m s^{-2}
H	ice thickness	m
h	ice surface elevation	m
M	climatic mass balance	m s^{-1}
n	exponent in Glen flow law	
ν	viscosity	Pa s
p	pressure	Pa
$\mathbf{q}$	map-plane ice flux: $\mathbf{q} = \int_b^h \mathbf{U}\,dx = \bar{\mathbf{U}}H$	$\text{m}^2\,\text{s}^{-1}$
ρ	density of ice	kg m^{-3}
ρ_w	density of sea water	kg m^{-3}
T	temperature	K
τ	magnitude of τ_{ij}: $2\tau^2 = \sum_{ij}\tau_{ij}^2$	Pa
τ_{ij}	deviatoric stress tensor	Pa
Du_{ij}	strain rate tensor	s^{-1}
$\mathbf{U}$	$= (u, v)$ horizontal ice velocity	m s^{-1}
$\mathbf{u}$	$= (u, v, w)$ 3D ice velocity	m s^{-1}

Of course a "usable" model is *understood* as much as it is *correct*. To get to my goal I will quickly recall the continuum mechanical equations of ice flow.

Ice in glaciers is a fluid, so we describe its motion by a velocity field $\mathbf{u}(t, x, y, z)$. If the ice fluid were faster-moving than it actually is, and if it were linearly-viscous like liquid water, then it would be a "typical" fluid. We would use the Navier-Stokes equations as the model:

$$\nabla \cdot \mathbf{u} = 0 \qquad \textit{incompressibility} \tag{1}$$

$$\rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \nabla\cdot(\nu\nabla\mathbf{u}) + \rho\mathbf{g} \qquad \textit{stress balance} \tag{2}$$

In Equation (2) the term $\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}$ is an acceleration. The right-hand side is the total stress, and the equation says "$ma = F$", i.e. Newton's second law.

Numerical ice flow modelling is a topic in computational fluid dynamics. Ice sheets are a large scale fluid problem like atmosphere and ocean circulation in climate systems. But ice flow is an odd one! Consider some topics which might make ocean circulation modelling exciting, for example:

turbulence convection coriolis force stratification

None of these topics are relevant to ice flow. What could be interesting about the flow of slow, laminar, boring old ice?

We first observe that ice is a slow, shear-thinning fluid. In terms of equations, "slow" means $\rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) \approx 0$, which says that the forces of inertia are negligible. "Shear-

thinning" flows have specific non-Newtonian behavior in which larger strain rate implies smaller viscosity. The viscosity ν in (2) is not constant so we add an empirical flow law.

Thus the standard model for isothermal flow is this set of Stokes equations:

$$\nabla \cdot \mathbf{u} = 0 \qquad \textit{incompressibility} \tag{3}$$

$$0 = -\nabla p + \nabla \cdot \tau_{ij} + \rho \mathbf{g} \qquad \textit{stress balance} \tag{4}$$

$$Du_{ij} = A\tau^2 \tau_{ij} \qquad \textit{n=3 Glen flow law} \tag{5}$$

In the flow law, the deviatoric stress tensor τ_{ij} and the strain rate tensor Du_{ij} appear; previous lectures cover these. Here we merely note that each tensor in (5) has trace zero and that $2\,Du_{ij} = (u_i)_{x_j} + (u_j)_{x_i}$ (denoting $x_1, x_2, x_3 = x, y, z$).

The Stokes equations do not contain a time derivative. Therefore geometry, boundary stresses, and ice softness together determine the velocity and stress fields (i.e. $\mathbf{u}$, p, τ_{ij}) instantaneously. Indeed, ice flow simulation codes have no memory of momentum or velocity, and velocity is a "diagnostic" output which is not needed for restarting a simulation.

Slab-on-a-slope

Consider now the x, z-plane flow case of equations (3), (4), and (5). "Plane flow" means that velocity component v is zero and that all derivatives with respect to y are zero:

$$u_x + w_z = 0 \qquad \textit{incompressibility} \tag{6}$$

$$p_x = \tau_{11,x} + \tau_{13,z} \qquad \textit{stress balance } (x) \tag{7}$$

$$p_z = \tau_{13,x} - \tau_{11,z} - \rho g \qquad \textit{stress balance } (z) \tag{8}$$

$$u_x = A\tau^2 \tau_{11} \qquad \textit{flow law} \text{ (diagonal)} \tag{9}$$

$$u_z + w_x = 2A\tau^2 \tau_{13} \qquad \textit{flow law} \text{ (off-diagonal)} \tag{10}$$

Note that τ_{13} is a "vertical" shear stress while τ_{11} and $\tau_{33} = -\tau_{11}$ are deviatoric longitudinal stresses. Also $\tau = (\tau_{11}^2 + \tau_{13}^2)^{1/2}$ in this case. This is a system of five nonlinear equations in five scalar unknowns $(u, w, p, \tau_{11}, \tau_{13})$.

This is complicated enough to make us pause before jumping in to numerical solution methods. Can we handle a simplified situation? Our first example, a uniform slab or a "slab-on-a-slope", is both a case in which we actually solve the Stokes equations, and a motivation for the shallow model in the next subsection.

We rotate our coordinates only for this example and not elsewhere in these notes. The two-dimensional axes (x,z) shown in Figure 1 are rotated downward at angle $\alpha > 0$ so that the gravity vector has components $\mathbf{g} = (g \sin\alpha, -g\cos\alpha)$. Equations (7), (8), in these rotated coordinates, are

$$p_x = \tau_{11,x} + \tau_{13,z} + \rho g \sin\alpha, \tag{11}$$

$$p_z = \tau_{13,x} - \tau_{11,z} - \rho g \cos\alpha. \tag{12}$$

By now assuming additionally that there is no variation with x, the whole set of Stokes equations (6), (9), (10), (11), (12) simplifies greatly:

$$w_z = 0 \qquad\qquad 0 = \tau_{11}$$

$$\tau_{13,z} = -\rho g \sin\alpha \qquad\qquad u_z = 2A\tau^2\tau_{13}$$

$$p_z = -\rho g \cos\alpha$$

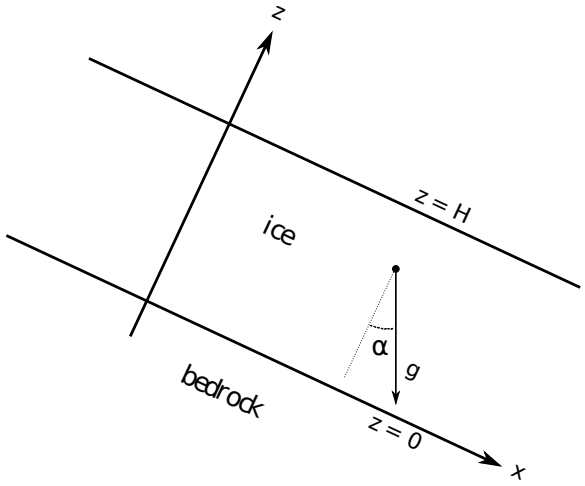


Figure 1: Rotated axes for a slab-on-a-slope flow calculation.

We apply boundary conditions for these functions of z:

$$w(0) = 0, \qquad p(H) = 0, \qquad u(0) = u_0.$$

The basal velocity u_0 will remain undetermined for now. By integrating vertically, we get $w = 0$, $p = \rho g \cos\alpha (H - z)$, and $\tau_{13} = \rho g \sin\alpha (H - z)$. Note that $H - z$ is the depth within the ice and that both the pressure p and shear stress τ_{13} are linear in depth.

By integrating vertically again we find a formula for the horizontal velocity,

$$\begin{aligned} u &= u_0 + 2A(\rho g \sin\alpha)^3 \int_0^z (H - z')^3 \, dz' \\ &= u_0 + \frac{1}{2} A(\rho g \sin\alpha)^3 \left(H^4 - (H - z)^4 \right) \end{aligned} \tag{13}$$

Do we believe formula (13)? Figure 2 compares to observations of a mountain glacier. This comparison shows we have done a credible job of capturing deformation flow velocity, though we do not yet have a model for the sliding velocity u_0 (i.e. basal motion).

Now that we have calculated the velocity u, so what? The equations so far do not address the change in shape of the glacier or ice sheet. For this we need another equation, the *mass continuity equation*. To explain it, define the vertical average of velocity:

$$\bar{u} = \frac{1}{H} \int_0^H u \, dz.$$

The flux $q = \bar{u}\, H$ is the rate of flow input into a side of the area in Figure 3. Now, how does the ice area A in the figure change?[1] We add the contributions around the edges,

$$\frac{dA}{dt} = \int_{x_1}^{x_2} M(x) \, dx + \bar{u}_1 H_1 - \bar{u}_2 H_2$$

If the width $\Delta x = x_2 - x_1$ is small then $A \approx \Delta x\, H$. So we divide by Δx, take $\Delta x \to 0$, and get

$$H_t = M - (\bar{u} H)_x \tag{14}$$

[1] In three-dimensions it would be "how does the ice volume change?"

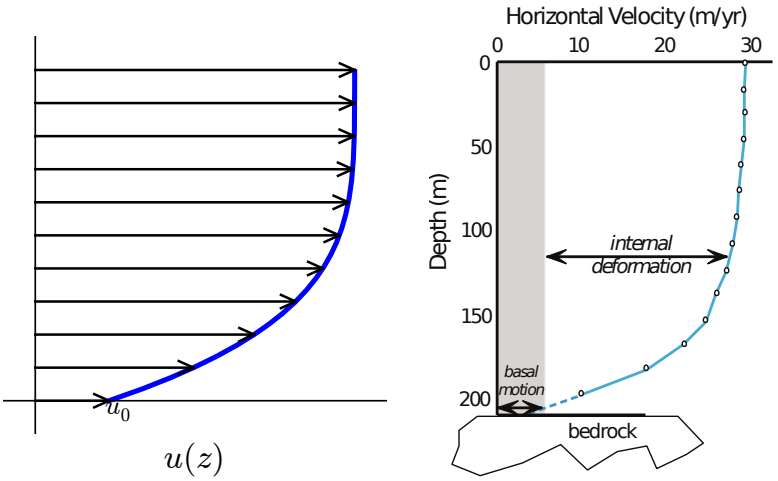


Figure 2: Left: Velocity from slab-on-a-slope formula (13). Right: Velocity in a mountain glacier (Athabasca Glacier, Canada), from inclinometry [**?**].

This mass continuity equation describes the change in the ice thickness in terms of surface mass balance and ice velocity. This is the most common "use" of the velocity in ice flow simulations.

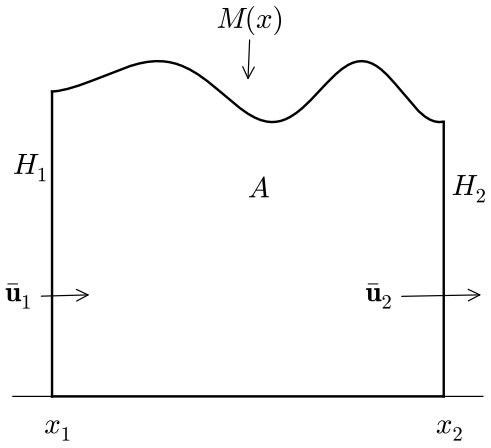


Figure 3: Mass continuity equation (14) follows from considering the changing area A of ice in a planar flow. Ice can be added at the top by mass balance M, but a difference of flux $q = \bar{u}H$ into left and right sides also changes A.

1.3 Shallow ice sheets

Our slab-on-a-slope example gives us a rough explanation of the shallow ice approximation (SIA), which is a model we will apply more generally. To complete it, vertically integrating velocity formula (13) in the $u_0 = 0$ (non-sliding) case gives

$$\bar{u}H = \int_0^H \frac{1}{2}A(\rho g \sin\alpha)^3 \left(H^4 - (H-z)^4\right)\, dz = \frac{2}{5}A(\rho g \sin\alpha)^3 H^5. \tag{15}$$

Note $\sin\alpha \approx \tan\alpha = -h_x$. Combining these statements with mass continuity (14) gives

$$H_t = M + \left(\frac{2}{5}(\rho g)^3 A H^5 |h_x|^2 h_x\right)_x. \tag{16}$$

Equation (16) is the SIA equation for nonsliding plane flow. It determines the evolution of an ice sheet's shape in terms of mass balance, ice softness, and bed elevation. Additional arguments (not shown) are needed to demonstrate that the SIA is "general purpose" and not special to a simple slab. Such arguments reduce the Stokes equations under the assumption that the surface and bed slopes, and the depth-to-width ratio, are small; see Notes and References. We will numerically solve (16) in section 1.5. First we state it in two horizontal dimensions and attempt to understand it better.

Understanding the shallow ice approx (SIA)

Ice sheets have four outstanding properties as fluids. They are (*i*) slow, (*ii*) shallow, (*iii*) non-Newtonian, and (*iv*) they experience some contact slip (basal sliding). The first ice flow model we consider, the non-sliding, isothermal SIA, accounts for (*i*)–(*iii*).

Regarding property (*ii*), Figure 4 shows a no-vertical-exaggeration cross-section of Greenland at 71°, as well as the standard vertically-exaggerated version which is more familiar in the glaciological literature. Thus: Remember that ice sheets are shallow.

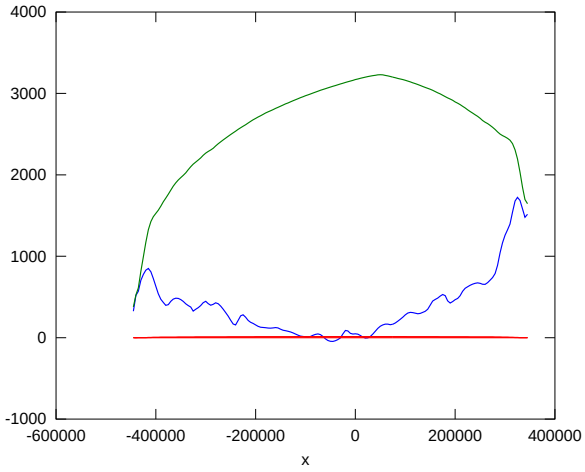


Figure 4: A vertically-exaggerated cross-section of the Greenland ice sheet (71° N) is shown by the green and blue curves. Without exaggeration it appears as a horizontal line (red).

The SIA uses the formulas from slab-on-a-slope, generalized to two horizontal dimensions. Let $\mathbf{U} = (u, v)$ be the vector horizontal velocity. The shear stress approximation is $(\tau_{13}, \tau_{23}) \approx -\rho g(h-z)\nabla h$, which becomes an equality in the SIA. From Equation (5), a further approximation of shear strain rates gives the SIA formula

$$\mathbf{U}_z = 2A|(\tau_{13}, \tau_{23})|^{n-1}(\tau_{13}, \tau_{23}) = -2A(\rho g)^n (h-z)^n |\nabla h|^{n-1}\nabla h.$$

By integrating vertically we get, in the non-sliding case,

$$\mathbf{U} = -\frac{2A(\rho g)^n}{n+1}\left[H^{n+1} - (h-z)^{n+1}\right]|\nabla h|^{n-1}\nabla h. \tag{17}$$

Mass continuity in two horizontal dimensions, which generalizes the 1D version (14), also applies:

$$H_t = M - \nabla \cdot \left(\bar{\mathbf{U}} H\right) \tag{18}$$

Equation (18) may be written $H_t = M - \nabla \cdot \mathbf{q}$ in terms of the map-plane flux $\mathbf{q} = \int_b^h \mathbf{U}\, dz = \bar{\mathbf{U}}\, H$.

Define the positive constant $\Gamma = 2A(\rho g)^n/(n+2)$. Combining Equations (17) and (18), we get an equation for the rate of thickness change in terms of mass balance M, thickness, and surface slope ∇h:

$$H_t = M + \nabla \cdot \left(\Gamma H^{n+2} |\nabla h|^{n-1} \nabla h\right) \tag{19}$$

Fulfilling an earlier promise, if we can solve (19) numerically then we have a usable model for the Barnes ice cap in Canada [**?**]. The Barnes ice cap is a particularly simple ice sheet on a rather flat bed.

Analogy with the heat equation

A step toward understanding the SIA model is to recognize its analogy with the better-known heat equation. All numerical methods for solving (19) can be understood as modifications of well-known heat equation methods.

The temperature $T(t,x)$ of a conducting rod satisfies the one-dimensional (1D) heat equation

$$T_t = DT_{xx}. \tag{20}$$

This form applies when material properties are constant. The constant D is the "diffusivity". This equation has a smoothing effect on the solution T as it evolves in time, because any local maximum in the temperature is flattened (i.e. $T_{xx} < 0$ implies T decreases), while any local minimum is also flattened (i.e. $T_{xx} > 0$ implies T increases).

The 2D heat equation, which we will see is analogous to equation (19), describes the temperature $T(t,x,y)$ at position x, y in a planar object at time t. Recall Fourier's law is the formula $\mathbf{Q} = -k\nabla T$ for heat flux. If we allow an additional heat source f, conservation of internal energy says $\rho c T_t = f - \nabla \cdot \mathbf{Q}$. Combining these we get the 2D heat equation:

$$\rho c T_t = f + \nabla \cdot (k \nabla T).$$

Here ρ is density, c is specific heat, and k is conductivity. Assuming ρc is constant, define the "diffusivity" $D = k/(\rho c)$ and source term $F = f/(\rho c)$; generally these may vary. The revised 2D heat equation is

$$T_t = F + \nabla \cdot (D\, \nabla T). \tag{21}$$

Figure 5 shows a solution of the heat equation, where the initial condition is a localized "hot spot". Solutions of the heat equation involve the spreading, in all directions, of any local heat maxima or minima, that is, diffusion.

The SIA equation (19) and the heat equation (21) are each diffusive, time-evolving partial differential equations (PDEs). A side-by-side comparison is illuminating:

SIA: H is ice thickness	heat: T is temperature
$\boxed{H_t = M + \nabla \cdot \left(\Gamma H^{n+2} \lvert\nabla h\rvert^{n-1} \nabla h\right)}$	$\boxed{T_t = F + \nabla \cdot (D\, \nabla T)}$

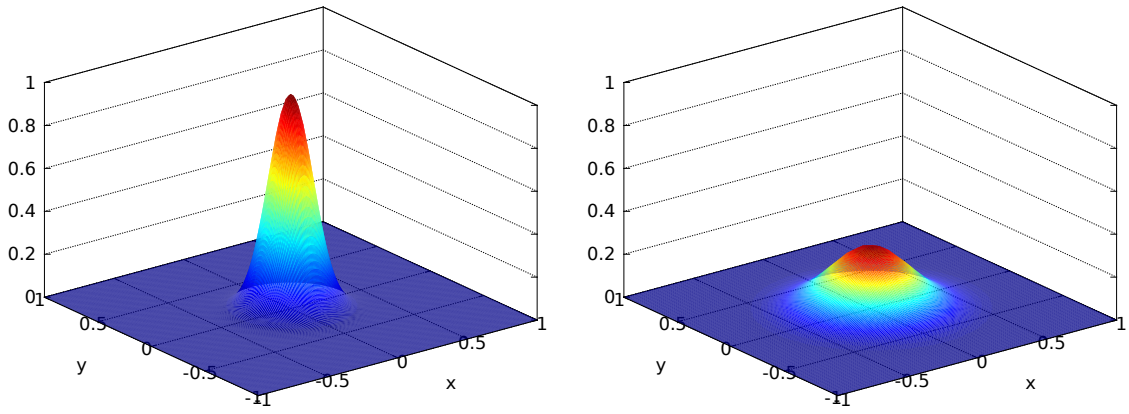


Figure 5: A solution of heat equation (21) with $D = 1$ and $F = 0$. Left: Initial condition $T(0, x, y)$. Right: Solution $T(t, x, y)$ at $t = 0.02$.

Notice that the number of derivatives (one time and two space derivatives) and the signs are the same. Mass balance M is analogous to heat source F.

The analogy suggests that we identify the diffusivity in the SIA:

$$D = \Gamma H^{n+2} |\nabla h|^{n-1}.$$

The non-sliding shallow ice flow *diffuses* the ice sheet. When Γ or H or $|\nabla h|$ are large, the diffusion acts more quickly because D is larger. This analogy explains generally why the surfaces of ice sheets are smooth, at least if we ignore non-fluid processes like crevassing and wind-driven snow. There are, however, some issues with this analogy:

- The diffusivity D depends on the solution thickness H and surface elevation h, and also these are not the same functions.
- The diffusivity D goes to zero at margins, where $H \to 0$, and at divides and domes, where $|\nabla h| \to 0$.

Nonetheless we will provide a verified numerical scheme for (19) in Section 1.5.

1.4 Finite difference numerics

As already noted, numerical schemes for the heat equation are a good starting place for solving the SIA equation (19) numerically. Finite difference numerical schemes, in particular, replace derivatives in a differential equation by arithmetic. The basic fact is *Taylor's theorem*, which says that for a smooth function $f(x)$,

$$f(x + \Delta) = f(x) + f'(x)\Delta + \frac{1}{2} f''(x)\Delta^2 + \frac{1}{3!} f'''(x)\Delta^3 + \dots$$

You can replace "Δ" by its multiples, for example:

$$f(x - \Delta) = f(x) - f'(x)\Delta + \frac{1}{2} f''(x)\Delta^2 - \frac{1}{3!} f'''(x)\Delta^3 + \dots$$

$$f(x + 2\Delta) = f(x) + 2f'(x)\Delta + 2f''(x)\Delta^2 + \frac{4}{3} f'''(x)\Delta^3 + \dots$$

The idea for constructing finite difference schemes is to combine expressions like these to give approximations of derivatives. Thereby function values on a grid combine to approximate the differential equation.

Here we want partial derivative approximations, so we apply the Taylor's expansions one variable at a time. For example with a general function $u = u(t, x)$,

$$\begin{aligned}
u_t(t,x) &= \frac{u(t+\Delta t, x) - u(t,x)}{\Delta t} + O(\Delta t), \\
u_t(t,x) &= \frac{u(t+\Delta t, x) - u(t-\Delta t, x)}{2\Delta t} + O((\Delta t)^2), \\
u_x(t,x) &= \frac{u(t, x+\Delta x) - u(t, x-\Delta x)}{2\Delta x} + O((\Delta x)^2), \\
u_{xx}(t,x) &= \frac{u(t, x+\Delta x) - 2\,u(t,x) + u(t, x-\Delta x)}{\Delta x^2} + O((\Delta x)^2)
\end{aligned}$$

Note that if Δ is a small number then "$+O(\Delta^2)$" is better, in the sense that the approximation is closer, than "$+O(\Delta)$"

Explicit scheme for the heat equation

Now we can state the simplest "explicit" scheme which approximates the 1D heat equation (20):

$$\frac{T(t+\Delta t, x) - T(t,x)}{\Delta t} \approx D\,\frac{T(t, x+\Delta x) - 2\,T(t,x) + T(t, x-\Delta x)}{\Delta x^2}.$$

The finite difference scheme is not just an approximation of the PDE, but an actual method for computing numbers on a grid. Let (t_n, x_j) denote the time-space grid points. Denote our approximation[2] of the solution value $T(t_n, x_j)$ by T_j^n. Then the finite difference scheme is

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = D\,\frac{T_{j+1}^n - 2\,T_j^n + T_{j-1}^n}{\Delta x^2}.$$

To get a computable formula, let $\mu = D\Delta t/(\Delta x)^2$ and solve for T_j^{n+1}:

$$T_j^{n+1} = \mu T_{j+1}^n + (1-2\mu)T_j^n + \mu T_{j-1}^n \tag{22}$$

The scheme is "explicit" because we have been able to solve it for T_j^{n+1} in terms of values at time t_n. Figure 6 shows the "stencil" for scheme (22): three values at the current time t_n are combined to update the one value at the next time t_{n+1}.

How accurate is scheme (22)? Its construction tells us that the difference between the scheme (22) and the PDE (20) is $O(\Delta t, (\Delta x)^2)$. This difference between scheme and PDE goes to zero as we refine the grid in space and time ("consistency"). With care about the smoothness of boundary conditions, and using mathematical facts about the heat equation itself, one can show that the difference between T_j^n and $T(t_n, x_j)$ is *also* $O(\Delta t, (\Delta x)^2)$, which is "convergence"; see Notes and References. However, to get convergence, the PDE problem must be known to generate smooth solutions and the scheme

[2]Evaluating the exact solution at a grid point would give a different value from the one we compute by the scheme! Of course we plan that these numbers will be close, but that needs checking or proof.

(22) must be stable, which we address below. Then the main theorem for numerical PDE schemes is "consistency plus stability implies convergence"; see Notes and References. In these notes we do something rather practical, namely we find problems for which we already know exact values $T(t,x)$, and we compute the differences $|T_j^n - T(t_n, x_j)|$. Thus we determine directly whether they go to zero. This is "verification"; more below.

A first implemented scheme

For the first Matlab implementation we consider the two spatial dimension Equation (21) with D constant and $F = 0$:

$$T_t = D(T_{xx} + T_{yy}). \tag{23}$$

Writing $T_{jk}^n \approx T(t_n, x_j, y_k)$, the 2D explicit scheme is

$$\frac{T_{jk}^{n+1} - T_{jk}^n}{\Delta t} = D\left(\frac{T_{j+1,k}^n - 2\,T_{jk}^n + T_{j-1,k}^n}{\Delta x^2} + \frac{T_{j,k+1}^n - 2\,T_{jk}^n + T_{j,k-1}^n}{\Delta y^2}\right). \tag{24}$$

The stencil for the right-hand side of (24) is in Figure 6.

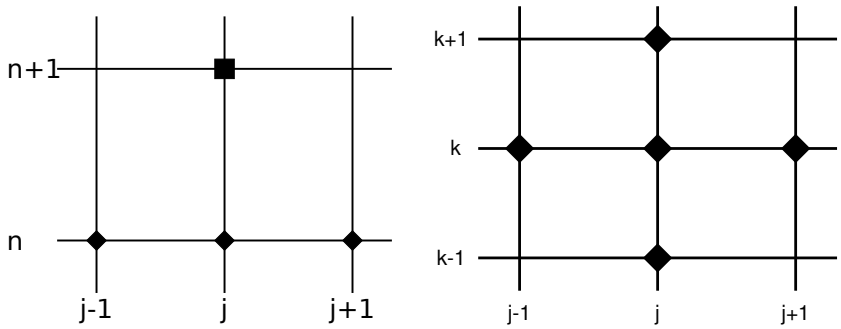


Figure 6: Left: Space-time stencil for the explicit scheme (22) for the 1D heat equation. Right: Two spatial dimension stencil for scheme (24).

Scheme (24) has implementation `heat.m` below. We use $T = 0$ values on the boundary of the square $-1 < x < 1$, $-1 < y < 1$. The initial condition is gaussian: $T(0,x,y) = \exp(-30(x^2+y^2))$. The code uses Matlab "colon" notation to remove loops over spatial variables. Here is an example run:

```
>>  heat(1.0,30,30,0.001,20);
```

This sets $D = 1.0$ and uses a 30×30 spatial grid. We take $N = 20$ time steps of $\Delta t = 0.001$. The result is shown in Figure 5, right. This is the look of success.

heat.m

```
function T = heat(D,J,K,dt,N)

dx = 2 / J;     dy = 2 / K;
[x,y] = meshgrid(-1:dx:1, -1:dy:1);
T = exp(-30*(x.*x + y.*y));

mu_x = dt * D / (dx*dx);
```

```
mu_y = dt * D / (dy*dy);
for n=1:N
   T(2:J,2:K) = T(2:J,2:K) + ...
       mu_x * ( T(3:J+1,2:K) - 2 * T(2:J,2:K) + T(1:J-1,2:K) ) + ...
       mu_y * ( T(2:J,3:K+1) - 2 * T(2:J,2:K) + T(2:J,1:K-1) );
end

surf(x,y,T),  shading('interp'),  xlabel x,  ylabel y
```

Very similar runs seem to fail, however. Results from these calls are in Figure 7:

```
>> heat(1.0,40,40,0.0005,100);     % Figure 7, left
>> heat(1.0,40,40,0.001,50);       % Figure 7, right
```

Both runs compute temperature T on the same spatial grid, at the same final time, but they have different time steps. The second run clearly shows instability.

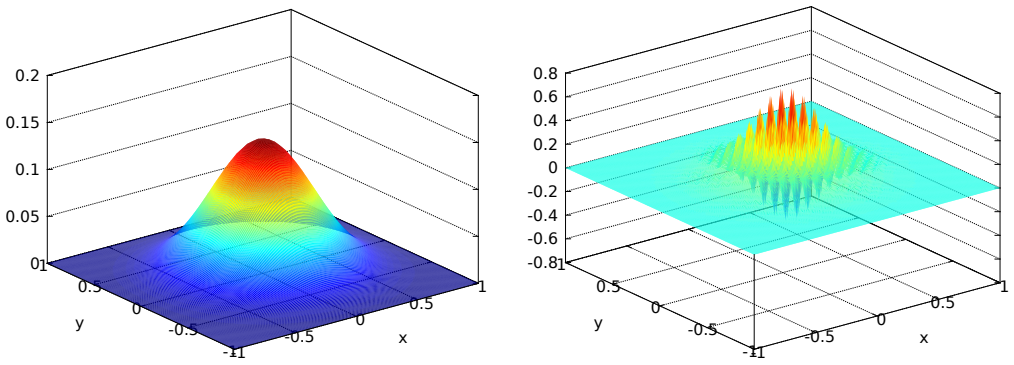


Figure 7: Numerically-computed temperature at time $t_f = 0.05$ on 40×40 grids. Everything in the two runs is the same, except that the left figure has $\Delta t = 0.0005$ so $D\Delta t/(\Delta x)^2 = 0.2$, while the right has $\Delta t = 0.001$ so $D\Delta t/(\Delta x)^2 = 0.4$.

Stability criteria and adaptive time stepping

How to avoid the instability? We need to understand the scheme better. It turns out we have not made an implementation error, but we must be more careful with choice of space and time steps.

Recall the 1D explicit scheme in form (22): $T_j^{n+1} = \mu T_{j+1}^n + (1-2\mu)T_j^n + \mu T_{j-1}^n$. The new value T_j^{n+1} is an average of the old values, in the sense that the coefficients add to one. Actually, this is an average *if* the middle coefficient is positive. A linear combination with coefficients which add to one is not an average if any coefficients are negative! (For example, we would not accept 13 as an "average" of 5 and 7, but of course we can write $13 = -3 \times 5 + 4 \times 7$, and $-3 + 4 = 1$.) Averaging is stable because averaged wiggles are always smaller than the original wiggles. We want this property.

So, what would follow from requiring the middle coefficient in (22) to be positive?:

$$1 - 2\mu \geq 0 \quad \Longleftrightarrow \quad \frac{D\Delta t}{\Delta x^2} \leq \frac{1}{2} \quad \Longleftrightarrow \quad \Delta t \leq \frac{\Delta x^2}{2D}. \tag{25}$$

This condition on the size of Δt is a *sufficient* stability criterion. (It is enough to guarantee stability, though something weaker might do.) For given Δx, shortening the time step so that $\Delta t \le \Delta x^2/(2D)$ will make the scheme into an averaging process.

Applying this same idea to the 2D heat equation (23) leads to the stability condition that $1 - 2\mu^x - 2\mu^y \ge 0$ where $\mu^x = D\Delta t/(\Delta x^2)$ and $\mu^y = D\Delta t/(\Delta y^2)$. In the cases shown in Figure 7 with $\Delta x = \Delta y$, this condition requires $D\Delta t/(\Delta x^2) \le 0.25$, which precisely distinguishes between the two parts of the Figure. In summary, instability in runs of `heat.m` happens if the time step Δt is too big relative to the spacing Δx.

The stability criterion above is easily satisfied by making each time step shorter. Doing so in a program is an *adaptive* implementation. It has guaranteed stability. To show how easy it is to implement, `heatadapt.m` (not shown) is the same as `heat.m` except that the time step is chosen from the stability criterion. It cannot generate the instability seen in Figure 7. However, if the diffusivity D is large or the spatial steps Δx, Δy are small, then explicit, adaptive implementations like this one must take short time steps to assure stability.

Implicit schemes

There is an alternative stability fix instead of adaptivity, namely "implicitness." For example, the finite difference scheme

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = D\,\frac{T_{j+1}^{n+1} - 2\,T_j^{n+1} + T_{j-1}^{n+1}}{\Delta x^2} \tag{26}$$

is an $O(\Delta t, (\Delta x)^2)$ accurate implicit scheme for Equation (20). Such implicit schemes for the heat equation are stable for *any* positive time step $\Delta t > 0$ ("unconditionally stable"). Another well-known example implicit scheme is *Crank-Nicolson*, which is unconditionally stable for the heat equation and which also has yet smaller error $O((\Delta t)^2, (\Delta x)^2)$.

But implicit schemes are harder to implement. First, the unknown solution values at time step t_{n+1} are treated as a vector in a large system of equations which must be formed and solved at each time step. If the PDE is nonlinear—the SIA is a highly nonlinear diffusion equation—then the system of equations may be hard to solve. For these notes we stay with the adaptive explicit idea. Generally, there is a tradeoff between the implementability of adaptive explicit schemes and the stability of implicit schemes.

Numerical solution of diffusion equations

We are trying to numerically model ice flows, not heat conduction. We have an analogy, however, which says that the SIA is like the heat equation: both are diffusions. Heat equation (21) is called the "diffusion equation" if the unknown is not a temperature. In this section, because we wish to solve the SIA on real bedrock, we construct a numerical scheme for a generalized diffusion which additionally has a "shift" inside the gradient, namely

$$T_t = F + \nabla \cdot (D\nabla(T + b))\,. \tag{27}$$

In equation (27), the source term $F(x, y)$, the diffusivity $D(x, y)$, and the "shift" $b(x, y)$ may all vary in space. We now provide a code that solves (27) and is used in SIA solutions.

An adaptive explicit method for the diffusion equation is conditionally stable, with the same essential time step restriction as for the constant diffusivity case, as long as we evaluate $D(x,y)$ at *staggered* grid points. That is, we use this expression for the second derivative:

$$\nabla\cdot(D\nabla X) \approx \frac{D_{j+1/2,k}(X_{j+1,k}-X_{j,k}) - D_{j-1/2,k}(X_{j,k}-X_{j-1,k})}{\Delta x^2} + \frac{D_{j,k+1/2}(X_{j,k+1}-X_{j,k}) - D_{j,k-1/2}(X_{j,k}-X_{j,k-1})}{\Delta y^2},$$

where $X = T + b$. Figure 8 shows the stencil.

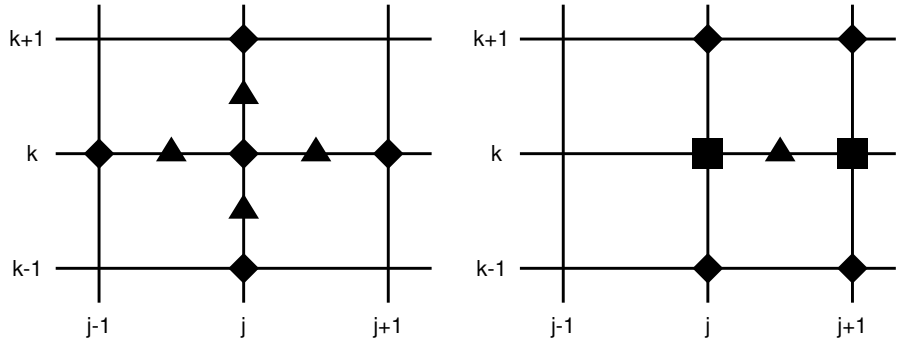


Figure 8: Left: Spatial stencil for staggered grid evaluation of diffusivity (at triangles) in the diffusion equation (27). Right: Stencil showing how the staggered-grid diffusivity (triangle) can be evaluated in the SIA, from surface elevation (diamonds) and thicknesses (squares).

Code `diffusion.m` (below) solves (27) numerically. The user supplies the diffusivity $D(x,y)$ on the staggered grid. Initial temperature $T(0,x,y)$, source term $F(x,y)$, and "shift" $b(x,y)$ are supplied on the regular grid. When using this code for standard diffusions, or the flat-bed SIA, we would take $b = 0$.

diffusion.m

```
function [T,dtav] = diffusion(Lx,Ly,J,K,Dup,Ddown,Dright,Dleft,T0,tf,F,b)

dx = 2 * Lx / J;    dy = 2 * Ly / K;
[x,y] = ndgrid(-Lx:dx:Lx, -Ly:dy:Ly);
T = T0;
if nargin < 11, F = zeros(size(T0)); end
if nargin < 12, b = zeros(size(T0)); end

t = 0.0;    count = 0;
while t < tf
   maxD = [max(max(Dup)) max(max(Ddown)) max(max(Dleft)) max(max(Dright))];
   maxD = max(maxD);
   if maxD <= 0.0
     dt = tf - t;
   else
     dt0 = 0.25 * min(dx,dy)^2 / maxD;
     dt = min(dt0, tf - t);
   end
   mu_x = dt / (dx*dx);    mu_y = dt / (dy*dy);
   Tb = T + b;
   T(2:J,2:K) = T(2:J,2:K) + ...
```

```
        mu_y * Dup    .* ( Tb(2:J,3:K+1) - Tb(2:J,2:K)   ) - ...
        mu_y * Ddown  .* ( Tb(2:J,2:K)   - Tb(2:J,1:K-1) ) + ...
        mu_x * Dright .* ( Tb(3:J+1,2:K) - Tb(2:J,2:K)   ) - ...
        mu_x * Dleft  .* ( Tb(2:J,2:K)   - Tb(1:J-1,2:K) );
    T = T + F * dt;
    t = t + dt;    count = count + 1;
end
dtav = tf / count;
```

1.5 Numerically solving the SIA

Recall that in SIA equation (19) we have diffusivity $D = \Gamma H^{n+2}|\nabla h|^{n-1}$. This diffusivity "degenerates" $D \to 0$ when either $H \to 0$ or $\nabla h \to 0$. Degenerate diffusion equations are automatically free boundary problems, but this is no surprise to a glaciologist. Determining the location of the margin, and thus the spatial extent of the ice flow, is a major part of the problem. To address this free boundary issue in our explicit time-stepping code it suffices to numerically compute new thicknesses and then set them to zero if they come out negative.

For numerical stability we compute the SIA diffusivity D on the staggered grid. Various finite difference schemes for computing it have been proposed but all involve averaging H and differencing h in a "balanced" way onto the staggered grid. In the code `siaflat.m` below we use the Mahaffy choice, with the diffusivity-computation stencil shown in Figure 8. This code only works for the flat bed, zero surface mass balance case, but we will correct these deficiencies later.

siaflat.m

```
function [H,dtlist] = siaflat(Lx,Ly,J,K,H0,deltat,tf)

g = 9.81;    rho = 910.0;    secpera = 31556926;
A = 1.0e-16/secpera;    Gamma  = 2 * A * (rho * g)^3 / 5;
H = H0;

dx = 2 * Lx / J;   dy = 2 * Ly / K;
N = ceil(tf / deltat);    deltat = tf / N;
j  = 2:J;    k = 2:K;
nk = 3:K+1;    sk = 1:K-1;    ej = 3:J+1;    wj = 1:J-1;

t = 0;   dtlist = [];
for n=1:N
  Hup = 0.5 * ( H(j,nk) + H(j,k) );   Hdn = 0.5 * ( H(j,k) + H(j,sk) );
  Hrt = 0.5 * ( H(ej,k) + H(j,k) );   Hlt = 0.5 * ( H(j,k) + H(wj,k) );
  a2up = (H(ej,nk) + H(ej,k) - H(wj,nk) - H(wj,k)).^2 / (4*dx)^2 + ...
         (H(j,nk) - H(j,k)).^2 / dy^2;
  a2dn = (H(ej,k) + H(ej,sk) - H(wj,k) - H(wj,sk)).^2 / (4*dx)^2 + ...
         (H(j,k) - H(j,sk)).^2 / dy^2;
  a2rt = (H(ej,k) - H(j,k)).^2 / dx^2 + ...
         (H(ej,nk) + H(j,nk) - H(ej,sk) - H(j,sk)).^2 / (4*dy)^2;
  a2lt = (H(j,k) - H(wj,k)).^2 / dx^2 + ...
         (H(wj,nk) + H(j,nk) - H(wj,sk) - H(j,sk)).^2 / (4*dy)^2;
  Dup = Gamma * Hup.^5 .* a2up;  Ddn = Gamma * Hdn.^5 .* a2dn;
  Drt = Gamma * Hrt.^5 .* a2rt;  Dlt = Gamma * Hlt.^5 .* a2lt;
  [H,dtadapt] = diffusion(Lx,Ly,J,K,Dup,Ddn,Drt,Dlt,H,deltat);
  t = t + deltat;    dtlist = [dtlist dtadapt];
end
```

1.6 Exact solutions and verification

In `siaflat.m`, which calls `diffusion.m`, we already have a fairly complicated code. How do we make sure that such an *implemented* numerical scheme is correct? Here are three proposed techniques:

1. don't make any mistakes, or
2. compare your numerical results with several others, and hope that the outliers are in error, or
3. compare your numerical results to an exact solution.

The last one of these, which we prefer to the first two when it is possible, is called "verification." Of course, when we build a new computer code we should test it in cases where we know the right answer; that is the idea of verification. To do so we need to return to the PDE itself, to get useful exact solutions.

Exact solution of heat equation

First we consider the simpler case of the 1D heat equation with constant D, namely $T_t = DT_{xx}$. Many exact solutions $T(t,x)$ to this heat equation are known, but let's consider the time-dependent "Green's function," also known as the "heat kernel". It starts at time $t = 0$ with a "delta function" $T(0,x) = \delta(x)$ of heat at the origin $x = 0$. Then it spreads out over time. It is a solution of the heat equation on the whole line $-\infty < x < \infty$ and for all $t > 0$. We can calculate this exact solution by a method which generalizes to the SIA.

The Green's function of the heat equation is "self-similar" over time, in the sense that it changes shape *only* by shrinking the output (vertical) axis and lengthening the input (horizontal) axis, as shown in Figure 9. These scalings are related to each other by the conservation of energy, which says that the total heat energy is independent of time.

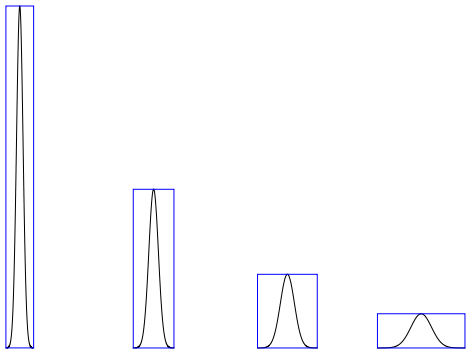


Figure 9: The Green's function of the heat equation in 1D has the same "shape" at each time, but there are time-dependent input- and output-scalings.

In particular, the Green's function of the 1D heat equation is

$$T(t,x) = (4\pi Dt)^{-1/2}\, e^{-x^2/(4Dt)}.$$

"Similarity" variables for this solution, the above-mentioned scalings, are

$$\underset{\text{input scaling}}{s} = t^{-1/2}x, \qquad \underset{\text{output scaling}}{T(t,x)} = t^{-1/2}\phi(s).$$

The invariant shape function is $\phi(s) = (4\pi D)^{-1/2}\, e^{-s^2/(4D)}$. All time-dependence is from the input and output scalings.

For a sufficiently-long spatial interval, so that zero boundary conditions cause no problem, a numerical solver for the 1D heat equation should produce numbers which are close to this exact solution; see the Exercises.

Halfar's exact "similarity" solution to the SIA

Now we jump forward from Green's idea in about 1830 to the year 1981 in which P. Halfar published the similarity solution of the SIA in the case of flat bed and zero surface mass balance. Halfar's solution to the 2D SIA model (19), using a Glen exponent $n = 3$, has scalings

$$s = t^{-1/18}r, \qquad H(t,r) = t^{-1/9}\phi(s) \tag{28}$$

where $r = (x^2 + y^2)^{1/2}$ is the distance from the origin. The scalings in (28) imply that, quite differently from heat, the diffusion of ice slows down severely as the shape flattens out. That is, the powers $t^{-1/9}$, $t^{-1/18}$ imply very little change for large times t.

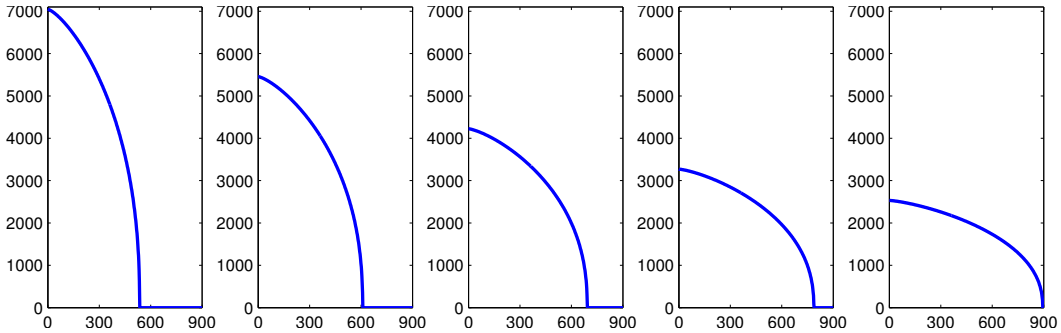


Figure 10: Halfar's solution $H(t,r)$ (29) of the SIA equation (19) on a flat bed, and with zero mass balance. The solution is shown on H (m) versus r (km) axes for times $t = 1, 10, 100, 1000, 10000$ years, in a case where $t_0 = 422$ years.

The formula for the Halfar solution to the SIA is remarkably simple,

$$H(t,r) = H_0 \left(\frac{t_0}{t}\right)^{1/9} \left[1 - \left(\left(\frac{t_0}{t}\right)^{1/18} \frac{r}{R_0}\right)^{4/3}\right]^{3/7} \tag{29}$$

where the "characteristic time" is $t_0 = (18\Gamma)^{-1}(7/4)^3 R_0^4 H_0^{-7}$; choosing center height H_0 and radius R_0 determines t_0. Time- and space-dependent formula (29) for thickness $H(t,r)$ is plotted in Figure 10. We see that for times significantly greater than t_0 (i.e. $t/t_0 \gg 1$) the solution changes very slowly. The change between years 1 and 100 is comparably large with that between years 1000 and 10000.

Using Halfar's solution

Formula (29) is simple enough to use for verifying time-dependent SIA models. The code `verifysia.m` takes as input the number of grid points in each (x, y) direction. It uses the Halfar solution at 200 a as the initial condition, does a numerical run to time 20000 a, using `siaflat.m` above, and then compares to the Halfar formula for that time:

```
>> verifysia(20)
average thickness error     = 22.310
>> verifysia(40)
average thickness error     = 9.490
>> verifysia(80)
average thickness error     = 2.800
>> verifysia(160)
average thickness error     = 1.059
```

We see that the thickness error, namely the absolute value of the difference between the numerical and exact thickness solutions at $t = 20000$ a, decreases with increasing grid resolution. This is as expected for a correctly-implemented code. What is less obvious, perhaps, is that almost any numerical implementation mistake—almost any bug—will break this property, and these errors will not shrink.

You might ask, is the Halfar solution also useful for modelling real ice masses? J. Nye and others (2000; [**?**]) compared the long-time consequences of different flow laws for the south polar cap on Mars. In particular, they evaluated CO_2 ice and H_2O ice softness parameters by comparing the long-time behavior of the corresponding Halfar solutions. Their conclusions:

> ... none of the three possible [CO_2] flow laws will allow a 3000-m cap, the thickness suggested by stereogrammetry, to survive for 10^7 years, indicating that the south polar ice cap is probably not composed of pure CO_2 ice [but rather] water ice, with an unknown admixture of dust.

This theoretical result has been confirmed by the observation and sampling of the polar geology of Mars.

Are exact solutions always available when needed? No. However, sometimes the absence of exact solutions means that not enough thought has gone into the continuum model itself. In any case, many ice flow models already have exact solutions which are relevant to verification. For example, we will use van der Veen's solution for ice shelves in a later section. Sources for additional exact solutions appear in Notes and References.

A test of robustness

Verification is a reasonable start to testing a code. Another kind of test is for "robustness": Does the model break when you ask it to do hard things? Unlike for verification, we might not have precise knowledge of what it should do, but we know the program should act in a "reasonable" way.

The robustness test in the program `roughice.m` (not shown) demonstrates that `siaflat.m` can handle an ice sheet with extraordinarily large driving stresses. Recall that glaciological "driving stress" is $\tau_d = -\rho g H \nabla h$. This quantity appears in the slab-on-a-slope example, and in the SIA model, as the value of the shear stress (τ_{13}, τ_{23}) at the base of

the ice. We give `siaflat.m` a randomly-generated initial ice sheet which is of the worst possible sort. It is both thick—mean thickness 3000 m—and it has large surface slopes. Such an initial sheet is shown in the left side of Figure 11. During the run of 50 model years, the time step is determined adaptively, increasing from 0.0002 years to close to the user-chosen maximum time step of 0.2 years. The maximum value of the driving stress decreases from 5.7×10^6 Pa to 3.6×10^5 Pa. At the end of the run the ice cap has the shape shown at right in Figure 11. The shape is rather close to a Halfar solution; indeed Halfar proved that all solutions of the SIA on a flat bed with no mass balance asymptotically approach the Halfar solution.

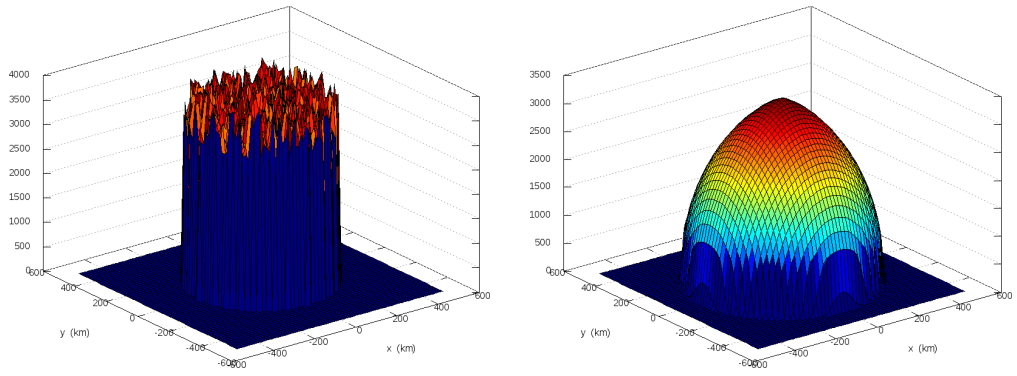


Figure 11: `siaflat.m` evolves the worst-possible initial ice sheet (left), an ice cap with huge driving stresses, to the ice cap at right after 50 model years.

1.7 Applying our numerical ice sheet model

Finally we apply the model to the Antarctic ice sheet. First we modify `siaflat.m` to allow non-flat bedrock elevation $b(x, y)$ and time-independent surface mass balance $M(x, y)$. Also we calve floating ice, and we enforce non-negative thickness at each timestep. The result is `siageneral.m` (not shown).

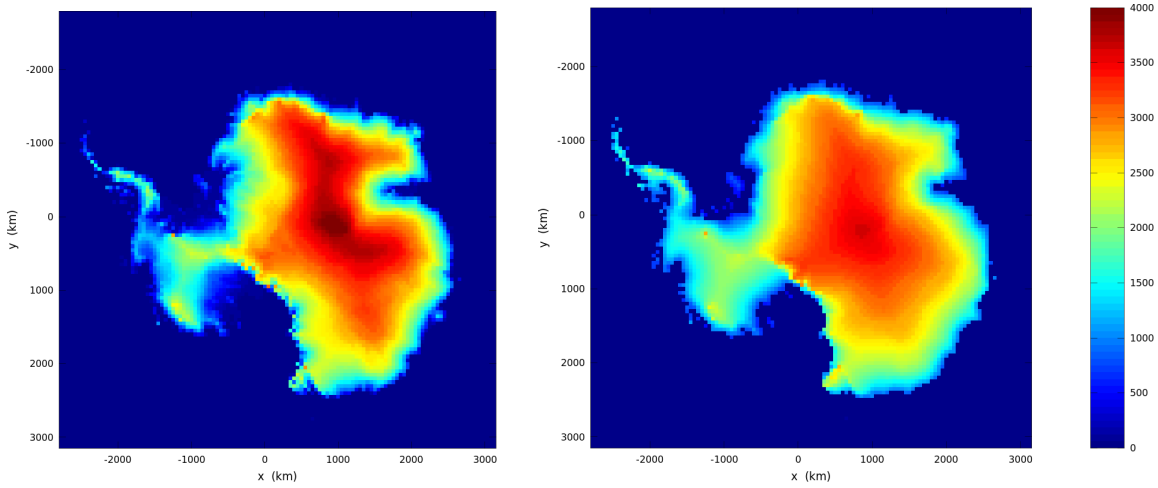


Figure 12: Left: Initial surface elevation (m) of Antarctic ice sheet. Right: Final surface elevation at end of 40 ka model run on 50 km grid.

We use measured accumulation, bedrock elevation, and surface elevation from ALBMAPv1 data [**?**]. Melt is not modelled so the surface mass balance is the accumulation rate. These input data are read from a NetCDF file by additional codes `buildant.m` and `netcdf.m` (not shown).

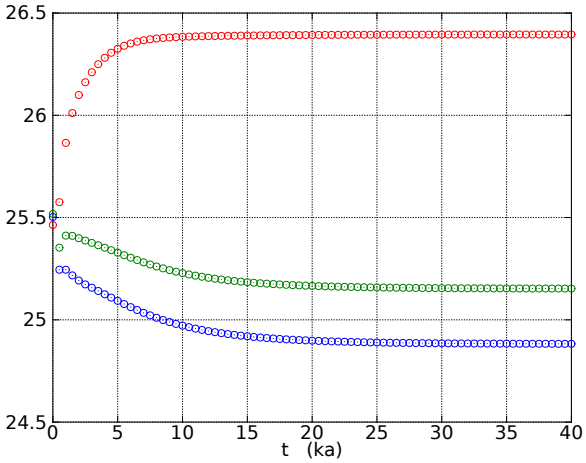


Figure 13: Ice volume time series of the modeled Antarctic ice sheet, in units of 10^6 km^3, from runs on 50 km (red), 25 km (green), and 20 km (blue) grids.

The code `ant.m` (not shown) calls `siageneral.m` to do the simulation in blocks of 500 model years. The volume is computed at the end of each block. Figure 12 shows the initial and final surface elevations from a run of 40,000 model years on a $\Delta x = \Delta y = 50$ km grid. The runtime on a typical laptop is a few minutes. Note that areas of low-slope and fast-flowing ice experience thickening, while near-divide ice in East Antarctica thins. Assuming the present-day Antarctic ice sheet is near steady state, these thickness differences reflect the lack of a sliding mechanism in the former case and the lack of thermomechanical coupling in the latter case.

Figure 13 compares the ice volume time series for 50 km, 25 km, and 20 km grids. This result, namely grid dependence of the ice volume, is typical. One cause is that most steep gradients near the ice margin are poorly resolved, but this is true on all of these coarse resolutions. Mainly this result is a warning about the interpretation of model runs on fixed grids. Even if the data is available only on a fixed grid, the model should be run at different resolutions to evaluate the robustness of the model results.

1.8 Mass continuity and kinematical equations

The previous SIA discussion combines two equations which we will now think about separately, namely the "stress balance" and mass continuity equations. In the SIA the stress balance is formula (17) for the velocity. It combines with the mass continuity equation (18) to give model (19) for the ice sheet thickness. In this section we consider the mass continuity equation more generally.

The basic shallow assumption made by most ice flow theories[3] is that the surface and base of the ice are differentiable functions $z = h(t, x, y)$ and $z = b(t, x, y)$. Thus surface overhang is not allowed. By contrast, the Stokes theory of slow viscous fluids only needs

[3]There are several inequivalent shallow theories: SIA, SSA, hybrids, Blatter-Pattyn, ...

a boundary surface for the fluid, i.e. a surface in three-dimensional space. Most ice sheet and glacier models take a map-plane perspective, however, and they have a well-defined ice thickness $H = h - b$.

We can already state an important mathematical fact which follows from this basic assumption. Namely that it implies that the surface kinematical and mass continuity equations are equivalent. More precisely, any pair of these equations implies the third:

- the surface kinematical equation (30) (below),
- the base kinematical equation (31) (below), and
- the map-plane mass continuity equation (18).

What are these kinematical equations? Let a be the upper surface mass balance function ($a > 0$ is accumulation) and s be the basal melt rate function ($s > 0$ is basal melting). The map-plane mass balance is $M = a - s$. The surface kinematical equation is

$$h_t = a - \mathbf{U}\big|_h \cdot \nabla h + w\big|_h, \tag{30}$$

and the base kinematical equation is

$$b_t = s - \mathbf{U}\big|_b \cdot \nabla b + w\big|_b. \tag{31}$$

These equations describe the movement of the ice upper surface and base from the velocity of the ice and the mass balance.

One proves the equivalences claim above by using the incompressibility of ice (1) and the Leibniz rule for differentiating integrals. The details are left for exercises.

The bedrock is often regarded as fixed (i.e. $b_t = 0$), and in fact the basal kinematical equation is often not explicitly mentioned in ice sheet modelling. Instead one gets a simplified view. In the case of no sliding, for example, the basal value of the vertical velocity equals the basal melt rate. This simplification corresponds to $b_t = 0$ and $\mathbf{U}\big|_b = 0$ so that $w\big|_b = -s$.

To conclude this short section we sketch a standard prognostic, isothermal ice sheet model. Like most ice sheet models, we use the mass continuity equation (18) to describe changes in ice sheet geometry, but we could use the surface kinematical equation instead. Each time step follows this recipe:

- numerically solve a stress balance, which gives velocity $\mathbf{u} = (u, v, w)$,
  - if the stress balance only gives $\mathbf{U} = (u, v)$, get w from incompressibility,
- decide on a time step Δt for (18) based on velocities and/or diffusivities,
- from the horizontal velocity $\mathbf{U} = (u, v)$, compute the flux $\mathbf{q} = \bar{\mathbf{U}}H$,
- update mass balance $M = a - s$ and do a time-step of (18) to get H_t,
- update the upper surface elevation and thickness (e.g. $h \mapsto h + H_t \Delta t$), and repeat.

The above "standard paradigm" has many variations, however. Some glaciological questions are answered just by solving the stress balance for the velocity. Sometimes the goal is the steady state configuration of the glacier, which can be computed more quickly than by time-stepping physical evolution equations to steady state. (I.e. there are faster iterative solvers of the coupled steady state equations.) And other processes are often simulated at each time step, such as the conservation of energy within the ice, or subglacial and supraglacial processes. Understanding the diverse time scales associated to these processes is key.

In the SIA simulations of previous sections we apparently did not compute the velocity. That is because the analogy to the heat equation allowed us to write the mass continuity equation as a diffusion, with $\mathbf{q} = -D\nabla h$ for the flux. Fast flow in ice sheets is associated with sliding and floating ice, however, and for these flows the ice geometry evolution is not a diffusion. One solves the stress balance for the velocity field. We consider such a stress balance associated to fast flow next.

1.9 Shelves and streams

The shallow shelf approximation (SSA) stress balance applies to ice shelves as its name suggests. It applies best to parts of ice shelves which are away from grounding lines and calving fronts. The SSA also applies reasonably well to ice streams, like those in Figure 14 which have not-too-steep bed topography and low basal resistance.

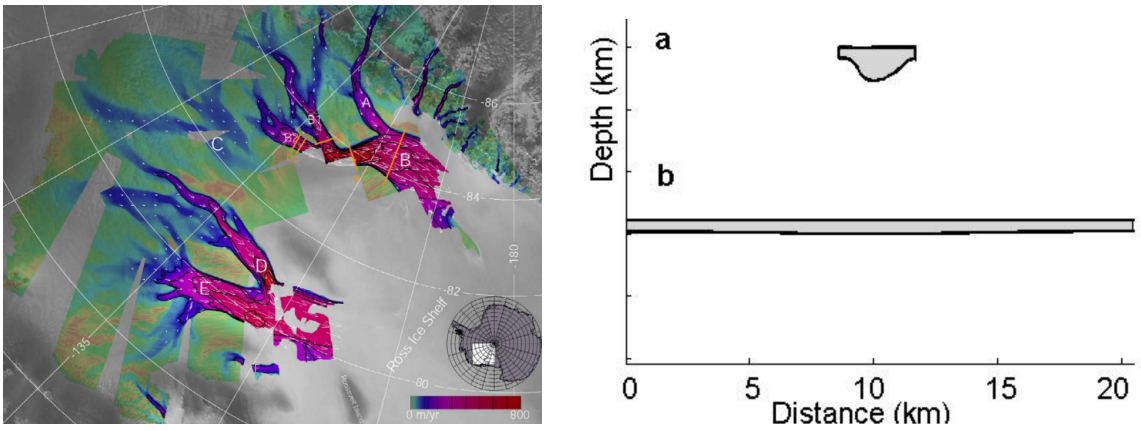


Figure 14: Left: The SSA model applies to ice streams like these on the Siple Coast in Antarctica. Color shows radar-derived surface speed. Right: Cross sections, *without* vertical exaggeration, of the Jakobshavns Isbrae outlet glacier in Greenland (**a**) and the Whillans Ice Stream on the Siple Coast (**b**); this is Figure 1 in [**?**].

But what is, and is not, an ice stream? Ice streams slide at 50 to 1000 m a^{-1}, they have a concentration of vertical shear in a thin layer near base, and typically they flow into ice shelves. Pressurized liquid water at their beds plays a critical role enabling their fast flow. There are other fast-flowing grounded parts of ice sheets, however, called "outlet glaciers". They can have even faster surface speed (up to 10 km a^{-1}), but it is typically uncertain how much of this speed is from sliding at the base. In an outlet glacier there is substantial vertical shear "up" in the ice column, sometimes caused

by soft temperate ice in a significant fraction of the thickness. Finally, outlet glaciers are strongly controlled by fjord-like, high slope bedrock topography. Figure 14 (right) compares the shallowness and bedrock topography of an outlet glacier and an ice stream. Few simplifying assumptions are appropriate for outlet glaciers, and the SSA may not be a sufficient model.

The shallow shelf approximation (SSA)

We state this stress balance equation only in the plane flow case, i.e. in a "flow line":

$$\left(2BH|u_x|^{1/n-1}u_x\right)_x - C|u|^{m-1}u = \rho g H h_x \tag{32}$$

The **red** term is the vertically-integrated longitudinal stress, also called the "membrane" stress when there are two horizontal variables. The **blue** term is the basal resistance, which is zero in an ice shelf. The negative of the term on the right is the driving stress. Thus the SSA equation is a balance wherein longitudinal strain rates are determined by the integrated ice hardness (i.e. the coefficient BH), the slipperyness of the bed (i.e. by the coefficient C and the power m) and the geometry of the ice sheet (i.e. the thickness H and the surface slope h_x).

In (32) the velocity u is independent of the vertical coordinate z. We assume that the ice hardness $B = A^{-1/n}$ is also independent of depth. (Thus models which are not isothermal must compute the vertical average of the hardness.) The basal resistance term $\tau_b = -C|u|^{m-1}u$ is often called a "sliding law" in power law form.

The coefficient $\bar\nu = B|u_x|^{1/n-1}$ in (32) is called the "effective viscosity", so that (32) can be written

$$\left(2\,\bar\nu\,Hu_x\right)_x - C|u|^{m-1}u = \rho g H h_x. \tag{33}$$

In form (33) it is understood that the viscosity $\bar\nu$ depends on the velocity solution u.

Equation (33) simplifies if the ice is floating, because we assume zero resistance is applied by the ocean. (And by the atmosphere, for that matter.) Also, the ice surface elevation is proportional to the thickness if the ice is floating. The inequality "$\rho H < -\rho_w b$" is sometimes called the *flotation criterion*. For grounded ice we know $\rho H > -\rho_w b$ and that the driving stress is $\rho g H h_x = \rho g H (H_x + b_x)$. On the floating side we know $\rho H < -\rho_w b$ and that $h = (1-\rho/\rho_w)H$ so the driving stress is $\rho g H h_x = \rho(1-\rho/\rho_w) g H H_x$. Thus the SSA becomes

$$\left(2\,\bar\nu\,Hu_x\right)_x = \rho g(1-\rho/\rho_w) H H_x \tag{34}$$

for floating ice. A useful observation about flow line equation (34) is that both left and right-hand expressions are derivatives; this fact can be used to build a 1D exact solution.

Steady ice shelf exact solution

For a steady 1D ice shelf, in which $H_t = 0$, the mass continuity equation (18) reduces to $M = (uH)_x$. Because this is simple, and because of the relative simplicity of the SSA equation (34) for 1D floating ice, the exact velocity and thickness for a steady ice shelf were computed by van der Veen [**?**]. This exact solution depends on the ice thickness H_g and velocity u_g at the grounding line. For the surface mass balance M we choose a positive constant M_0. These choices determine a unique solution, the derivation of which is left to the exercises.

Supposing $H_g = 500$ m, $u_g = 50\,\mathrm{m\,a^{-1}}$, and $M_0 = 30\,\mathrm{cm\,a^{-1}}$ we get the results in Figure 15, which are from code `exactshelf.m` (not shown). We will use this exact solution to verify a numerical SSA code. Note that driving stresses are much higher near the grounding line than away from it, and thus the highest longitudinal stresses, strain rates, and thinning rates occur near the grounding line.

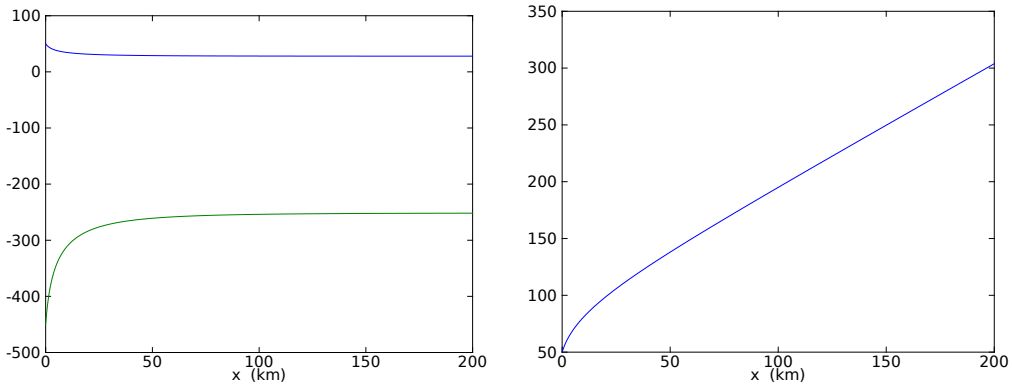


Figure 15: The upper and lower surface elevation (m; left) of the exact ice shelf solution and its velocity (m/a; right); $x = 0$ is the grounding line.

Numerical solution of the SSA

Suppose the ice thickness is a fixed function $H(x)$. To find the velocity we must solve the nonlinear PDE (33) or (34) for the unknown $u(x)$. When we do this numerically an iteration is needed because of the nonlinearity. The simplest iteration idea is to use an initial guess at the velocity, which allows us to compute an effective viscosity and then get a new velocity solution from a linear PDE problem. Then we recompute the effective viscosity, solve for a new velocity, and repeat until things stop changing. This is a "Picard" iteration.

Specifically, denote the previous velocity iterate as $u^{(k-1)}$ and the current iterate as $u^{(k)}$. Compute $\bar\nu^{(k-1)} = B|u_x^{(k-1)}|^{1/n-1}$ and define $W^{(k-1)} = 2\bar\nu^{(k-1)}H$. Solve this linear elliptic PDE for the unknown $u^{(k)}$ at each iteration:

$$\left(W^{(k-1)}u_x^{(k)}\right)_x - C|u^{(k-1)}|^{m-1}u^{(k)} = \rho g H h_x. \tag{35}$$

If the difference between $u^{(k-1)}$ and $u^{(k)}$ were ever to be zero then we would have a solution of (33). In practice we stop the iteration when the difference is small enough. Equation (35) is a linear boundary value problem. We can write it abstractly

$$\left(W(x)\,u_x\right)_x - \alpha(x)\,u = \beta(x) \tag{36}$$

where the functions $W(x)$, $\alpha(x)$, $\beta(x)$ are known in thiscontext. Equation (36) applies on an interval of the x-axis. For boundary conditions we will suppose that $x = x_g$ is a location where the velocity is known, $u(x_g) = u_g$, as in Figure 15. In the ice shelf case we also have the calving front condition

$$2BH|u_x|^{1/n-1}u_x = \frac{1}{2}\rho(1-\rho/\rho_w)gH^2 \tag{37}$$

at the end of the ice shelf $x = x_c$. (See Notes and References for a derivation of (37).) Boundary condition (37) can be solved for $u_x(x_c) = \gamma$ in terms of known quantities including the thickness at the calving front.

Where to get an initial guess $u^{(0)}$? Generally this may require effort, but we can give brief answers for the 1D cases. For floating ice, an initial velocity comes from assuming a uniform strain rate provided by the calving front condition: $u^{(0)}(x) = \gamma(x - x_g) + u_g$. For grounded ice, we may assume ice is held by basal resistance only: $u^{(0)}(x) = (-C^{-1}\rho g H h_x)^{1/m}$.

Numerics of the linear boundary value problem

Suppose equation (36) applies on $[x_g, x_c] = [0, L]$. We discretize with equal spacing Δx and index $j = 1, 2, \dots, J+1$, so that $x_1 = 0$ and $x_{J+1} = L$ are endpoints. The coefficient $W(x)$ is needed on a staggered grid, for stability and accuracy reasons similar to those for the SIA diffusivity. Our finite difference approximation of (36) is

$$\frac{W_{j+1/2}(u_{j+1} - u_j) - W_{j-1/2}(u_j - u_{j-1})}{\Delta x^2} - \alpha_j u_j = \beta_j \tag{38}$$

For the left end boundary condition we have $u_1 = u_g$ given, which is easy to include in the linear system (below). For the right end boundary condition we have $u_x(L) = \gamma$, which requires more thought [**?**]. First introduce a notional point x_{J+2}. Now require $(u_{J+2} - u_J)/(2\Delta x) = \gamma$. Using equation (38) in $j = J + 1$ case, eliminate the u_{J+2} variable "by-hand". This determines the form of the last equation in our linear system.

Thus each iteration to solve the SSA stress balance has the form

$$A\mathbf{v} = \mathbf{b}. \tag{39}$$

Indeed, at each location $x_1, \dots, x_{J+1}$ we can write an equation, basically a row of the matrix A in (39), involving the unknown velocities. In more detail it is this linear system of $J + 1$ equations:

$$\begin{bmatrix} 1 & & & & & \\ W_{3/2} & A_{22} & W_{5/2} & & & \\ & W_{5/2} & A_{33} & & & \\ & & \ddots & \ddots & & \\ & & W_{J-1/2} & A_{JJ} & W_{J+1/2} \\ & & & A_{J+1,J} & A_{J+1,J+1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_J \\ u_{J+1} \end{bmatrix} = \begin{bmatrix} u_g \\ \beta_2 \Delta x^2 \\ \beta_3 \Delta x^2 \\ \vdots \\ \beta_J \Delta x^2 \\ b_{J+1} \end{bmatrix} \tag{40}$$

The diagonal entries are

$$A_{22} = -(W_{3/2} + W_{5/2} + \alpha_2 \Delta x^2), \quad \dots \quad A_{JJ} = -(W_{J-1/2} + W_{J+1/2} + \alpha_J \Delta x^2).$$

There are special cases for the coefficients in the last equation,

$$A_{J+1,J} = 2W_{J+1/2}, \quad A_{J+1,J+1} = -(2W_{J+1/2} + \alpha_{J+1}\Delta x^2).$$

For the right side of the last equation, $b_{J+1} = -2\gamma \Delta x W_{J+3/2} + \beta_{J+1}\Delta x^2$.

System (40) is a tridiagonal linear system. But don't bother looking up how to solve such a linear system unless you really need to! It is fully appropriate to give system (40)

to Matlab's linear solver, the "backslash" operator $\mathbf{v} = A \setminus \mathbf{b}$, and not worry further about solving finite linear systems at this initial implementation stage. So we now have a code to solve (36) by finite differences and linear algebra, namely `flowline.m` below.

flowline.m

```
function u = flowline(L,J,gamma,W,alpha,beta,V0)

dx = L / J;
rhs = dx^2 * beta(:);
rhs(1) = V0;
rhs(J+1) = rhs(J+1) - 2 * gamma * dx * W(J+1);

A = sparse(J+1,J+1);
A(1,1) = 1.0;
for j=2:J
  A(j,j-1:j+1) = [ W(j-1), -(W(j-1) + W(j) + alpha(j) * dx^2), W(j) ];
end
A(J+1,J) = W(J) + W(J+1);
A(J+1,J+1) = - (W(J) + W(J+1) + alpha(J+1) * dx^2);

scale = full(max(abs(A),[],2));
for j=1:J+1,  A(j,:) = A(j,:) ./ scale(j);  end
rhs = rhs ./ scale;

u = A \ rhs;
```

By "manufacturing" exact solutions to (36)—see Notes and References—we can test this code. In fact, results from `testflowline.m` (not shown) demonstrate that our implemented numerical scheme converges at the optimal rate $O(\Delta x^2)$. We do this test before proceeding to solve the nonlinear SSA problem.

Solving the stress balance for an ice shelf

The code `ssaflowline.m` (below) numerically computes the velocity for an ice shelf. The thickness is assumed to be given, so we are not solving the full, "coupled" ice shelf problem. This code implements Picard iteration to solve the nonlinear equation (34). It calls `ssainit.m` (not shown) to get the initial iterate $u^{(0)}(x)$, as already described, and it calls `flowline.m` at each iteration. It also calls small helper functions `stagav,regslope,stagslope` (not shown) to computed certain gridded values.

ssaflowline.m

```
function [u,u0] = ssaflowline(p,J,H,b,ug,initchoice)

if nargin ~= 6, error('exactly 6 input arguments required'), end

dx = p.L / J;
x = (0:dx:p.L)';
xstag = (dx/2:dx:p.L+dx/2)';

alpha = p.C * ones(size(x));
h = H + b;
hx = regslope(dx,h);
beta = p.rho * p.g * H .* hx;
gamma = ( 0.25 * p.A^(1/p.n) * (1 - p.rho/p.rhow) *...
          p.rho * p.g * H(end) )^p.n;
```

```
u0 = ssainit(p,x,beta,gamma,initchoice);
u = u0;

Hstag = stagav(H);
tol = 1.0e-14;
eps_reg = (1.0 / p.secpera) / p.L;
maxdiff = Inf;
W = zeros(J+1,1);
iter = 0;
while maxdiff > tol
  uxstag = stagslope(dx,u);
  sqr_ux_reg = uxstag.^2 + eps_reg^2;
  W(1:J) = 2 * p.A^(-1/p.n) * Hstag .* sqr_ux_reg.^(((1/p.n)-1)/2.0);
  W(J+1) = W(J);

  unew = flowline(p.L,J,gamma,W,alpha,beta,ug);
  maxdiff = max(abs(unew-u));
  u = unew;
  iter = iter + 1;
end
```

Now we can ask precisely: Does `ssaflowline.m` work correctly? The exact velocity solution shown in Figure 15 allows us to compare the numerical and exact solutions by finding the maximum difference between them. To make a test out of this we take the exact thickness shown in Figure 15, from `exactshelf.m`. A convergence comparison, shown in Figure 16, is done by codes `testshelf.m` and `shelfconv.m` (not shown). Thus, at screen resolution, the numerical velocity solution from a coarse grid computation is identical to that shown in the right part of Figure 15. There is no need to show the numerical solution.

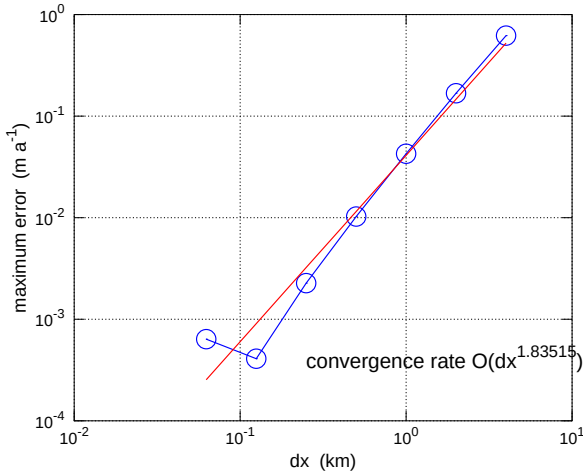


Figure 16: The numerical SSA velocity solution from `ssaflowline.m` converges to the exact solution, at nearly the optimal rate $O(\Delta x^2)$, as the grid is refined from spacing $\Delta x = 4$ km to $\Delta x = 62$ m. Each blue circle gives the maximum velocity error for a solution on a given grid. Even on the coarsest grid the error is less than 1 m/a while the maximum velocity is ~ 300 m/a.

Realistic ice shelf modelling

Real ice shelves have two horizontal variables. They are frequently confined in bays, and thus they experience "side drag". Their velocities vary spatially and temporally along their grounding "lines" (i.e. the curve where the flotation criterion is an equality). Furthermore real ice shelves have interesting boundary processes, including high basal melt near grounding lines, marine ice basal freeze-on, and fracturing which nears full thickness at the calving front.

Nonetheless "diagnostic" (i.e. fixed geometry) ice shelf modelling in two horizontal variables, done like the above example where the velocity is the model unknown but the thickness is known, is quite successful using only the isothermal SSA model. For example, Figure 17 shows a Parallel Ice Sheet Model (PISM) result for the Ross ice shelf, compared to observed velocities. There is only one tuned parameter, the constant value of the ice hardness B. Many current ice shelf models yield comparable match [**?**].

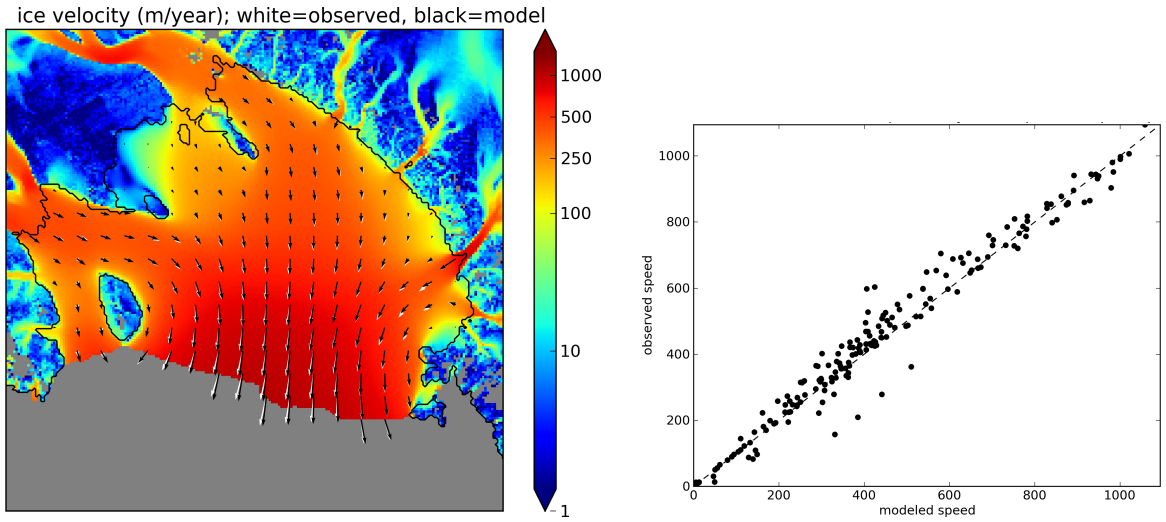


Figure 17: Results from tuning the single, constant ice hardness parameter B. Left: Observed radar-derived (white) and modeled (black) ice velocities are nearly coincident across the whole Ross ice shelf. The grounding line, at which observed velocities were applied as boundary conditions, is the thin black curve. Right: This scatter plot has one point for each velocity arrow at left.

1.10 A summary of numerical ice sheet modelling

These notes illustrate some general principles about numerical modelling:

- Spend time with the continuum model.
- Modularize your codes.
- Test the parts: Is the component robust? Does it show convergence to exact solutions?

Regarding the specific problems of ice flow, distributed stress balance equations like the SSA determine horizontal velocity from geometry and boundary conditions. These

equations are nonlinear so iteration is necessary. At each iteration a sparse matrix "inner" problem is solved; non-experts should give this matrix problem to a solver package. The SIA stress balance is not horizontally-distributed, however, and velocity follows immediately by integration of the driving stress.

The mass continuity equation has a time-derivative, and it describes how the ice geometry evolves. The numerical nature of this equation depends on which is the stress balance which supplies the velocity. It is a diffusion for frozen bed, large scale flows (SIA). It is *not* very diffusive for membrane stresses (e.g. SSA), especially with no basal resistance as in ice shelves. It has some diffusiveness for ice streams (but how much?), and more for outlet glaciers. In any case it is a kind of transport equation, but with diffusive character at larger spatial scales.

1.11 Notes

The SIA model, which was derived by several authors [**?**, **?**, **?**], follows by scaling the Stokes equations using the aspect ratio $\epsilon = [H]/[L]$, where $[H]$ is a typical thickness of an ice sheet and $[L]$ is a typical horizontal dimension. After scaling one drops the terms that are small if ϵ is small [**?**, **?**]; this is a "small-parameter argument". In one such derivation there are no $O(\epsilon)$ terms in the scaled equations so one only drops $O(\epsilon^2)$ terms [**?**]. The SIA is re-formulated as a well-posed free boundary problem in [**?**], which provides the correct boundary condition at grounded margins. The Mahaffy scheme for diffusivity used here [**?**] is not the only one [**?**].

The SSA model [**?**] was derived in [**?**] for ice shelves and in [**?**] for ice streams. A well-posed steady whole ice sheet model using only the SSA is in [**?**].

As noted, basal water is required for ice streaming. To model such water one must at least compute the ice base temperature and the basal melt rate through the energy conservation equation [**?**, **?**, **?**, **?**]. Evidence that outlet glaciers have thick temperate ice is in [**?**].

One of the most significant issues in modelling ice sheets using shallow models is to describe the "switch", in space and time, between shear-dominated and membrane-stress-dominated flow. It is not a good idea to abruptly switch from the SIA model to the SSA model at the edge of an ice stream, by whatever criterion that switch might be applied, though this has been attempted [**?**, **?**]; these might be called "horizontal hybrid" schemes. However, "vertical" hybrid schemes exist which solve the SIA and SSA everywhere in the ice sheet, combining the results without such an either/or decision [**?**, **?**, **?**, **?**]. The latter hybrids can be used at high spatial resolution and long time scales because the SSA model is an easier stress balance to solve than "higher-order" membrane-stress-resolving alternatives.

"Higher-order" approximations of the Stokes equations, such as the Blatter-Pattyn model [**?**, **?**], are also shallow approximations. Indeed, as both the SIA and the SSA are derived by small-parameter arguments from the Stokes equations, one might ask whether there is a common shallow antecedent model, to which Schoof and Hindmarsh [**?**] answer that Blatter-Pattyn is such. Computational limitations generally restrict either the spatial extent, the spatial resolution, or the model run duration of the more complete "higher-order", primarily because 3D stress balances involve more memory. Questions remain about what are the most important losses, relative to the Stokes model, from using either higher-order [**?**] or shallow hybrid models.

Solving the Stokes model itself [**?**] requires explicit accounting for incompressibility through a pressure variable. Numerical approximations of this stress balance are indefinite, thus harder to solve, essentially because incompressibility is an equality constraint on the flow. In the flow line one can address the incompressibility constraint by using stream functions [**?**].

Which are the best numerical models for moving grounding lines? Even when the minimal

SSA stress balance is used, this is still an open question [**?**, **?**]. The physics requires at least that the quantities H and u are continuous at $x = x_g$.

Where to find exact solutions for ice flow models? The textbook Greve and Blatter [**?**] has a few. Halfar's similarity solution to the SIA [**?**, **?**] was generalized to non-zero mass balance [**?**]. There are flowline [**?**] and cross-flow [**?**] solutions to the SSA model. For the Stokes equations themselves there are flowline solutions for constant viscosity [**?**]. As a last resort for numerical verification one can "manufacture" solutions by starting with a specified solution and then deriving a source term (or right-hand side) so that the specified function is actually a solution [**?**]. There are manufactured solutions to the thermomechanically-coupled SIA [**?**], manufactured flowline Blatter model solutions [**?**], and even manufactured solutions to the Glen flow law Stokes equations [**?**, **?**].

A key modelling issue omitted in this chapter is thermomechanical coupling. This is important because the ice softness $A(T)$ varies by three orders of magnitude in the temperature range relevant to ice sheet modelling. Ice temperature therefore gives ice sheet dynamics a long memory of past climate (e.g. [**?**]; note $\delta^{18}O$ and ice temperature are distinct measurements), and we see that geothermal flux is significant in slow-flowing parts of ice sheets. Equally important, dissipation of gravitational potential energy is major part of basal melt. For example, each year the Jakobshavn drainage basin in Greenland dissipates enough gravitational potential energy to melt more than $1\,\mathrm{km}^3$ of ice. "Enthalpy" is good way to track the energy content of polythermal ice masses [**?**], but it is not the only way [**?**].

The finite difference material in this Chapter should probably be read with reference [**?**] or similar in hand. The "main theorem for numerical PDE schemes" mentioned in the text is the Lax equivalence theorem [**?**]. Alternative numerical discretization techniques include the finite element [**?**], finite volume [**?**], and spectral [**?**] methods. Newton iteration for the nonlinear discrete equations is superior to Picard iteration used here, in terms of rapid convergence once iterates are near the solution, but implementation care is needed [**?**].

Exercises

1. Assume f has continuous derivatives of all orders. Show using Taylor's theorem:

$$f'(x) = \frac{f(x+\Delta) - f(x-\Delta)}{2\Delta} + O(\Delta^2) \quad \text{and} \quad f''(x) = \frac{f(x+\Delta) - 2f(x) + f(x-\Delta)}{\Delta^2} + O(\Delta^2).$$

2. Sometimes we want finite difference approximations for derivatives in-between grid points. Continuing exercise **1**, show $f'(x + (\Delta/2)) = (f(x+\Delta) - f(x))/\Delta + O(\Delta^2)$.

3. Rewrite `heat.m` using `for` loops instead of colon notation. (The only purpose here is to help understand colon notation.)

4. The 1D explicit scheme (22) for the heat equation, namely $T_j^{n+1} = \mu T_{j+1}^n + (1-2\mu)T_j^n + \mu T_{j-1}^n$, is averaging if stability criterion (25) holds. But of course we must be stepping *forward* in time. Show that the scheme is not averaging for any values of $\Delta t < 0$ and $\Delta x > 0$. Try running `heat.m` backward in time to see what happens. In general *there are no consistent stable numerical schemes for unstable PDE problems.*

5. Show that when written as a formula for T_j^{n+1}, scheme (26) has only positive coefficients. By looking into the literature as needed, explain why this shows it is unconditionally stable.

6. *This multi-part exercise concerns the numerical treatment of "*$\nabla \cdot (D\,\nabla)$*".*

(a) Show that if $D = D(x,y)$ and $u = u(x,y)$ then $\nabla \cdot (D\,\nabla u) = D\nabla^2 u + \nabla D \cdot \nabla u$.

(b) Write down the centered $O(\Delta t) + O(\Delta x^2)$ explicit finite difference method for the equation $u_t = D_0 u_{xx} + E_0 u_x$, assuming $D_0 > 0$ and E_0 are constant. Solve the scheme for the unknown u_j^{n+1}.

(c) Stability for your method will occur if the right hand side from the last answer in part (b) has all positive coefficients. If $|E_0| \gg D_0$, what does this say about Δt?

(d) Why do we use the staggered grid for "$\nabla \cdot (D\,\nabla)$," instead of expanding by the product rule as in part (a)?

7. Derive the Green's function of the 1D heat equation, namely $T = (4\pi Dt)^{-1/2}\, e^{-x^2/(4Dt)}$, which is a solution to $T_t = DT_{xx}$. Start by supposing there is a solution of the form $T(t,x) = t^{-1/2}\phi(s)$ where $s = t^{-1/2}x$ is the similarity variable. Thereby write down an ordinary differential equation for ϕ, and solve it.

8. In the text, and in code `verifysia.m`, we used Halfar's solution to verify our numerical scheme `siaflat.m`. Create the analogous code `verifyheat.m` to use the Green's function of the 2D heat equation (23), namely $T(t,x,y) = (4\pi Dt)^{-1}\, e^{-(x^2+y^2)/(4Dt)}$, to verify `heatadapt.m`. You can use the high quality approximation $e^{-A^2} \approx 0$ for $|A| > 10$ to choose a rectangular domain in space for which you may use $T = 0$ Dirichlet boundary conditions.

9. Is P. Halfar male or female, and what does the first initial "P." stand for?

10. Show that formula (29) solves (19) in the case of flat bed ($h = H$) and zero climatic mass balance ($M = 0$). You will want to express divergence and gradient in polar coordinates.

11. In the text it is claimed that any modification of `siaflat.m` will make the output of `verifysia.m` show non-convergence, e.g. the reported average thickness error will not go to zero as the grid is refined. By randomly altering lines of `siaflat.m`, or by other methods of your choice, evaluate this claim.

12. Some output from `verifysia.m` has been suppressed in the text, including a map-plane view of the numerical ice thickness error. Near the grounded margin of an ice sheet this error is much larger than elsewhere. Why? Would a higher-order model with moving margin on the same grid have significantly smaller thickness error, supposing we knew a relevant exact solution?

13. Derive the surface kinematical equation (30) from the mass continuity (18) and basal kinematical (31) equations. Note that the Leibniz rule for differentiating integrals, mentioned in the text, is

$$\frac{d}{dx}\left(\int_{g(x)}^{f(x)} h(x,y)\,dy\right) = f'(x)h(x,f(x)) - g'(x)h(x,g(x)) + \int_{g(x)}^{f(x)} h_x(x,y)\,dy.$$

14. Let $C_s = A(\rho g(1-r)/4)^n$ and assume $x_g = 0$ is the location of the grounding line. Derive the two parts of the van der Veen exact ice shelf solution, namely

$$u = \left[u_g^{n+1} + (C_s/M_0)\left((M_0 x + u_g H_g)^{n+1} - (u_g H_g)^{n+1}\right)\right]^{1/(n+1)}, \quad H = (M_0 x + u_g H_g)/u,$$

Start from equation (34), and use the fact $(H^2)_x = 2HH_x$ to generate the first integral of (34). Also use boundary condition (37). On the other hand, note that the mass continuity equation $M_0 = (uH)_x$ can be integrated to give the formula $uH = M_0(x - x_g) + u_g H_g$ for the flux. Find $u(x)$ first, then $H(x)$ is in terms of $u(x)$ as above. These exact solutions are from [**?**]. They are used in code `exactshelf.m`.

15. Modify the code `ssaflowline.m` to solve the ice stream SSA equation (33). For boundary conditions it would be reasonable to have fixed velocity at the upstream end, but keep the calving front boundary condition at the downstream end. (I.e. consider the case where the calving front is at the point where the ice stream reaches flotation.) Build a test case. Find an exact solution, if you can, to verify your code.