EISMINT: Lessons in Ice-Sheet Modeling [1]

Douglas R. MacAyeal

Department of Geophysical Sciences
University of Chicago
Chicago, Illinois

May 21, 1997

[1]These notes stem from many inspirations: (1) lecture notes associated with a glaciololgy class (GEOSCI 352) taught at the University of Chicago (this class was taught twice, improvements to the notes were made in Winter of 1995 with the help of H. Paul Jacobson, C. Hulbe, C. Jackson and V. Rommelaere), (2) notes from an ice-shelf modelling workshop held at the University of Chicago in July, 1992 (attended by Ed Waddington, Craig Lingle and David Schilling), (3) the EISMINT model-intercomparison (level 1) exercises (designed by P. Huybrechts and A. Payne), (4) an EISMINT workshop held at the Alfred Wegener Institut in Bremerhaven, Germany in June 1994, (5) the aftermath of the Bremerhaven EISMINT Workshop (where flaws in the ice-shelf code were found), (6) the 1994 MGM meeting at Ohio State University, (7) the 1995 EISMINT summer school in Grindlewald, (8) the 1996 EISMINT meeting in Brussels, and (9) the visit by Ralf Greve of THD Darmstadt to Chicago in 1994. These notes are in rough-draft form and many of the computing results have not been checked. In addition, the Matlab programs presented here reflect a progression of increasing programming skill on my part; some of the early programs are very inefficient and inelegant by today's standards. The latest chapter was created with the help of Byron Parizek who was on loan from Richard Alley's group at Penn State University.

Contents

Chapter 1

Level 1: Axisymmetric Ice-Sheet Flowline Model vs. Exact Solution

This chapter covers the finite-difference version of the ice-sheet model exercise in which a flowline model of a azimuthally symmetric ice sheet of circular plan form is compared with an exact, analytic solution. The main focus of this chapter is on how to numerically treat the diffusive mass-balance evolution equation of a grounded, frozen-bed ice sheet. The temperature profile of such an ice sheet has a strong influence on the speed of ice flow, however, to keep things simple, thermodynamics will be considered separately in Chapter 8. We will begin our analysis of grounded ice-sheet models by developing an exact, analytic solution of the governing equations (in steady state) following the early glaciological pioneers Nye and Vialov. We will use this analytic solution as a performance test on a finite-difference solution which we shall also develop. Here, our focus will be on one-dimensional flowline models of an axially symmetric ice sheet of circular plan form. We will consider a more general ice-sheet plan geometry, in particular a square geometry, in the next chapter.

1.1 Governing Equation

The steady-state mass balance equation [Huybrechts, 1992, ch. 4] for an azimuthally symmetric ice-sheet under the assumption that the ice-sheet bed is at a uniform elevation $z = 0$ is

$$\nabla \cdot \mathbf{q} - a = 0 \tag{1.1}$$

where $a = 0.3$ m/yr is the accumulation rate (assumed spatially uniform), and $\mathbf{q}$ is the ice-transport vector due to internal ice deformation [Huybrechts, 1992, ch. 4]

$$\mathbf{q} = -2(\rho g)^3 \left(\nabla z_s \cdot \nabla z_s\right) \nabla z_s \int_0^{z_s} \int_0^{z} A_o \left(z_s - z'\right)^3 dz' dz \tag{1.2}$$

where $\rho = 910$ kg/m^3 is the mean density of ice, $g = 9.81$ m/s^2 is the gravitational acceleration, z_s is the surface elevation (also assumed to be equal to the ice thickness for these exercises), ∇ is the two-dimensional (plan view) gradient operator, $\nabla\cdot$ is the two-dimensional (plan view) divergence operator, z is the vertical coordinate, and $A_o = 10^{-16}$ Pa^{-3}/yr is the assumed uniform value of a temperature-dependent flow-law rate constant. The integral on the right-hand side of Eqn. (1.2) may be evaluated as follows:

$$\begin{aligned}
\int_0^{z_s} \int_0^{z} A_o \left(z_s - z'\right)^3 dz' dz &= -A_o \int_0^{z_s} \int_{z_s}^{z_s - z} u^3 du dz \\
&= -A_o \int_o^{z_s} \frac{u^4}{4} \Big|_{z_s}^{z_s - z} dz \\
&= -A_o \frac{1}{4} \int_0^{z_s} \left((z_s - z)^4 - z_s^4 \right) dz \\
&= \frac{-A_o}{4} \left(\frac{-u^5}{5} \Big|_{z_s}^{0} - z_s^5 \right) \\
&= \frac{A_o z_s^5}{5}
\end{aligned} \tag{1.3}$$

The expression for $\mathbf{q}$ may now be simplified as follows (valid when the ice-sheet bed is flat at $z = 0$ and when the temperature-dependent flow-law constant A_o is uniform throughout the ice sheet):

$$\mathbf{q} = \frac{-2\left(\rho g\right)^3 A_o z_s^5}{5} \left(\nabla z_s \cdot \nabla z_s\right) \nabla z_s \tag{1.4}$$

A quick check of the units reveals that $\mathbf{q}$ has dimension of $\mathrm{m}^2\ \mathrm{s}^{-1}$, it is thus interpreted as a volume flux per unit cross-width.

With substitution of the expression given in Eqn. (1.4) for $\mathbf{q}$ in Eqn. (1.1), the steady-state mass balance equation becomes,

$$\nabla \cdot \left(\frac{2\left(\rho g\right)^3 A_o z_s^5}{5} \left(\nabla z_s\right)^3 \right) + a = 0 \tag{1.5}$$

It will be convenient for what follows to adopt nondimensional variables. Accordingly, we set

$$\begin{aligned} z_s &\rightarrow Z s \\ x, y &\rightarrow L x, y \end{aligned}$$

and choose scales Z and L to satisfy the following identity

$$\frac{2 A_o (\rho g)^3 Z^8}{5 L^4} = a \tag{1.6}$$

For $L = 750$ km and $a = 0.3$ m/yr, the above expression gives $Z = 2756.7$ m. Thus a nondimensional surface elevation s of 1 corresponds with a dimensional surface elevation of 2756.7 m. Our adoption of nondimensional variables means nothing more than an agreement about what "meter stick" we plan to measure space and velocity with. The use of nondimensional variables is completely arbitrary, and need not be followed by the reader. We choose to use nondimensional variables because of the possible simplifications and clarifications to the equations which may come later.

In nondimensional form, the governing equation becomes

$$\nabla \cdot \left(\left(\nabla s\right)^3 s^5 \right) + 1 = 0 \tag{1.7}$$

For future reference (see the next section), we record the nondimensional form of the radius coordinate used in cylindrical coordinate systems: $r \rightarrow Lr$.

1.2 Analytic Solution of Axisymmetric Ice-Sheet Mass Balance Equation

1.2.1 Justification

The Level 1 grounded ice-sheet model intercomparison test suggested by the EISMINT intercomparison group (as defined in the Bremerhaven workshop of 1994) involves a square domain for which there is not an exact, analytic solution for the steady-state ice-sheet surface profile. To achieve a comparison between model and exact, analytic solution, we must temporarily turn our attention to a new circular domain (with azimuthal symmetry) as shown in Fig. (1.1).

1.2.2 Derivation

The azimuthally symmetric form of Eqn. (1.7) written in cylindrical coordinates is

$$\frac{1}{r}\frac{d}{dr}\left(rs^5\left(\frac{ds}{dr}\right)^3\right)+1=0 \tag{1.8}$$

We integrate this equation once as follows [1]:

$$\begin{aligned}\frac{d}{dr}\left(rs^5\left(\frac{ds}{dr}\right)^3\right) &= -r\\ rs^5\left(\frac{ds}{dr}\right)^3 &= -\frac{r^2}{2}+c\\ s^5\left(\frac{ds}{dr}\right)^3 &= -\frac{r}{2}+\frac{c}{r}\end{aligned}$$

[1] This analytic solution was apparently derived by Nye and by Vialov. This references is in P. Huybrecht's paper on the EISMINT intercomparison test to be presented in September 1995 at Chamonix.

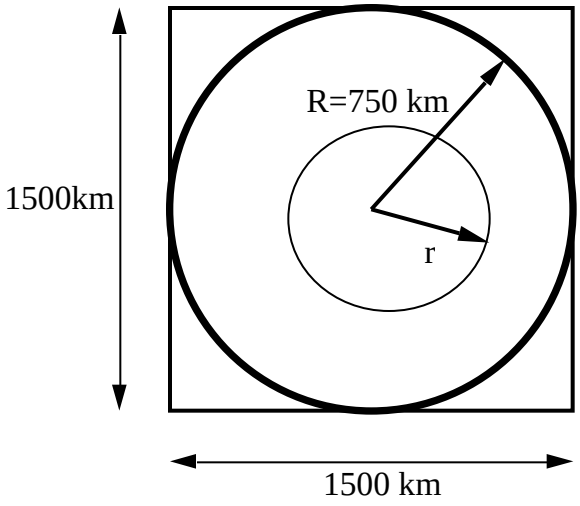

Figure 1.1: To get an analytic solution of the steady-state mass balance equation, it is necessary to simplify the problem by adopting cylindrical coordinates r and θ, and to assume azimuthal symmetry (*i.e.,* that z_s is independent of θ. The ice-sheet boundary condition is $z_s = 0$ at $r = R$.

$$s^{\frac{5}{3}}\frac{ds}{dr} = \left(-\frac{r}{2}+\frac{c}{r}\right)^{\frac{1}{3}} \tag{1.9}$$

We make use of the boundary condition $\frac{ds}{dr} = 0$ at $r = 0$ to deduce that the constant of integration c is equal to zero; thus,

$$s^{\frac{5}{3}}\frac{ds}{dr} = \left(-\frac{r}{2}\right)^{\frac{1}{3}} \tag{1.10}$$

Integrating again, we have

$$\begin{aligned}\frac{3}{8}s^{\frac{8}{3}}+d &= \int\left(\frac{-r}{2}\right)^{\frac{1}{3}}dr \\ &= -\int 6u^3du \\ &= -\frac{6}{4}\left(\frac{r}{2}\right)^{\frac{4}{3}}\end{aligned} \tag{1.11}$$

The constant of integration d can be evaluated by using the boundary condition $s(1) = 0$, where $r = 1$ is the outer edge of the ice sheet:

$$d = \frac{-6}{4}\left(\frac{1}{2}\right)^{\frac{4}{3}} \tag{1.12}$$

The final result is an analytic expression for the surface elevation in nondimensional variables:

$$s(r) = \left\{4\left[\left(\frac{1}{2}\right)^{\frac{4}{3}}-\left(\frac{r}{2}\right)^{\frac{4}{3}}\right]\right\}^{\frac{3}{8}} \tag{1.13}$$

In dimensional form, the above equation is written:

$$z_s(r) = Z\left\{4\left[\left(\frac{1}{2}\right)^{\frac{4}{3}}-\left(\frac{r}{2L}\right)^{\frac{4}{3}}\right]\right\}^{\frac{3}{8}} \tag{1.14}$$

A graph of this solution is created using the following MATLAB program on a MACINTOSH:

```
% This routine computes the analytic solution for an
% ice sheet of radius 1500/2 km.
%
g=9.81;
rho=910;
Ao=1/31556926 * 1e-16;
a=0.3/31556926;
L=1500e3/2;
Z=( 5*a*L^ 4/( 2 * Ao * (rho*g)^ 3 ) )^ (1/8);
r=linspace(0,1,101)';
s=( 4 * ( (1/2).^ (4/3) - (r/2).^ (4/3) ) ).^ (3/8);
plot(L*r,Z*s)
```

The graph of $z_s(r)$ created by the above MATLAB script is presented in Fig. (1.2). In the next section, we shall reproduce this exact, analytic solution using a numerical, finite-difference method.

1.3 Finite-Difference Solution of Axisymmetric Ice-Sheet Mass Balance Equation

We now attempt to reproduce the analytical solution using a time-dependent finite-difference model. Our strategy will be to begin our time-dependent model with an arbitrary initial condition (say, a uniform, near zero ice thickness) and to time-step the model through a sufficiently long period for the ice thickness to settle down to a steady, or near-steady state.

The nondimensional time dependent form of the mass-continuity equation is written

$$\frac{\partial s}{\partial t} = 1 + \frac{1}{r}\frac{\partial}{\partial r}\left(r s^5 \left(\frac{\partial s}{\partial r} \right)^3 \right) \tag{1.15}$$

where the nondimensional time t is defined using $t \rightarrow Tt$ where $T = \frac{5L^4}{2A_o(\rho g)^3 Z^7} =$

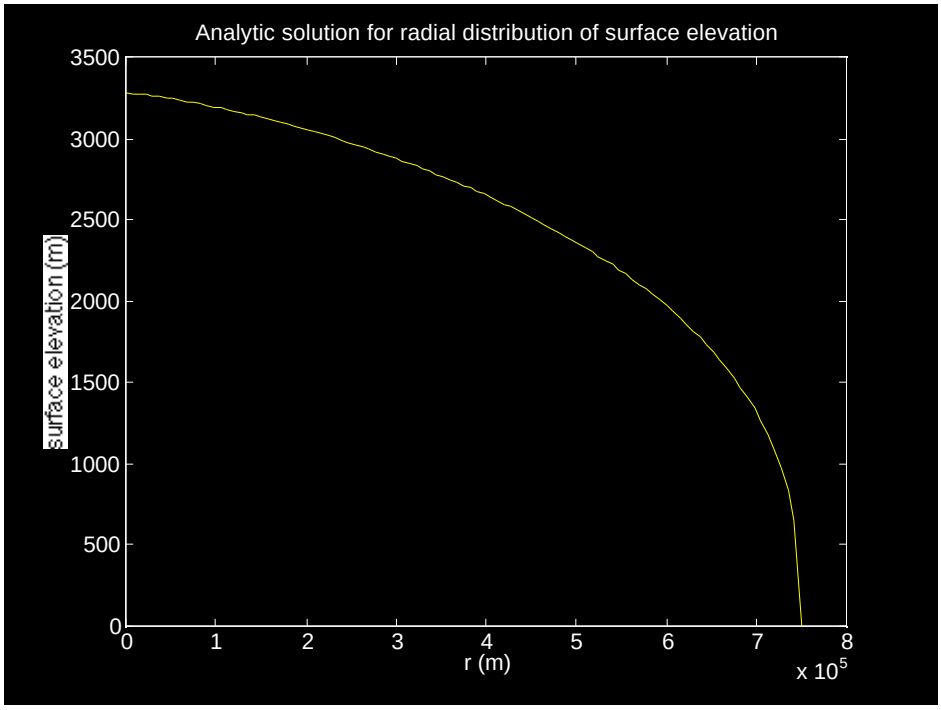

Figure 1.2: The analytic profile $z_s(r)$ computed using the expression in Eqn. (1.14). This profile provides a benchmark to which a numerical solution of the ice-sheet mass balance equation can be compared.

$\frac{Z}{a} \approx 9189$ years. Defining an effective diffusivity $d(s)$:

$$d = rs^5 \left(\frac{\partial s}{\partial r}\right)^2 \tag{1.16}$$

the mass-continuity equation becomes

$$\frac{\partial s}{\partial t} = 1 + \frac{1}{r}\frac{\partial}{\partial r}\left(d\frac{\partial s}{\partial r}\right) \tag{1.17}$$

We adopt a staggered-grid "flux-centered" numerical scheme to assure mass conservation [*e.g.*, Waddington, 1981]. This scheme is pictured in Fig. (1.3). Surface elevation s is defined as s_i^n at grid point $i = 1, \ldots, N$ and at time $t = (n-1)\Delta t$ where Δt is the time-step size. The effective diffusivity d is defined as d_i^n at $N-1$ grid points that are offset from the grid points where s is defined by half a grid spacing $\frac{\Delta r}{2}$, where $\Delta r = \frac{1}{N-1}$ is the nondimensional grid spacing. The finite-difference expression for d_i^n is

$$d_i^n = \frac{1}{2}\left(r_i + r_{i+1}\right)\left(\frac{s_i^n + s_{i+1}^n}{2}\right)^5 \left(\frac{s_{i+1}^n - s_i^n}{\Delta r}\right)^2 \tag{1.18}$$

where r_i is the value of r at the i'th grid point. The finite-difference version of Eqn. (1.17) is written using a fully implicit time step:

$$\frac{s_i^{n+1} - s_i^n}{\Delta t} = 1 + \frac{1}{r_i \Delta r^2}\left\{d_i^n\left(s_{i+1}^{n+1} - s_i^{n+1}\right) - d_{i-1}^n\left(s_i^{n+1} - s_{i-1}^{n+1}\right)\right\} \tag{1.19}$$

or,

$$s_{i-1}^{n+1}\left\{\frac{-d_{i-1}^n}{r_i \Delta r^2}\right\} + s_i^{n+1}\left\{\frac{1}{\Delta t} + \frac{d_i^n + d_{i-1}^n}{r_i \Delta r^2}\right\} + s_{i+1}^{n+1}\left\{\frac{-d_i^n}{r_i \Delta r^2}\right\} = \frac{s_i^n}{\Delta t} + 1 \tag{1.20}$$

The boundary conditions are $s = 0$ at $r = 1$ and $\mathbf{q} = 0$ at $r = 0$. The latter condition is tricky because it necessitates performing a "micro analysis" of mass balance at the ice divide.

Micro-analysis of ice-divide boundary condition

Following Waddington [1981], the ice-divide boundary condition is developed by considering the mass balance of a small region of circular planform centered on the ice divide of the ice sheet. As depicted in Fig. (1.4), a circular

area of radius $\frac{\Delta r}{2}$ is isolated and its mass balance is considered. The mass flux into the area due to snow fall is (in nondimensional units) πr^2 (where $r = \Delta r/2$). The mass flowing out of the area across the boundary at $r = \frac{\Delta r}{2}$ is $2\pi r s^5 \left(\frac{\partial s}{\partial r}\right)^3$ (where $r = \Delta r/2$). The rate of mass accumulation within the circular area is related to the rate of thickening, $\pi r^2 \frac{\partial s}{\partial t}$ (where $r = \Delta r/2$). The mass balance equation gives:

$$\pi r^2 \frac{\partial s}{\partial t} = \pi r^2 + 2\pi r \left(s^5 \left(\frac{\partial s}{\partial r} \right)^3 \right) \tag{1.21}$$

In finite-difference form, the above equation becomes:

$$s_1^{n+1} \left\{ \frac{1}{\Delta t} + \frac{4}{\Delta r^2} \left(\frac{s_1^n + s_2^n}{2} \right)^5 \left(\frac{s_2^n - s_1^n}{\Delta r} \right)^2 \right\} - s_2^{n+1} \left\{ \frac{4}{\Delta r^2} \left(\frac{s_1^n + s_2^n}{2} \right)^5 \left(\frac{s_2^n - s_1^n}{\Delta r} \right)^2 \right\} = 1 + \frac{s_1^n}{\Delta t} \tag{1.22}$$

(I am not entirely sure whether this above form is consistent with the finite-difference formulation used elsewhere in the grid. The results of the steady state experiment done below suggest that it is not.)

1.3.1 Exercise 1

Create a MATLAB script which simulates the approach to steady state ice-sheet thickness from an arbitrary initial condition (that you are free to choose). Use an explicit scheme (where the effective ice diffusivity d and the surface slope $\frac{\partial s}{\partial r}$ are evaluated at time step n in order to determine the solution at time step $n + 1$).

1.3.2 Implicit Time-Stepping Solution Using an Implicit Time Step and Matrix Notation

The finite-difference equation (1.20) together with the boundary conditions derived above, including (1.22), may be written conveniently in matrix notation:

$$\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R} \tag{1.23}$$

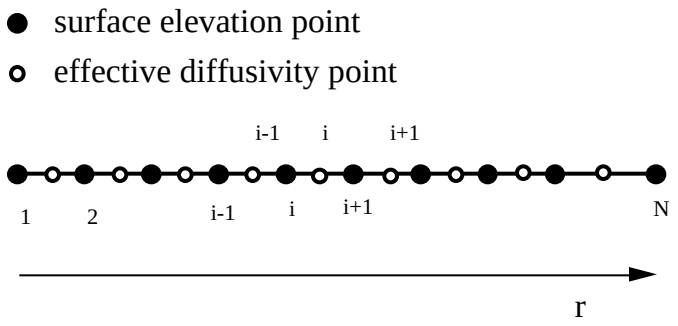

Figure 1.3: The staggered finite-difference grid used to solve the mass-continuity equation for comparison with the analytic solution.

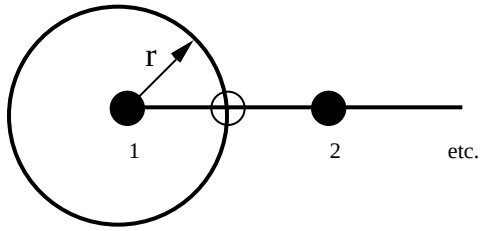

Figure 1.4: The mass balance of the ice divide is computed by considering the mass flux into and out of the circular cell with radius equal to half the grid spacing. Dark circles are grid points where s is defined, open circles are grid points where d is defined. The analysis considers the snow accumulation, the net volume gained, and the net flux across the boundary of the cell.

where, $\mathbf{s}^n$ is the column vector containing the s_i^n's:

$$\mathbf{s}^n = \begin{bmatrix} s_1^n \\ s_2^n \\ \vdots \\ s_N^n \end{bmatrix} \tag{1.24}$$

where $\mathbf{A}$ is the tri-diagonal matrix who's elements are:

$$\begin{aligned} A_{i,i} &= \frac{1}{\Delta t} + \frac{d_i^n + d_{i-1}^n}{r_i \Delta r^2} \\ A_{i,i-1} &= \frac{-d_{i-1}^n}{r_i \Delta r^2} \\ A_{i,i+1} &= \frac{-d_i^n}{r_i \Delta r^2} \end{aligned}$$

for $i = 2, \ldots, N-1$, and

$$\begin{aligned} A_{1,1} &= \frac{1}{\Delta t} + \frac{2}{\Delta r}\left(\frac{s_1^n + s_2^n}{2}\right)^5 \left(\frac{s_2^n - s_1^n}{\Delta r}\right)^2 \\ A_{1,2} &= \frac{-2}{\Delta r}\left(\frac{s_1^n + s_2^n}{2}\right)^5 \left(\frac{s_2^n - s_1^n}{\Delta r}\right)^2 \\ A_{N,N} &= 1 \end{aligned} \tag{1.25}$$

and $A_{i,j} = 0$ otherwise, and where the right-hand-side vector $\mathbf{R}$ is defined as follows:

$$\begin{aligned} R_i &= 1 + \frac{s_i^n}{\Delta t} \quad \text{for } i = 2, \ldots, N-1 \\ R_1 &= 1 + \frac{s_1^n}{\Delta t} \\ R_N &= 0 \end{aligned}$$

The above finite-difference implementation was employed (using native sparse matrix routines of MATLAB) to conduct the Level 1, steady state experiment of the EISMINT intercomparison exercise as articulated at the EISMINT intercomparison workshop held in Bremerhaven in 1994. Using an initial ice thickness of zero and a constant accumulation rate, the $s(r,t)$

was marched to steady state using 50-year time steps as shown in Fig. (1.5. (Longer time steps introduced grid-point-to-grid-point oscillations.) Due to the slowness of the MACINTOSH platform used to perform the computation, the run was conducted for only 50,000 years. A comparison between the finite-difference calculated ice-sheet surface profile at 50,000 years and the exact, analytic solution is made in Fig. (1.5). As mentioned previously, the temperature-depth profile is not computed in this particular implementation of the Level 1 test (the ice flow is uncoupled from the ice temperature in Level 1 tests, thus the lack of thermodynamics at this stage still yields a result that can be intercompared between different models). Following the convention of the EISMINT tests, 16 grid points were used to resolve the radial transect plotted in Fig. (1.5).

The surface-elevation gradient propagates "inland" in the model run shown in Fig. (1.5) by only one grid point per time step. This somewhat unphysical result stems from the fact that longitudinal stresses in the ice sheet are disregarded [*e.g.,* Huybrechts, 1992, ch. 4].

The MATLAB program used to compute the finite-difference evolution of the axisymmetric ice sheet is listed as follows:

```
% Finite-difference solution of mass balance equations in an
% axisymmetric domain:
%
hold off
clg
N=16;
g=9.81;
rho=910;
Ao=1/31556926 * 1e-16;
a=0.3/31556926;
L=1500e3/2;
Z=( 5*a*L^ 4/( 2 * Ao * (rho*g)^ 3 ) )^ (1/8);
r=linspace(0,1,N)';
s_exact=( 4 * ( (1/2).^ (4/3) - (r/2).^ (4/3) ) ).^ (3/8);
%plot(L*r,Z*s_exact); pause
%
```

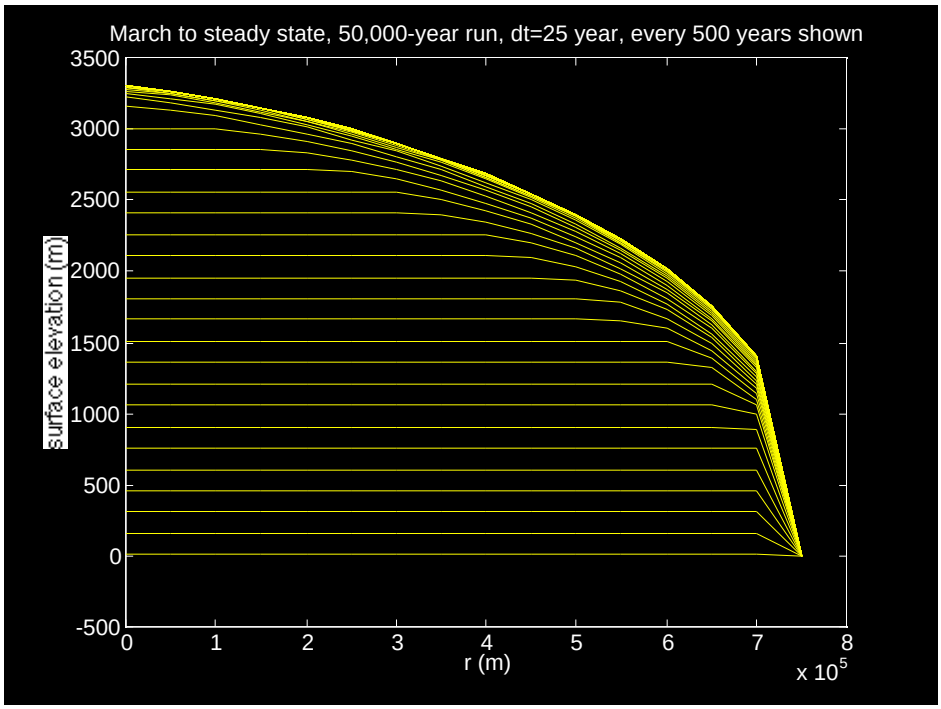

Figure 1.5: Plots of surface elevation $z_s(r,t)$ at every 500 years for constant accumulation rate. At $t = 0$ the ice thickness is assumed zero. Steady-state is reached after about 50,000 years. A run longer than 50,000 years (at a 25-year time step) was unfeasable for a MACINTOSH computing platform. This corresponds to the Level 1, steady state exercise of the EISMINT notes. The exact, analytic surface profile is denoted by asterisks. Internal ice temperature is not calculated in this experiment.

```
%
nsteps=2000;
dt=25*31556926*a/Z;
dr=1.0/(N-1);
%
sn=zeros(N,1);
R=zeros(N,1);
d=zeros(N,1);
AU=zeros(N,1);
AD=zeros(N,1);
AL=zeros(N,1);
%
plot(L*r,Z*s_exact,'r*'); hold on
for n=1:nsteps
for i=1:N-1
d(i)=.5*(r(i)+r(i+1))* (.5*(sn(i)+sn(i+1)))^ 5 ...
* ((sn(i+1)-sn(i))/dr)^ 2;
end
AD(1)=1/dt+((sn(1)+sn(2))/2)^ 5*((sn(2)-sn(1))/dr)^ 2*4/dr^ 2;
AU(2)=-((sn(1)+sn(2))/2)^ 5*((sn(2)-sn(1))/dr)^ 2*4/dr^ 2;
AD(N)=1;
AL(N-1)=0;
R(1)=1+sn(1)/dt;
R(N)=0;
for i=2:N-1
AD(i)= 1/dt + (d(i) + d(i-1))/(r(i)*dr^ 2);
AL(i-1)= -d(i-1)/(r(i)*dr^ 2);
AU(i+1)= -d(i)/(r(i)*dr^ 2);
R(i)=1+sn(i)/dt;
end
T=spdiags([AL AD AU],[-1 0 1],N,N);
sn=T\ R;
if rem(n,20) == 1
plot(L*r,Z*sn);
end
end
```

Observe that the tridiagonal nature of the matrix **A** has been used to make the computation more efficient. The MATLAB -native sparse matrix routine `spdiags` have been used to construct the sparse matrix `T` from the three main diagonals of **A**. The solution of `T*sn = R` for `sn` is performed using the MATLAB -native sparse matrix solver denoted by the backslash, *i.e.* `sn=T\R`.

A much better MATLAB script developed by students at the University of Chicago is listed as follows. Can you identify the coding improvements?

```
% This is an implicit time-stepping solution for the azimuthally
symmetric ice sheet.

N=25;
r=linspace(0,1,N)';
dr=1/(N-1);

% Initial condition
s_exact=( 4*( (1/2)^(4/3) - (r/2).^(4/3) ) ).^(3/8);
%s=zeros(N,1);
s=1.05*s_exact;
d=zeros(N,1);
%A=zeros(N,N);
R=zeros(N,1);

figure(2)
clg
plot(L*r,Z*s_exact,'ro'), hold on


nsteps=150;
g=9.81;
rho=910;
Ao=1e-16 *1/31556926;
L=750e3;
a=0.3/31556926;
Z=( 5*a*L^4 / (2*Ao*(rho*g)^3) )^(1/8)
```

```
dt=10*31556926*a/Z;

row=zeros(3*(N-2),1);
col=zeros(3*(N-2),1);
value=zeros(3*(N-2),1);
countrow=-2;
for i=2:N-1
countrow=countrow+3;
row(countrow:countrow+2)=[i i i]';
col(countrow:countrow+2)=[i i-1 i+1]';
end


count=0;
ngraph=10;
flops(0);
for n=1:nsteps

% compute d from s:
for i=1:N-1
d(i)= (r(i)+r(i+1))/2 * ( (s(i)+s(i+1))/2 )^5 * ( (s(i+1)-s(i))/dr
)^2 ;
end
% construct A and R:


countrow=-2;
for i=2:N-1
countrow=countrow+3;
value(countrow:countrow+2)= ...
[1/dt+(d(i)+d(i-1))/r(i)/dr^2 -d(i-1)/r(i)/dr^2 -d(i)/r(i)/dr^2 ]';
R(i)=1+s(i)/dt;
end
A=sparse(row,col,value,N,N);
A(N,N)=1;
R(N)=0;
A(1,1)=1/dt+16/dr^3*d(1);
```

```
A(1,2)=-16/dr3*d(1);
R(1)=1+s(1)/dt;

% solve for new value of s at time step n+1:

s=A\R;

count=count+1;
if count==ngraph
count=0;
figure(2)
plot(L*r,Z*s,'g-')
end

end
flopstodoimplicit=flops
```

The solution, $z_s(r, t = 50000)$ at the end of the model run for the Level 1 steady-state exercise is listed as follows (see also it's graph in Fig. (1.5)):

$$z_s(t = 50000) = 1.0 \times 10^3 \times \begin{bmatrix} 3.3017 \\ 3.2666 \\ 3.2149 \\ 3.1517 \\ 3.0785 \\ 2.9954 \\ 2.9021 \\ 2.7976 \\ 2.6806 \\ 2.5488 \\ 2.3987 \\ 2.2251 \\ 2.0185 \\ 1.7610 \\ 1.4062 \\ 0.0000 \end{bmatrix} \tag{1.26}$$

The corresponding exact, analytic solution is:

$$z_s(t = 50000) = 1.0 \times 10^3 \times \begin{bmatrix} 3.2783 \\ 3.2448 \\ 3.1928 \\ 3.1289 \\ 3.0548 \\ 2.9706 \\ 2.8760 \\ 2.7699 \\ 2.6509 \\ 2.5164 \\ 2.3629 \\ 2.1844 \\ 1.9706 \\ 1.7005 \\ 1.3171 \\ 0 \end{bmatrix} \tag{1.27}$$

The finite-difference solution appears to produce a steady-state ice sheet of slightly larger volume than that of the exact, analytic solution. Although not checked rigorously, I suspect that the problem lies in the implementation of the ice-divide boundary condition described above. Consult [Waddington, 1981] for a discussion of the implementation of ice-divide boundary conditions.

1.3.3 Diagnostics: Fluxes and Velocities

The dimensional forms of the radial mass flux q (m^2/s) and depth-averaged radial velocity $\bar{u}_r$ (m/s) are computed diagnostically using the following finite-difference formulae:

$$q(r) = \frac{2(\rho g)^3 A_o z_s^5}{5} \left(\frac{\partial z_s}{\partial r} \right)^3 \tag{1.28}$$

$$\bar{u}_r(r) = \frac{q}{z_s} \tag{1.29}$$

(recall that the ice thickness in this exercise is equal to the surface elevation z_s due to the flat bottom topography at $z = z_b = 0$). These diagnostics can be compared with the exact, steady-state results derived from mass-balance considerations:

$$q_e(r) = \frac{\pi r^2 a}{2\pi r} = \frac{ra}{2} \tag{1.30}$$

$$\bar{u}_{re} = \frac{q_e}{z_{se}} \tag{1.31}$$

where subscripts e denote the exact values. These diagnostic quantities are compared in Figs. (1.6) and 1.7). The MATLAB script used to obtain these quantities is listed below:

```
% This program computes diagnostics associated with ice-sheet model
%
% Mass Flux:  (defined at half-step grid points)
%
q=zeros(N-1,1);
q_exact=zeros(N-1,1);
ubar=zeros(N-1,1);
ubar_exact=zeros(N-1,1);
for i=1:N-1
q(i)=((-sn(i)-sn(i+1))/2)^ 5*((sn(i+1)-sn(i))/dr)^ 3...
Z^ 8/L^ 3*(2/5)*(rho*g)^ 3*Ao;
q_exact(i)=a*(dr*(i-1)+dr/2)*L/2;
ubar(i)=q(i)/((sn(i)+sn(i+1))/2);
ubar_exact(i)=q_exact(i)/((s_exact(i)+s_exact(i+1))/2);
end
hold off
clg
plot(L*r(1:N-1)+dr/2*L,q); hold on
plot(L*r(1:N-1)+dr/2*L,q_exact,'g-');pause
hold off
clg
plot(L*r(1:N-1)+dr/2*L,ubar);hold on
plot(L*r(1:N-1)+dr/2*L,ubar_exact,'g-')
```

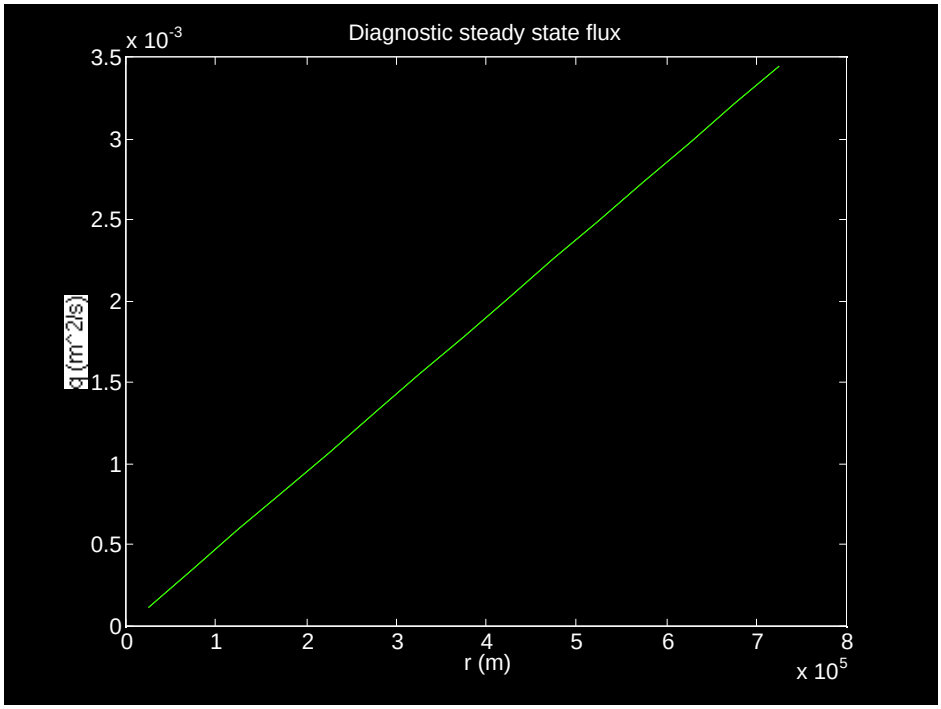

Figure 1.6: Diagnostic mass flux q (m^2/s) (finite-difference and exact).

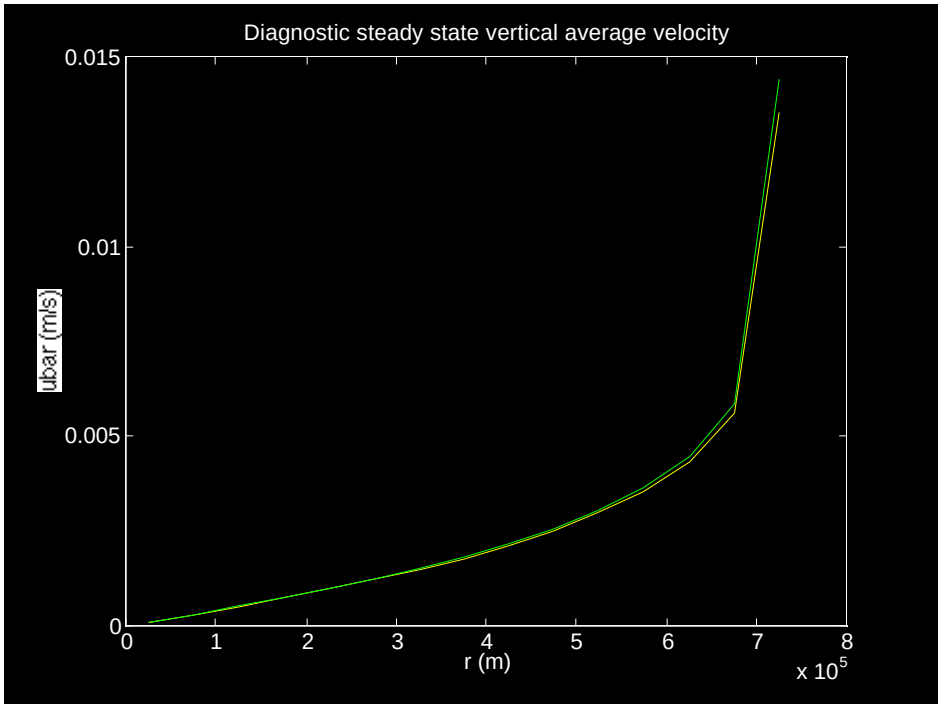

Figure 1.7: Diagnostic depth-average radial velocity (finite-difference and exact).

The radial velocity profile $u_r(z)$ is defined in terms of z_s and $\frac{\partial z_s}{\partial r}$ as follows [*e.g.,* Huybrechts, 1992, ch. 4]:

$$u_r(z) = -2(\rho g)^3 \left(\frac{\partial z_s}{\partial r}\right)^3 \int_0^z A_o (z_s - z')^3 dz' \tag{1.32}$$

The simple, temperature-independent form of the flow law parameter A_o allows us to integrate the right-hand side of the above expression to obtain:

$$u_r(z) = \frac{A_o}{2}(\rho g)^3 \left\{(z_s - z)^4 - z_s^4\right\} \left(\frac{\partial z_s}{\partial r}\right)^3 \tag{1.33}$$

The vertical velocity $w(r, z, t)$ is obtained from $u_r(r, z, t)$ using the incompressibility condition which, in cylindrical coordinates, is written:

$$\frac{\partial w}{\partial z} + \frac{1}{r}\frac{\partial}{\partial r}(r u_r) = 0 \tag{1.34}$$

By integrating the above expression for $\frac{\partial w}{\partial z}$ over z, and use of the no-vertical-flow boundary condition at $z = 0$, we obtain the rather tedious expression for $w(r, z, t)$:

$$\begin{aligned} w(z) = \frac{-A_o(\rho g)^3}{2} & \\ & \times \left\{ \frac{1}{r}\left(\frac{\partial z_s}{\partial r}\right)^3 \left\{ \frac{1}{5}\left[z_s^5 - (z_s - z)^5\right] - z_s^4 z \right\} \right. \\ & +4\left(\frac{\partial z_s}{\partial r}\right)^4 \left\{ \frac{1}{4}\left[z_s^4 - (z_s - z)^4\right] - z_s^3 z \right\} \\ & \left. +3\left(\frac{\partial z_s}{\partial r}\right)^2 \frac{\partial^2 z_s}{\partial r^2} \left\{ \frac{1}{5}\left[z_s^5 - (z_s - z)^5\right] - z_s^4 z \right\} \right\} \end{aligned} \tag{1.35}$$

It is important to note that the above expression does *not* hold for the ice divide. The ice divide is a special location where the vertical velocity field cannot be defined without appeal to second-order effects such as the longitudinal strain rates [*e.g.,* Raymond, 1983].

The expression for the vertical velocity given in Eqn. (1.35) is rather tedious to derive (although not difficult) by integration of Eqn. (1.34). To

check the result, we make use of the definition of the vertical velocity at the ice-sheet surface given to us by the kinematic boundary condition at the free surface:

$$w(z_s) = -a - \frac{-A_o(\rho g)^3}{2} z_s^4 \left(\frac{\partial z_s}{\partial r}\right)^4 \tag{1.36}$$

The expression we wish to check (Eqn. 1.35), when evaluated at $z = z_s$, gives

$$\begin{aligned}
w(z_s) &= \frac{A_o(\rho g)^3}{2}\left\{\frac{4}{5r}\left(\frac{\partial z_s}{\partial r}\right)^3 z_s^5 + 3\left(\frac{\partial z_s}{\partial r}\right)^4 z_s^4 + \frac{12}{5}\left(\frac{\partial z_s}{\partial r}\right)^2 \frac{\partial^2 z_s}{\partial r^2} z_s^5\right\} \\
&= \frac{2A_o(\rho g)^3}{5}\left\{\left(\frac{\partial z_s}{\partial r}\right)^3 z_s^5 + \frac{15}{4}\left(\frac{\partial z_s}{\partial r}\right)^4 z_s^4 + 3\left(\frac{\partial z_s}{\partial r}\right)^2 \frac{\partial^2 z_s}{\partial r^2} z_s^5\right\} \\
&= \frac{2A_o(\rho g)^3}{5}\left\{\left(\frac{\partial z_s}{\partial r}\right)^3 z_s^5 + 5\left(\frac{\partial z_s}{\partial r}\right)^4 z_s^4 + 3\left(\frac{\partial z_s}{\partial r}\right)^2 \frac{\partial^2 z_s}{\partial r^2} z_s^5 - \frac{5}{4}\left(\frac{\partial z_s}{\partial r}\right)^4 z_s^4\right\} \\
&= \frac{2A_o(\rho g)^3}{5}\left\{\frac{1}{r}\frac{\partial}{\partial r}\left(r z_s^5 \left(\frac{\partial z_s}{\partial r}\right)^3\right) - \frac{5}{4}\left(\frac{\partial z_s}{\partial r}\right)^4 z_s^4\right\} \\
&= -a - \frac{A_o(\rho g)^3}{2}\left(\frac{\partial z_s}{\partial r}\right)^4 z_s^4
\end{aligned} \tag{1.37}$$

where we have made use of the mass-continuity equation $\frac{2(\rho g)^3 A_o}{5}\frac{1}{r}\frac{\partial}{\partial r}\left(r z_s^5 \left(\frac{\partial z_s}{\partial r}\right)^3\right) = -a$. The above result is the same as that defined by the kinematic boundary condition for the free surface, and this gives us confidence that the complicated expression for $w(z)$ in Eqn. (1.35) is correct.

Horizontal and vertical velocity associated with Level 1 test

The horizontal and vertical velocity fields are defined at the half-grid points (the open circles in the staggered grid scheme shown in Fig. 1.3). Thus they are not defined at the ice divide where EISMINT Level 1 test diagnostics are requested. They are, however, defined at the half-way point between the ice divide and terminus. With 16 grid points defining the length of the model domain from ice divide to terminus, the half-grid point that is $\frac{1}{2}$

the distance to the margin (where diagnostics are requested by the EISMINT intercomparison test) corresponds to half-grid point number 8. The following MATLAB script was used to perform the analysis for the horizontal (radial) velocity component:

```
% This program computes horizontal velocity
% associated with the axisymmetric ice-sheet model in the Level 1
steady state test
%
M=10; % Number of points in vertical
z=zeros(M,N-1);
z_exact=zeros(M,N-1);
u=zeros(M,N-1);
u_exact=zeros(M,N-1);
dr=L/(N-1);
zs=Z*sn;
zs_exact=Z*s_exact;
for i=1:N-1
z(:,i)=linspace(0,(zs(i)+zs(i+1))/2,M)';
z_exact(:,i)=linspace(0,(zs_exact(i)+zs_exact(i+1))/2,M)';
u(:,i)=Ao*(rho*g)^ 3/2*((zs(i+1)-zs(i))/dr)^ 3...
( ((zs(i)+zs(i+1))/2 - z(:,i)).^ 4 ...
- ((zs(i)+zs(i+1))/2)^ 4 );
u_exact(:,i)=Ao*(rho*g)^ 3/2...
((zs_exact(i+1)-zs_exact(i))/dr)^ 3 ...
( ((zs_exact(i)+zs_exact(i+1))/2 - z_exact(:,i)).^ 4 ...
- ((zs_exact(i)+zs_exact(i+1))/2)^ 4 );
end
hold off
clg
plot(u(:,8),z(:,8))
hold on
plot(u_exact(:,8),z_exact(:,8))
```

A graph of the $u(z, r = \frac{L}{2})$ associated with the finite-difference and exact versions of the steady-state ice thickness profile is displayed in Fig. (1.8).

The numerical data is presented in the form of a column vector $\mathbf{u}$ where the first element corresponds to the radial velocity at $z = 0$ and the last (tenth) element corresponds to the radial velocity at $z = z_s$. For the finite-difference solution:

$$\mathbf{u}(r = \frac{L}{2}) = 10^{-6} \times \begin{bmatrix} 0 \\ 0.3056 \\ 0.5158 \\ 0.6528 \\ 0.7360 \\ 0.7817 \\ 0.8034 \\ 0.8115 \\ 0.8133 \\ 0.8134 \end{bmatrix} \quad \text{m/s} \tag{1.38}$$

For the exact solution:

$$\mathbf{u}_e(r = \frac{L}{2}) = 10^{-6} \times \begin{bmatrix} 0 \\ 0.3086 \\ 0.5208 \\ 0.6592 \\ 0.7432 \\ 0.7894 \\ 0.8113 \\ 0.8194 \\ 0.8213 \\ 0.8214 \end{bmatrix} \quad \text{m/s} \tag{1.39}$$

As expected, the finite-difference solution does not exactly match the exact, analytic solution. Again, I attribute this to the ice-divide boundary condition of the mass-balance equation which may not be completely consistent in the formulation I have developed here.

The following MATLAB script was used to display the vertical velocity at the half-way point between ice divide and terminus:

```
% This program computes vertical velocity associated with the ice-sheet
model
% in the Level 1, axisymmetric steady state test
```

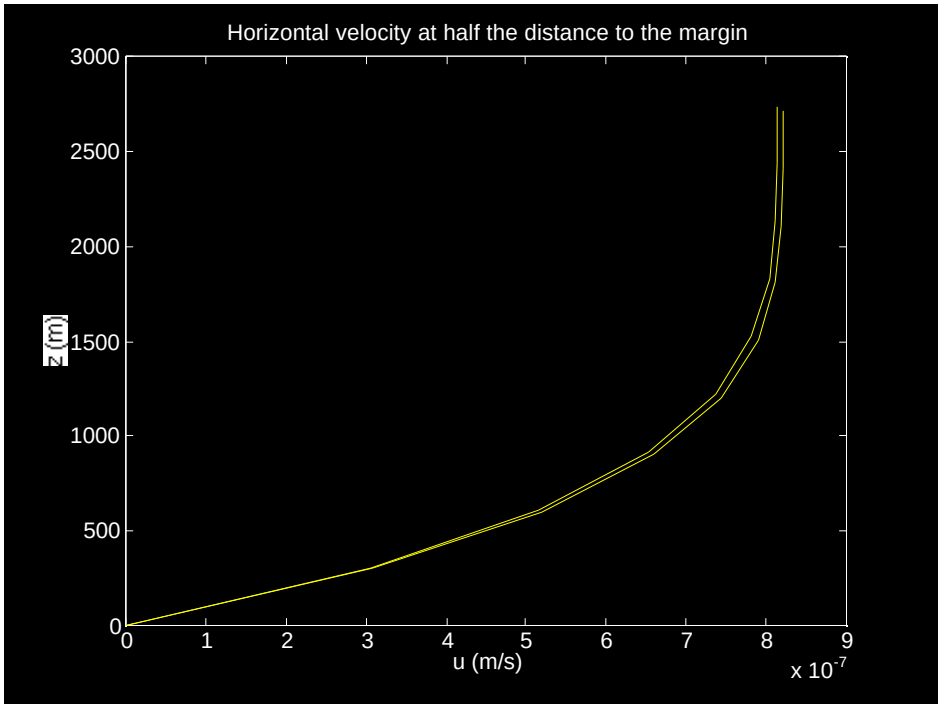

Figure 1.8: Diagnostic radial velocity (m/s) as a function of z at the point half-way to between the ice divide and the terminus (finite-difference and exact are shown together).

```
M=10; z=zeros(M,N-1);
z_exact=zeros(M,N-1);
w=zeros(M,N-1);
w_exact=zeros(M,N-1);
dr=L/(N-1);
zs=Z*sn;
zs_exact=Z*s_exact;
for i=2:N-2 z(:,i)=linspace(0,(zs(i)+zs(i+1))/2,M)';
z_exact(:,i)=linspace(0,(zs_exact(i)+zs_exact(i+1))/2,M)';
w(:,i)=-Ao*(rho*g)^ 3/2* ( ...
1/((i-1)*dr+dr/2) * ((zs(i+1)-zs(i))/dr)^ 3 ...
( 1/5 * ( ((zs(i)+zs(i+1))/2)^ 5 - ...
( ((zs(i)+zs(i+1))/2) - z(:,i) ).^ 5 ) ...
- ((zs(i)+zs(i+1))/2)^ 4*z(:,i) ) ...
+4 * ((zs(i+1)-zs(i))/dr)^ 4 ...
( 1/4 * ( ((zs(i)+zs(i+1))/2)^ 4 -...
( ((zs(i)+zs(i+1))/2) - z(:,i) ).^ 4 ) ...
- ((zs(i)+zs(i+1))/2)^ 3*z(:,i) ) ...
+3 * ((zs(i+1)-zs(i))/dr)^ 2 ...
( (zs(i+1)+zs(i-1)-2*zs(i))/dr^ 2 + ...
(zs(i+2)+zs(i)-2*zs(i+1))/dr^ 2 )/2 ...
( 1/5 * ( ((zs(i)+zs(i+1))/2)^ 5 - ...
( ((zs(i)+zs(i+1))/2) - z(:,i) ).^ 5 ) ...
- ((zs(i)+zs(i+1))/2)^ 4*z(:,i) )...
);


w_exact(:,i)=-Ao*(rho*g)^ 3/2* ( ...
1/((i-1)*dr+dr/2) * ((zs_exact(i+1)-zs_exact(i))/dr)^ 3 ...
( 1/5 * ( ((zs_exact(i)+zs_exact(i+1))/2)^ 5 - ...
( ((zs_exact(i)+zs_exact(i+1))/2) - z_exact(:,i) ).^ 5 ) ...
- ((zs_exact(i)+zs_exact(i+1))/2)^ 4*z_exact(:,i) ) ...
+4 * ((zs_exact(i+1)-zs_exact(i))/dr)^ 4 ...
( 1/4 * ( ((zs_exact(i)+zs_exact(i+1))/2)^ 4 - ...
( ((zs_exact(i)+zs_exact(i+1))/2) - z_exact(:,i) ).^ 4 ) ...
- ((zs_exact(i)+zs_exact(i+1))/2)^ 3*z_exact(:,i) ) ...
```

```
+3 * ((zs_exact(i+1)-zs_exact(i))/dr)^ 2 ...
( (zs_exact(i+1)+zs_exact(i-1)-2*zs_exact(i))/dr^ 2 + ...
(zs_exact(i+2)+zs_exact(i)-2*zs_exact(i+1))/dr^ 2 )/2 ...
( 1/5 * ( ((zs_exact(i)+zs_exact(i+1))/2)^ 5 - ...
( ((zs_exact(i)+zs_exact(i+1))/2) - z_exact(:,i) ).^ 5 ) ...
- ((zs_exact(i)+zs_exact(i+1))/2)^ 4*z_exact(:,i) )...
);


end
hold off
clg
plot(31556926*w(:,8),z(:,8))
hold on
plot(31556926*w_exact(:,8),z_exact(:,8))
```

A graph of $w(z)$ at the half-way point between the ice divide and terminus is displayed in Fig. (1.9). The finite-difference and exact versions of the column vector $\mathbf{w}$ are:

$$\mathbf{w}(r = \frac{L}{2}) = \begin{bmatrix} 0 \\ -0.0109 \\ -0.0386 \\ -0.0769 \\ -0.1213 \\ -0.1688 \\ -0.2177 \\ -0.2668 \\ -0.3158 \\ -0.3648 \end{bmatrix} \quad \text{m/yr} \tag{1.40}$$

For the exact solution:

$$\mathbf{w}_e(r = \frac{L}{2}) = \begin{bmatrix} 0 \\ -0.0110 \\ -0.0389 \\ -0.0774 \\ -0.1220 \\ -0.1698 \\ -0.2188 \\ -0.2682 \\ -0.3174 \\ -0.3666 \end{bmatrix} \quad \text{m/ry} \tag{1.41}$$

summary

In the preceding analysis, we have created a finite-difference model of an axisymmetric ice sheet and have compared it's solution with an exact, analytic expression for the steady-state thickness profile.

1.4 Exercise 2

Set up the finite-difference equations to determine the steady-state ice-sheet profile without using a time marching scheme. In other words, set up the finite difference form of

$$\frac{\partial}{\partial r}\left(d\frac{\partial s}{\partial r}\right) + r = 0 \tag{1.42}$$

where,

$$d = rs^5\left(\frac{\partial s}{\partial r}\right) \tag{1.43}$$

Part A Determine a solution using MATLAB for $d = .1471$, compare your result graphically with the exact, analytic expression for steady-state s.

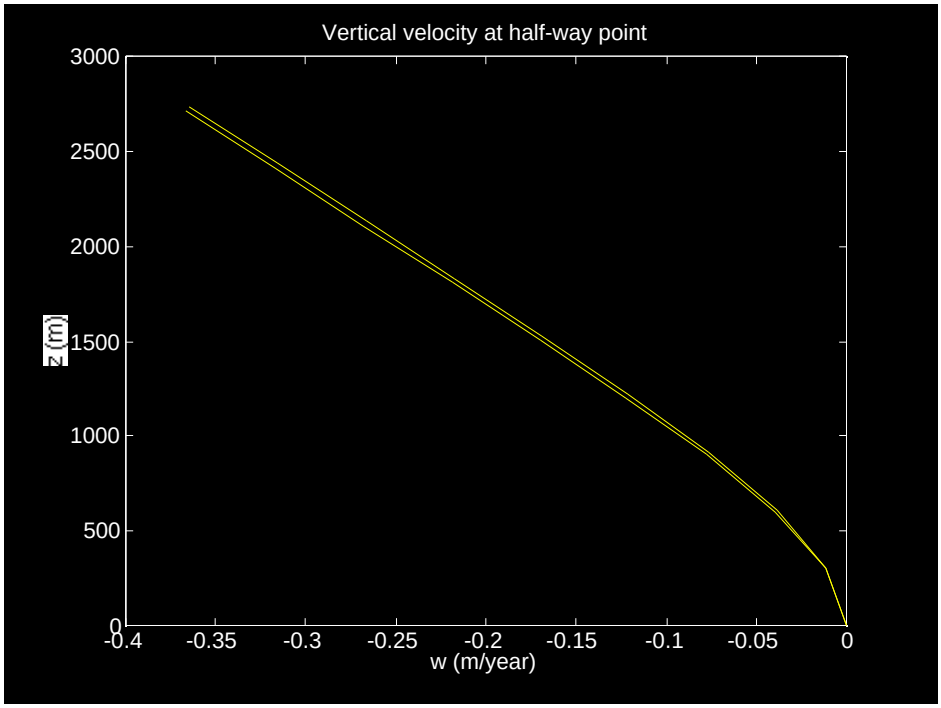

Figure 1.9: Diagnostic vertical velocity (m/yr) as a function of z at the point half-way to between the ice divide and the terminus (finite-difference and exact are shown together).

Part B Solve for steady-state s using an iterative scheme to satisfy the constraint $d = rs^5\left(\frac{\partial s}{\partial r}\right)$. Begin the iteration scheme with a guess $d = .15$. Produce a solution s. Determine a new d using this solution and the expression $d = rs^5\left(\frac{\partial s}{\partial r}\right)$, then repeat until d converges to it's steady state value.

1.5 Milankovitch-Forced Model Runs

Following the EISMINT intercomparison exercise guidelines suggested at the Bremerhaven workshop, we now consider forcing the finite-difference model of the axisymmetric ice sheet with a sinusoidally varying surface accumulation rate. (As mentioned previously, we shall not consider the effects of surface temperature variation. This will be done separately at a later time.) Our exercise is to conduct two experiments, each with a different frequency of accumulation-rate variation:

$$\text{Experiment 1} - \qquad a = 0.3 + 0.2 \sin \frac{2\pi t}{T_1} \quad \text{m/year} \tag{1.44}$$

and

$$\text{Experiment 1} - \qquad a = 0.3 + 0.2 \sin \frac{2\pi t}{T_1} \quad \text{m/year} \tag{1.45}$$

where $T_1 = 2 \times 10^4$ years and $T_2 = 4 \times 10^4$ years. The diagnostic output of each model run will be time-series (one sample every 1,000 years) of the ice-divide surface elevation (ice thickness) $z_s(r = 0, t)$ and of the transect ice "volume" $V(t)$ defined by:

$$V(t) = \int_0^1 z_s(r, t) dr \tag{1.46}$$

In nondimensional variables, the equation we must solve to undertake the two model experiments is written:

$$\frac{\partial s}{\partial t} = A(t) + \frac{1}{r}\frac{\partial}{\partial r}\left(rs^5\left(\frac{\partial s}{\partial r}\right)^3\right) \tag{1.47}$$

where the nondimensional time t is defined using the scale $T = \frac{5L^4}{2A_o(\rho g)^3 Z^7} = \frac{Z}{a} \approx 9189$ years, and $A(t)$ is the nondimensional accumulation rate written in terms of nondimensional time:

$$A(t) = 1.0 + \frac{2}{3}\sin\frac{2\pi t}{T_i} \tag{1.48}$$

with $T_1 = 2.1765$ (nondimensional) and $T_2 = 4.3530$ (nondimensional). To find the solution we use the finite-difference algorithm defined previously in § (1.3.2). The one change is that the right-hand-side-vector $\mathbf{R}$ is defined to accommodate the time-dependent accumulation $A(t)$:

$$\begin{aligned} R_i &= 1.0 + \frac{2}{3}\sin\frac{2\pi t}{T_k} + \frac{s_i^n}{\Delta t} \quad \text{for } i = 2, \ldots, N-1 \; k = 1, 2 \\ R_1 &= 1.0 + \frac{2}{3}\sin\frac{2\pi t}{T_k} + \frac{s_1^n}{\Delta t} \quad \text{for } k = 1, 2 \\ R_N &= 0 \end{aligned}$$

The MATLAB routine used to conduct the two modeling experiments is listed as follows:

```
% Finite-difference solution of mass balance equations in an
% axisymmetric domain:
%
N=16;
g=9.81;
rho=910;
Ao=1/31556926 * 1e-16;
a=0.3/31556926;
L=1500e3/2;
Z=( 5*a*L^ 4/( 2 * Ao * (rho*g)^ 3 ) )^ (1/8);
r=linspace(0,1,N)';
s_exact=( 4 * ( (1/2).^ (4/3) - (r/2).^ (4/3) ) ).^ (3/8);
%
%
T1=2.1765;
T2=2*T1;
nsteps=8000;
```

```
dt=25*31556926*a/Z;
dr=1.0/(N-1);
%
sn=s_exact;
R=zeros(N,1);
d=zeros(N,1);
AU=zeros(N,1);
AD=zeros(N,1);
AL=zeros(N,1);
%
divide=[Z*s_exact(1)];
dummy=0;
for i=1:N-1
dummy=dummy+.5*Z*(s_exact(i)+s_exact(i+1))*dr*L;
end
volume=[dummy];
for n=1:nsteps
t=(n-1)*dt;
phase=2*pi*t/T1;
for i=1:N-1
d(i)=.5*(r(i)+r(i+1))* (.5*(sn(i)+sn(i+1)))^ 5 * ((sn(i+1)-sn(i))/dr)^
2;
end
AD(1)=1/dt+((sn(1)+sn(2))/2)^ 5*((sn(2)-sn(1))/dr)^ 2*4/dr^ 2;
AU(2)=-((sn(1)+sn(2))/2)^ 5*((sn(2)-sn(1))/dr)^ 2*4/dr^ 2;
AD(N)=1;
AL(N-1)=0;
R(1)=1+2/3*sin(phase)+sn(1)/dt;
R(N)=0;
for i=2:N-1
AD(i)= 1/dt + (d(i) + d(i-1))/(r(i)*dr^ 2);
AL(i-1)= -d(i-1)/(r(i)*dr^ 2);
AU(i+1)= -d(i)/(r(i)*dr^ 2);
R(i)=1+2/3*sin(phase)+sn(i)/dt;
end
T=spdiags([AL AD AU],[-1 0 1],N,N);
sn=T\ R;
```

```
if rem(n,40) == 1
divide=[divide
Z*sn(1)];
dummy=0;
for i=1:N-1
dummy=dummy+.5*Z*(sn(i)+sn(i+1))*dr*L;
end
volume=[volume
dummy];
end
end
```

The result of the 20,000-year and 40,000-year Milankovitch cycle experiments are shown in Figs. (1.10) - (1.13).

1.6 Concluding Remarks

We have developed an exact, analytic solution for a steady-state ice sheet of azimuthally symmetric circular plan form. The finite-difference "flowline" model used to determine the time-dependent evolution of such an ice sheet has been compared with the analytic solution and found to be satisfactory. (Differences on the order of a percent or two are unaccounted for; I suspect that they arise from the discretization of the ice divide boundary condition and from numerical inaccuracy at the terminus.)

Computer Comments

Some of the time-stepping simulations presented in this chapter were conducted on a Macintosh IIfx computer with a 40Mhz 68030 chip and FPU using Matlab 4.1. The 200,000-year run of the 20,000-year Milankovitch cycle took approximately 4000 seconds of CPU time. Computer memory was not an issue (storage of the tridiagonal array required only 16×3 words

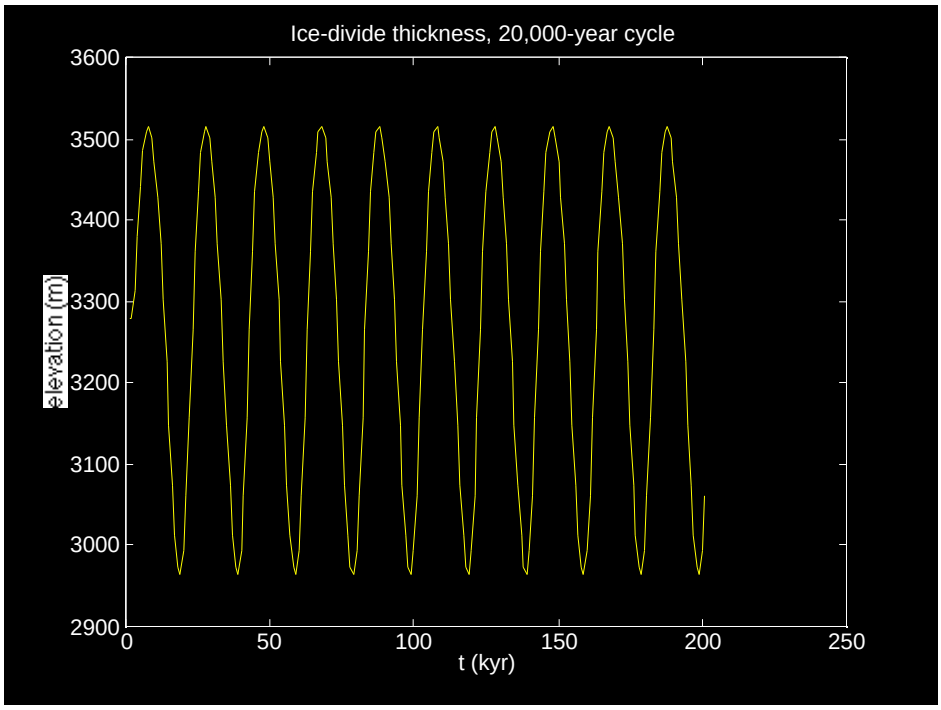

Figure 1.10: Ice-divide surface elevation, $z_s(r = 0)$ (m) for a climate cycle of 20,000 years.

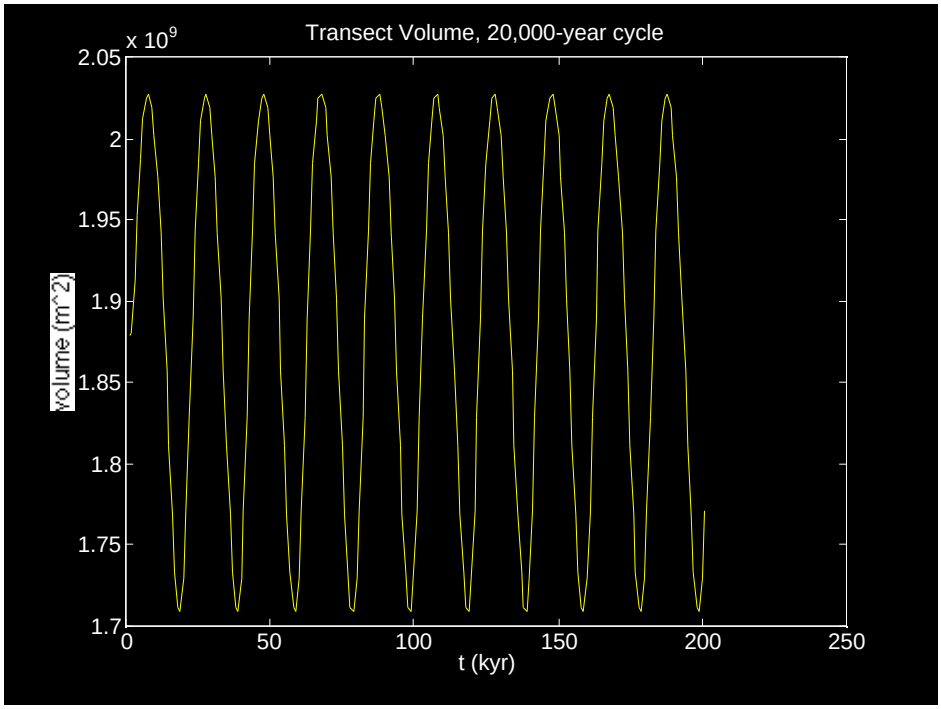

Figure 1.11: Transect volume (m^2) for a climate cycle of 20,000 years.

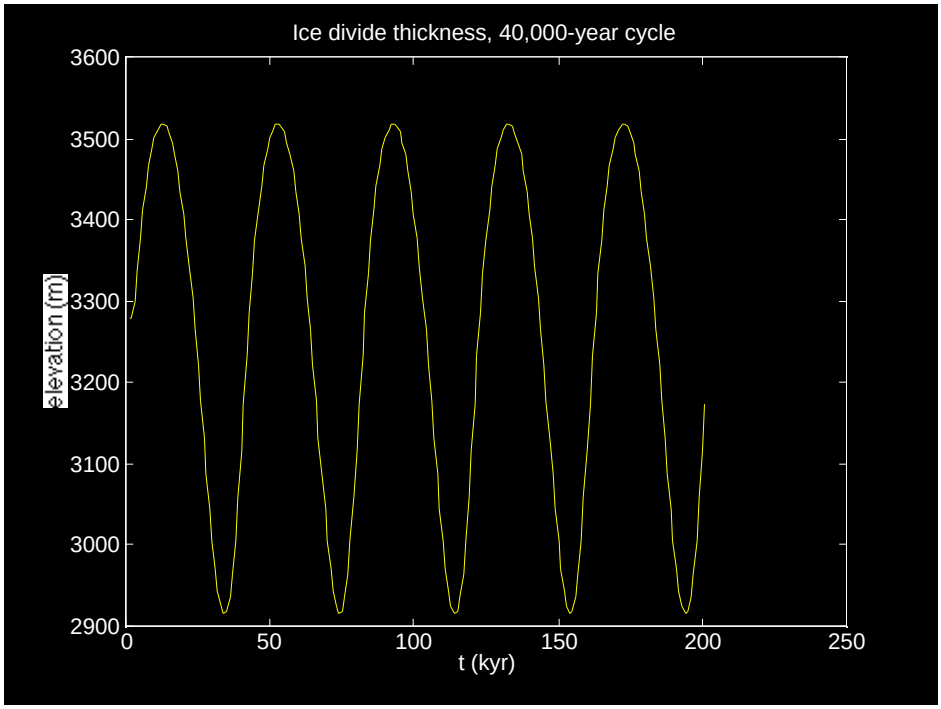

Figure 1.12: Transect volume (m^2) for a climate cycle of 20,000 years.

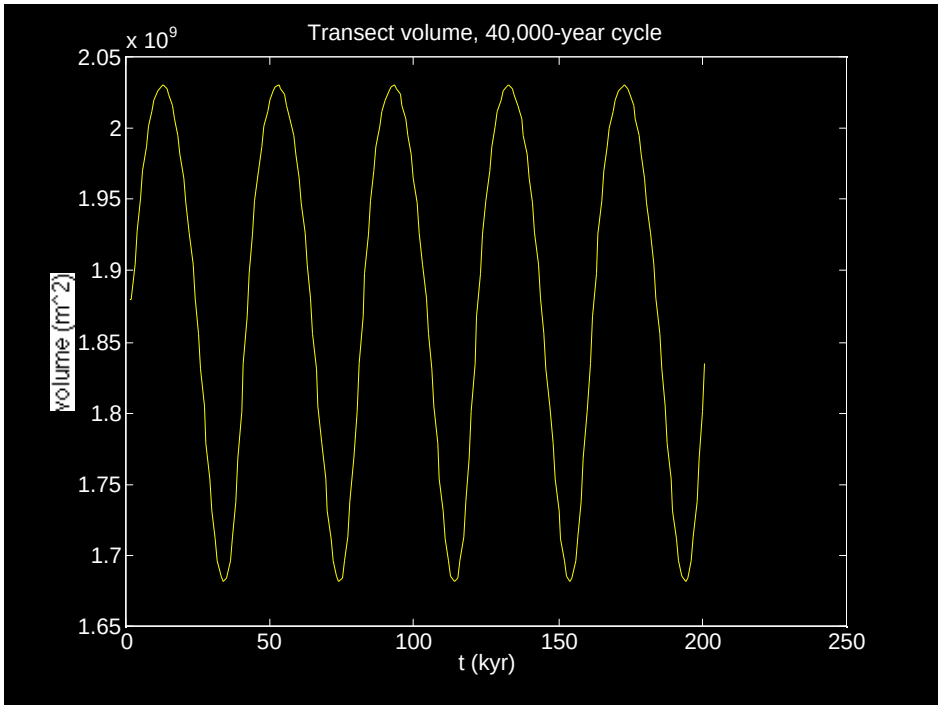

Figure 1.13: Transect volume (m^2) for a climate cycle of 20,000 years.

of memory. A MACINTOSH Quadra 650 was later used and found to be approximately 5 times faster than the MACINTOSH IIfx.

Chapter 2

Level 1: Two-Dimensional Ice-Sheet Model

In this chapter, we shall construct a two-dimensional ice-sheet model using *both* the finite-difference and the finite-element methods. Our goal will be to compare the techniques, programming aspects, and numerical accuracy of the two methods in the context of simulating an ice-sheet with square planform such as suggested by the Level 1 intercomparison test (Bremerhaven, 1994). One important digression will be highlighted in this chapter. This concerns the computational aspects of matrix construction, storage, factorization and backsubstitution. We shall learn that modern sparse matrix algebra, much of which is readily available in MATLAB (Gilbert, Moler and Schreiber, 1992), offers significant computational efficiencies not previously recognized.

2.1 Finite-Difference Model

The nondimensional governing equation for mass balance of the idealized grounded ice sheet we wish to simulate is the same as that derived in the

previous chapter

$$\frac{\partial s}{\partial t} = 1 + \nabla \cdot \left((\nabla s)^3 \, s^5 \right) \tag{2.1}$$

For convenience, we define an effective diffusivity $d = (\nabla s \cdot \nabla s) s^5$ to render the above equation into a form that is easily linearized:

$$\frac{\partial s}{\partial t} = 1 + \nabla \cdot (d \nabla s) \tag{2.2}$$

where, as spelled out in the previous chapter, all variables are nondimensional using the scaling convention outlined in § (1.1).

2.1.1 Implicit time-stepping with a staggered grid

Following finite-difference conventions [*e.g.*, Waddington, 1981], we adopt a staggered finite-difference scheme where variables s and d are defined as shown in Fig. (2.1). The discrete form of Eqn. (2.1) is composed of the following parts:

$$\begin{aligned} d^n_{i,j} &= \left(\frac{1}{4} \left(s^n_{i,j} + s^n_{i+1,j} + s^n_{i+1,j+1} + s^n_{i,j+1} \right) \right)^5 \\ &\times \frac{1}{4\Delta^2} \Big[\left(s^n_{i+1,j} - s^n_{i,j} + s^n_{i+1,j+1} - s^n_{i,j+1} \right)^2 \\ &+ \left(s^n_{i,j+1} - s^n_{i,j} + s^n_{i+1,j+1} - s^n_{i+1,j} \right)^2 \Big] \end{aligned} \tag{2.3}$$

$$\begin{aligned} \frac{\partial}{\partial x} \left(d \frac{\partial s}{\partial x} \right) \Big|_{i,j} &= \frac{1}{2\Delta^2} \Big[\left(d^n_{i,j} + d^n_{i,j-1} \right) \left(s^{n+1}_{i+1,j} - s^{n+1}_{i,j} \right) \\ &- \left(d^n_{i-1,j} + d^n_{i-1,j-1} \right) \left(s^{n+1}_{i,j} - s^{n+1}_{i-1,j} \right) \Big] \end{aligned} \tag{2.4}$$

$$\begin{aligned} \frac{\partial}{\partial y} \left(d \frac{\partial s}{\partial y} \right) \Big|_{i,j} &= \frac{1}{2\Delta^2} \Big[\left(d^n_{i,j} + d^n_{i-1,j} \right) \left(s^{n+1}_{i,j+1} - s^{n+1}_{i,j} \right) \\ &- \left(d^n_{i,j-1} + d^n_{i-1,j-1} \right) \left(s^{n+1}_{i,j} - s^{n+1}_{i,j-1} \right) \Big] \end{aligned} \tag{2.5}$$

where Δ is the grid spacing (assumed to be the same in the two spatial directions), subscripts denote the grid point where the designated variables

are evaluated, and the superscripts n and $n+1$ denote the time-step level at which the designated variables are evaluated. Convention dictates that variables at time step n are considered known (possibly from the specification of an initial condition) and variables at time step $n+1$ are unknown. The goal of the finite-difference model is to determine the variables at time step $n+1$ in an iterative fashion so to accomplish a time marching.

The above finite-difference parts are put together to yield the following finite-difference equation for the $s_{i,j}^{n+1}$'s (implicit time step):

$$\begin{aligned} s_{i,j}^{n+1}\left[\frac{1}{\Delta t}+\frac{1}{\Delta^2}\left(d_{i,j}^n+d_{i-1,j}^n+d_{i,j-1}^n+d_{i-1,j-1}^n\right)\right] & \\ +s_{i,j+1}^{n+1}\left[\frac{-1}{2\Delta^2}\left(d_{i,j}^n+d_{i-1,j}^n\right)\right] & \\ +s_{i,j-1}^{n+1}\left[\frac{-1}{2\Delta^2}\left(d_{i,j-1}^n+d_{i-1,j-1}^n\right)\right] & \\ +s_{i+1,j}^{n+1}\left[\frac{-1}{2\Delta^2}\left(d_{i,j}^n+d_{i,j-1}^n\right)\right] & \\ +s_{i-1,j}^{n+1}\left[\frac{-1}{2\Delta^2}\left(d_{i-1,j}^n+d_{i-1,j-1}^n\right)\right] &= 1+\frac{s_{i,j}^n}{\Delta t} \end{aligned} \quad (2.6)$$

The above equation may be conveniently expressed in matrix notation as follows:

$$\mathbf{A}\mathbf{s}^{n+1}=\mathbf{R} \quad (2.7)$$

where $\mathbf{s}^n$ is the column vector composed of the values of $s_{i,j}^n$ arranged by the order in which the grid points are numbered. Thus, if grid point (i,j) is numbered $\gamma_{i,j}$ where $\gamma_{i,j}$ is an integer in the interval $[1, 31^2]$, the p'th element of $\mathbf{s}^n$ is the value of s^n at the grid point (i,j) who's $\gamma_{i,j}$ is equal to p. The matrix element A_{pq} describes how the solution at grid point $\gamma_{i,j}=p$ depends on the solution at grid point $\gamma_{i,j}=q$. The matrix-construction algorithm may be summarized from the finite-difference version of the problem described above as follows:

$$s_{i,j}^{n+1}\left[\frac{1}{\Delta t}+\frac{1}{\Delta^2}\left(d_{i,j}^n+d_{i-1,j}^n+d_{i,j-1}^n+d_{i-1,j-1}^n\right)\right] \;\rightarrow\; A_{\gamma_{i,j},\gamma_{i,j}}$$

$$
\begin{aligned}
s^{n+1}_{i,j+1}\left[\frac{-1}{2\Delta^2}\left(d^n_{i,j}+d^n_{i-1,j}\right)\right] &\rightarrow A_{\gamma_{i,j},\gamma_{i,j+1}} \\
s^{n+1}_{i,j-1}\left[\frac{-1}{2\Delta^2}\left(d^n_{i,j-1}+d^n_{i-1,j-1}\right)\right] &\rightarrow A_{\gamma_{i,j},\gamma_{i,j-1}} \\
s^{n+1}_{i+1,j}\left[\frac{-1}{2\Delta^2}\left(d^n_{i,j}+d^n_{i,j-1}\right)\right] &\rightarrow A_{\gamma_{i,j},\gamma_{i+1,j}} \\
s^{n+1}_{i-1,j}\left[\frac{-1}{2\Delta^2}\left(d^n_{i-1,j}+d^n_{i-1,j-1}\right)\right] &\rightarrow A_{\gamma_{i,j},\gamma_{i-1,j}} \\
1+\frac{s^n_{i,j}}{\Delta t} &\rightarrow R_{\gamma_{i,j}} \qquad (2.8)
\end{aligned}
$$

These "matrix-stuffing" conventions apply for grid points that are not on the boundaries of the numerical domain. For the boundaries of the EISMINT exercise, we apply the simple condition:

$$A_{\gamma_{i,j},\gamma_{i,j}} = 1 \quad \text{for } i = 1, i_{max}\ j = 1, j_{max} \qquad (2.9)$$

with all other elements in rows of $\mathbf{A}$ corresponding to these boundary grid points being zero. We also have, for the boundary nodes

$$R_{\gamma_{i,j}} = 0 \quad \text{for } i = 1, i_{max}\ j = 1, j_{max} \qquad (2.10)$$

The time stepping model ice-sheet model is thus described by solution of the linear equation $\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R}$ for as many time steps as desired starting from a specified initial condition $\mathbf{s}^0$. After the solution of the linear equation at each time step, the matrix $\mathbf{A}$ and the right-hand-side vector $\mathbf{R}$ must be reconstructed using updated values of the $d^{n+1}_{i,j}$'s and the $s^{n+1}_{i,j}$'s.

2.1.2 A digression about sparse matrices

Before using the above-described finite-difference model, we must consider the fact that the matrix $\mathbf{A}$ can be very large, and may stress the limits of computer memory just for it's storage. For the EISMINT exercise, the 31×31 finite-difference grid possesses $31^2 = 961$ grid points. The matrix $\mathbf{A}$ thus has 961 rows and 961 columns; or $961^2 = 923,521$ elements (almost a million!). The solution of the linear equation $\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R}$ may thus exceed

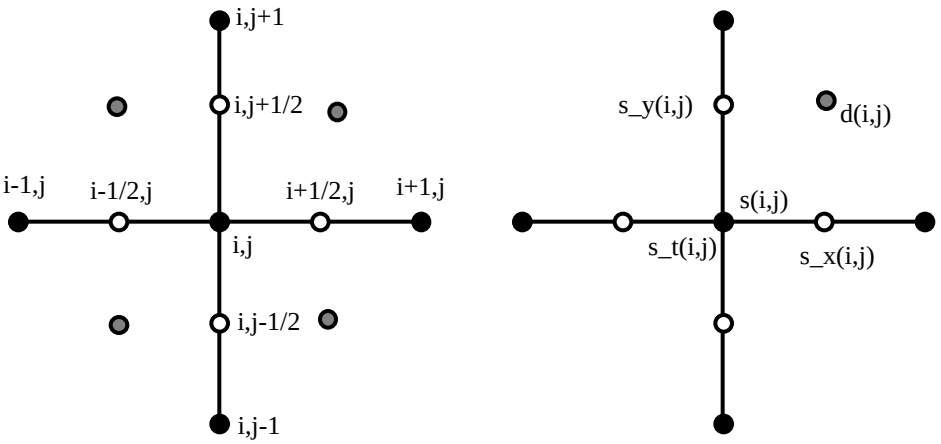

Figure 2.1: staggered grid scheme associated with finite-difference version of a 2-d ice-sheet model.

the performance capabilities of even the most advanced scientific computer for even the smallest ice-sheet modelling experiment (such as the EISMINT exercise).

Two approaches may be taken to overcome the difficulty imposed by the vast size of **A**. The first is to use an iterative "relaxation" scheme to find the solution of the equation without ever having to store or factor the matrix **A**. This is the approach Huybrechts [1992] used to investigate the evolution of the Antarctic ice sheet, for example. Today, there are many modern iterative techniques that are available "off the shelf" from software developers. The most efficient may be the MUDPACK (multigrid iteration package) available from the NCAR software development team (consult the NCAR software distribution specialist by email: consult1ncar.ucar.edu). We will examine several iterative techniques (Gauss-Seidel and ADI) and compare them with direct matrix algorithms in the exercise at the end of this section.

The other approach to the difficulty of the size of **A** is to take advantage of the fact that **A** has very few nonzero elements. It is instructive to investigate the matrix associated with the 31 by 31 grid of the EISMINT exercise in detail using the MATLAB routines available for this purpose. We first construct the symmetric adjacency matrix **C** which has only 1's and 0's as its elements. The elements which have 1's indicate that the grid points corresponding to the row number and the column number, respectively, are connected. The matrix **A** will have zeros in the same locations as the zeros of **C**. We construct the adjacency matrix for the EISMINT exercise using the following MATLAB algorithm:

```
% This program determines optimal node numbering of a 2-D ice sheet
%
imax=31;
jmax=31;
nodes=imax*jmax;
node=zeros(imax,jmax);
Adj=zeros(nodes,nodes);
xy=zeros(node,2);
counter=0;
for j=1:jmax
```

```
for i=1:imax
counter=counter+1;
node(i,j)=counter;
end
end
% Construct the adjacency matrix
for j=2:jmax-1
for i=2:imax-1
Adj(node(i,j),node(i,j))=1;
Adj(node(i,j),node(i+1,j))=1;
Adj(node(i,j),node(i-1,j))=1;
Adj(node(i,j),node(i,j+1))=1;
Adj(node(i,j),node(i,j-1))=1;
end
end
j=1;
for i=2:imax-1
Adj(node(i,j),node(i,j))=1;
Adj(node(i,j),node(i+1,j))=1;
Adj(node(i,j),node(i-1,j))=1;
Adj(node(i,j),node(i,j+1))=1;
end
j=jmax;
for i=2:imax-1
Adj(node(i,j),node(i,j))=1;
Adj(node(i,j),node(i+1,j))=1;
Adj(node(i,j),node(i-1,j))=1;
Adj(node(i,j),node(i,j-1))=1;
end
i=1;
for j=2:imax-1
Adj(node(i,j),node(i,j))=1;
Adj(node(i,j),node(i,j+1))=1;
Adj(node(i,j),node(i,j-1))=1;
Adj(node(i,j),node(i+1,j))=1;
end
i=imax;
```

```
for j=2:imax-1
Adj(node(i,j),node(i,j))=1;
Adj(node(i,j),node(i,j+1))=1;
Adj(node(i,j),node(i,j-1))=1;
Adj(node(i,j),node(i-1,j))=1;
end
Adj(node(1,1),node(1,1))=1;
Adj(node(1,1),node(1,2))=1;
Adj(node(1,1),node(2,1))=1;
Adj(node(1,jmax),node(1,jmax))=1;
Adj(node(1,jmax),node(2,jmax))=1;
Adj(node(1,jmax),node(1,jmax-1))=1;
Adj(node(imax,jmax),node(imax,jmax))=1;
Adj(node(imax,jmax),node(imax,jmax-1))=1;
Adj(node(imax,jmax),node(imax-1,jmax))=1;
Adj(node(imax,1),node(imax,1))=1;
Adj(node(imax,1),node(imax,2))=1;
Adj(node(imax,1),node(imax-1,1))=1;
% Construct grid-point coordinate map:
delta=2/(imax-1);
for j=1:jmax
for i=1:imax
xy(node(i,j),:)  = [(i-1)*delta (j-1)*delta];
end
end
% Construct mesh plot
gplot(Adj,xy);
pause
% Look at sparseness structure
spy(Adj)
% Determine matrix density
nnz(Adj)/nodes^ 2
```

The sparse structure of $\mathbf{C}$ (called `Adj` in the above MATLAB routine) is emphasized by the fact that there are only 4681 nonzero entries in a matrix of almost a million elements. The MATLAB function `gplot` displays the finite-difference grid and gives Fig. (2.2). The silhouette of the adjacency

matrix is generated by the MATLAB function `spy()`, and is shown in Fig. (2.3). As can be readily appreciated from the silhouette, $\mathbf{C}$ is extremely sparse.

An alternative to storing $\mathbf{C}$ or, for that matter, $\mathbf{A}$ in full matrix format is to store just the nonzero entries of $\mathbf{C}$ or $\mathbf{A}$. The MATLAB native sparse matrix storage, for example, allows the definition of a sparse matrix using three column vectors (not described explicitly here, consult the MATLAB user's guide for more information). Two column vectors are required to store the row numbers and column numbers of the nonzero entries. The third column vector is used to store the actual matrix values that enter into the nonzero entries. Using sparse matrix storage format, the 961×961 (923,521) words of computer memory needed to store $\mathbf{A}$ (or $\mathbf{C}$) is reduced to 3×4681 (14043) words of computer memory. The sparse matrix format requires only a little over 1.5% of the memory used by the full matrix format.

For the finite-difference and finite-element models constructed here, we will use sparse matrix storage for the matrix $\mathbf{A}$. In addition, we shall consider a node-numbering scheme which minimizes the number of floating point operations necessary to factor $\mathbf{A}$ to produce the solution to the equation $\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R}$. We shall take up the subject of node-numbering schemes in the following digression.

Node-numbering schemes: Does it make a difference?

There are two approaches that can be taken to solve a matrix equation $\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R}$. One approach is to compute the LU-factorization of $\mathbf{A}$, and to then perform a forward substitution and a back substitution step to determine $\mathbf{s}^{n+1}$. The other approach is to compute the Cholesky factorization of $\mathbf{A}$, and then to perform two triangular matrix solves on $\mathbf{R}$ to obtain $\mathbf{s}^{n+1}$. For the finite-difference scheme associated with the ice-sheet model derived here, the matrix $\mathbf{A}$ is sparse, symmetric and positive definite. In this situation the latter approach will work most efficiently (with fewest floating point operations, abbreviated as FLOPS). In both approaches, the most computationally intensive (and expensive) step is the factorization step. We may thus concentrate on the question of whether one node-numbering scheme over

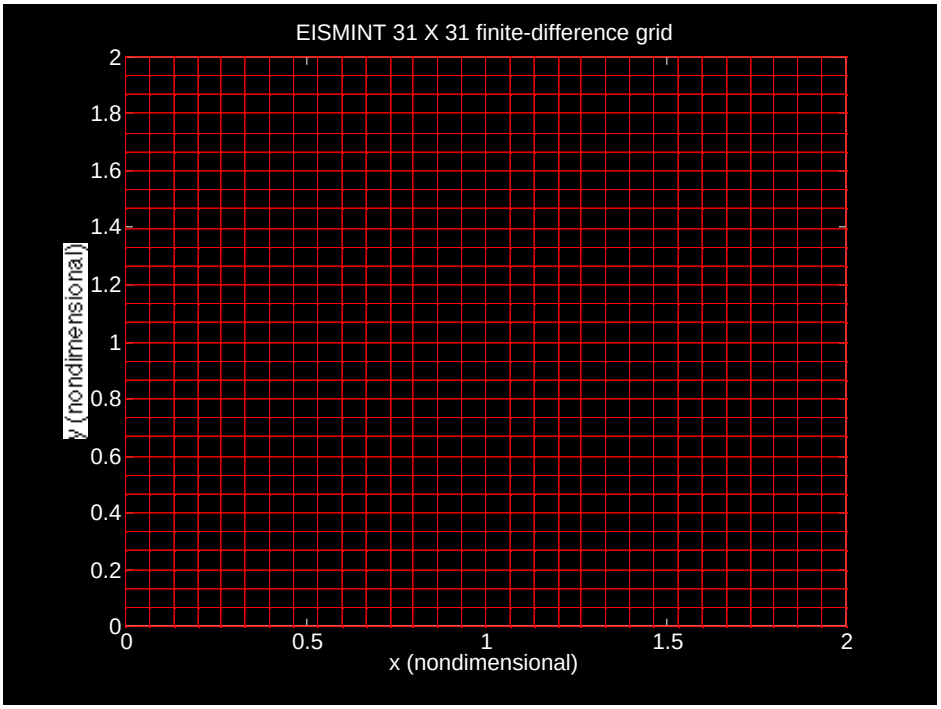

Figure 2.2: The 31 by 31 finite-difference grid used in the EISMINT ice-sheet modelling exercise.

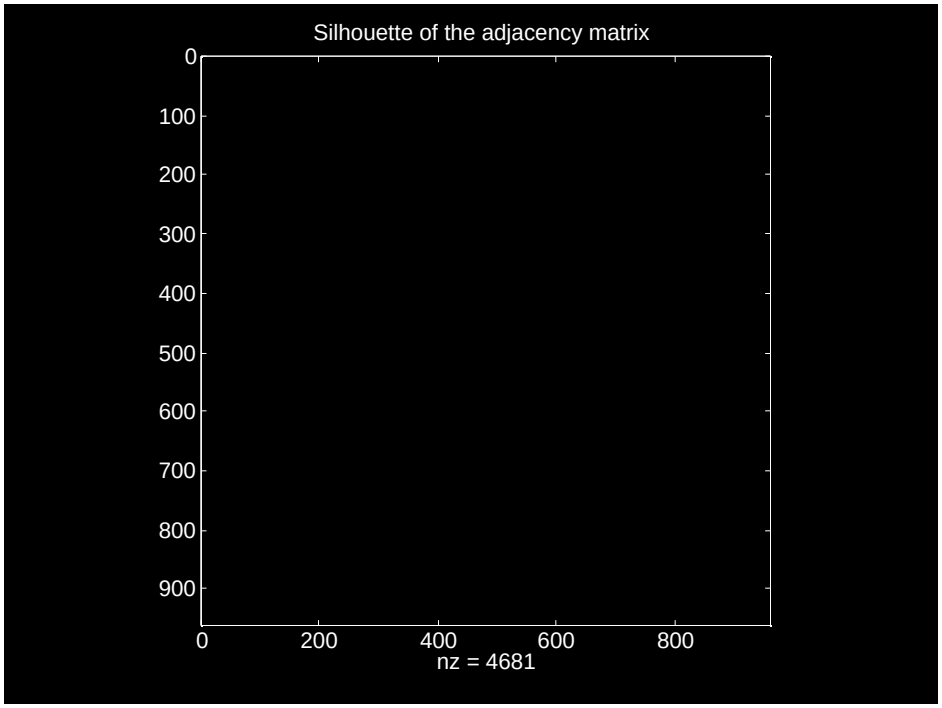

Figure 2.3: The silhouette of the adjacency matrix generated by the 31 by 31 finite-difference grid used in the EISMINT ice-sheet modelling exercise. Notice that non-zero elements are very rare and are crowded along the main diagonal of the matrix.

another might make the factorization step more efficient.

To investigate the computational demands required to factorize **A** for various node-numbering schemes, we can compute the number of FLOPS required to do either the LU or the Cholesky factorizations of **C**, the adjacency matrix. (It is worth remarking that the factorization of the adjacency matrix is of no practical relevance to ice-sheet physics. What counts is that the number of FLOPS necessary to factorize **C** is the same as that for **A**.) In order to make **C** positive definite, we must replace its diagonal of 1's with a diagonal of, say, 10's. We shall perform these factorizations (counting up the FLOPS for each) using the sparse matrix storage convention described above and three different numbering schemes.

The first numbering scheme is the "plain vanilla" scheme that comes from counting the grid points consecutively in row-order or column-order format. The second numbering scheme is referred to as the "reversed Cuthill-McKee" scheme. This scheme produces a silhouette of **C** which has minimum bandwidth (non-zero elements of **C** are crowded most "tightly" along the main diagonal). The third scheme is referred to as the "minimum degree" scheme; and this scheme produces a "fractal" looking silhouette which optimizes the size of continuous blocks of zeros. We shall see that the reversed Cuthill-McKee ordering minimizes the FLOPS for the LU decomposition, and that the minimum-degree ordering minimizes the FLOPS for the Cholesky factorization. For illustration, the silhouettes of **C** corresponding to these two ordering schemes are shown in Figs. (2.4) and (2.5). The two improved numbering schemes are too difficult to explain here, I thus refer the reader to the MATLAB user's guide for further description.

As a benchmark, we note the fact that the computational workload required to perform the LU decomposition of **C** when **C** is stored in full matrix format is about 591,669,120 FLOPS. In this circumstance, it would take about 10 CPU seconds of a Cray YMP processor to accomplish one time step of the 31 by 31 point finite-difference model. Clearly, the problem cannot be done without sparse matrix storage schemes (or the iterative technique mentioned above). The following table lists the FLOPS required to factor **C** given the various node-numbering schemes:

	LU	Cholesky

row − order	1,821,702	943,451
reversed Cuthill − McKee	1,352,162	522,226
minimum degree	4,143,152	248,306

The above data suggests that, for the ice-sheet modelling problem, the most efficient way to conduct a time step is to use minimum-degree grid point numbering coupled with Cholesky factorization. A factor of almost 10 improvement is seen over the row-order numbering coupled with LU decomposition. Using a Cray YMP that can vectorize the Cholesky factorization (obtaining approximately 80 MFLOPS per second), approximately 400 time steps of the 31 by 31 finite-difference model can be accomplished in one CPU second. This represents a 4000 to 1 improvement over LU decomposition with the full-storage convention. It is not clear, however, whether these improvements are comparable to those achieved using iterative relaxation techniques mentioned above. The one sure advantage of the "direct solution" approach taken here is that it is less complicated (one does not have to consider convergence criteria necessary to ensure that the iterative relaxation technique converges to an accurate answer).

2.1.3 Exercise 1

Solve the problem

$$\nabla^2 u - 1 = 0 \tag{2.11}$$

in the square domain $[0 < x < 1, 0 < y < 1]$, with $u = 0$ boundary conditions. Refer to the instructor's lecture, or to texts on the subject of relaxation methods, to obtain the algorithmic details of each of the methods cited below. Keep track of the FLOPS needed to acquire the solution in each part and compare the various methods when you have completed the exercise. Use an 11×11 grid.

Part A Solve the problem using Gauss relaxation.

Part B Solve the problem using Gauss-Seidel relaxation.

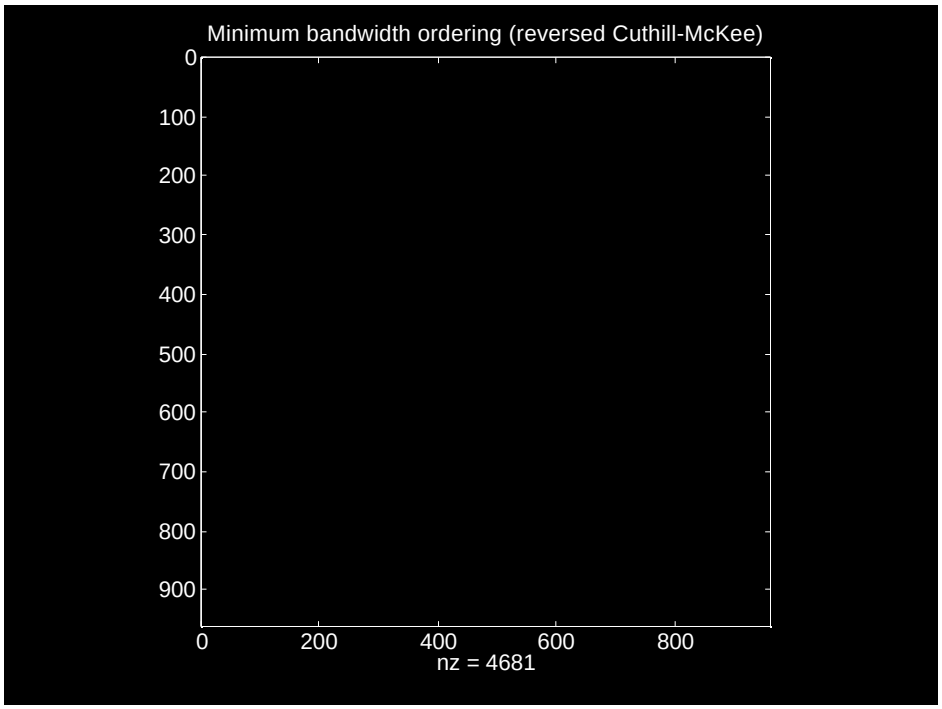

Figure 2.4: The silhouette of the adjacency matrix generated by the 31 by 31 finite-difference grid used in the EISMINT ice-sheet modelling exercise and the reversed Cuthill-McKee numbering convention. Notice that the "bandwidth" of the matrix has been reduced over that displayed in Fig. (2.3).

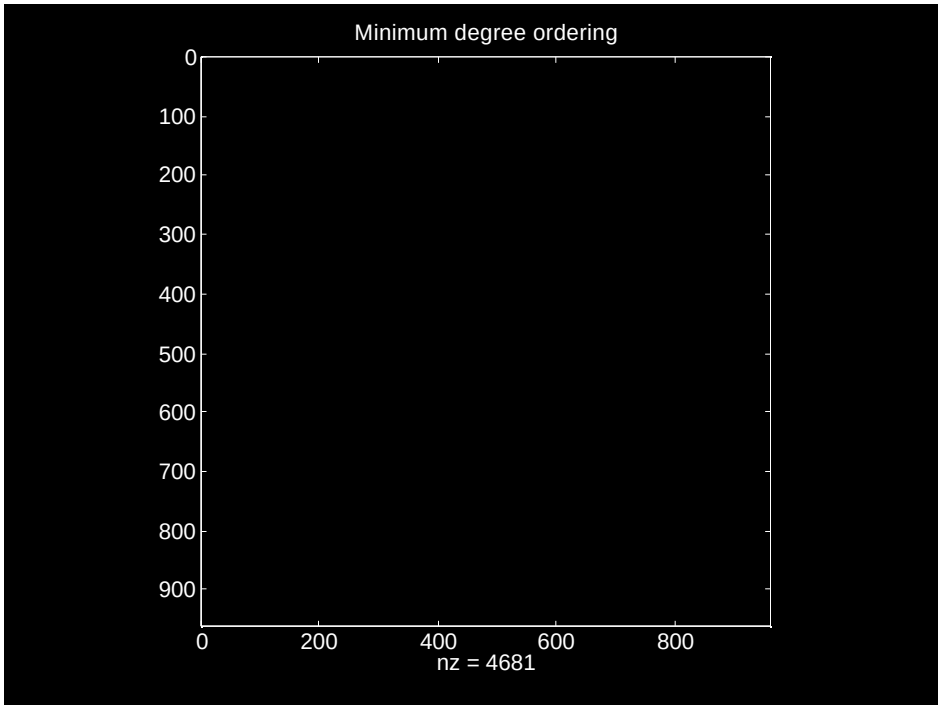

Figure 2.5: The silhouette of the adjacency matrix generated by the 31 by 31 finite-difference grid used in the EISMINT ice-sheet modelling exercise and the minimum-degree numbering convention. Notice that the "fractal" appearance of the matrix silhouette. The advantage of the minimum-degree ordering scheme is that large blocks of zeros are produced. These large blocks are preserved through the Cholesky factorization. The minimum-degree ordering scheme can sometimes reduces the number of floating point operations necessary to solve the matrix equation $\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R}$ when $\mathbf{A}$ is symmetric and positive definite, and when Cholesky factorization is used in the solution procedure.

Part C Solve the problem using Alternate Direction Implicit (ADI) relaxation.

Part D Solve the problem using Sparse matrix algebra without relaxation.

My effort in this exercise yielded the following result:

	FLOPS	Residual Error
ADI	$63,918$	0.044
Sparse Matrix	$13,527$	6.217×10^{-15}
Full Matrix	$1,272,151$	1.176×10^{-14}

ADI is preferable to other forms of relaxation and to full matrix implementation without relaxation. However, if sparse matrix algebra is available to the user, a non-relaxation technique with sparse-matrix algebra is preferred.

2.2 Finite Difference Solution: Steady-State Ice-Sheet Experiment

We construct the finite-difference model using the finite-difference conventions defined above, and integrate the model for 50,000 years starting from a zero ice thickness to generate a steady state. The results of the integration are shown in Figs. (2.6) - (2.8). The size of the final, steady-state ice sheet in this exercise is slightly larger than that of the steady-state ice sheet of circular planform developed in the previous chapter. This is expected because the square planform represents a greater area over which the ice sheet accumulates snow. The dimensional form of the surface elevation transect

from the ice divide to the right margin is listed as follows:

$$(z_s)_{i=16,31;j=16} = 10^3 \times \begin{bmatrix} 3.4218 \\ 3.3911 \\ 3.3398 \\ 3.2772 \\ 3.2044 \\ 3.1212 \\ 3.0273 \\ 2.9217 \\ 2.8027 \\ 2.6679 \\ 2.5138 \\ 2.3346 \\ 2.1204 \\ 1.8521 \\ 1.4807 \\ 0.0000 \end{bmatrix} \quad \text{m} \tag{2.12}$$

The ice-divide surface elevation is 3421.8 m.

The MATLAB script which performs the finite-difference calculation is listed below. (Since we are using MATLAB we don't have to explicitly remember to use minimum-degree ordering and Cholesky factorization. This is done automatically by MATLAB . In FORTRAN, however, we must be careful to remember the benefits of a careful choice of numbering scheme and factorization technique.)

```
% This script represents a finite-difference model
% of a 2-D ice sheet
%
N=16;
imax=31;
jmax=31;
g=9.81;
rho=910;
Ao=1/31556926 * 1e-16;
a=0.3/31556926;
```

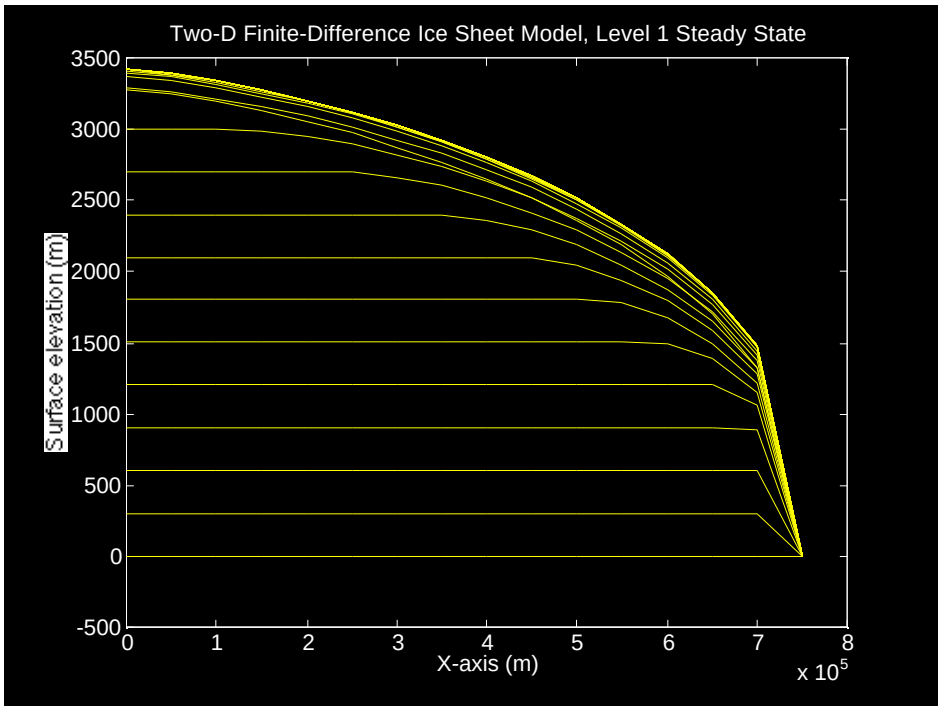

Figure 2.6: Transect of surface elevation (m) from ice divide to right margin of the two-dimensional ice sheet. Each line represents the state of the ice-sheet surface at 1,000-year intervals. The initial condition was zero ice thickness and the accumulation rate was constant. The asterisks denote the exact, analytic profile deduced in the previous chapter. The comparison between the asterisks and the final, steady-state profile of the finite difference model (at 50,000 years) suggests that the ice sheet with square planform has slightly greater volume than the azimuthally symmetric ice sheet of circular planform.

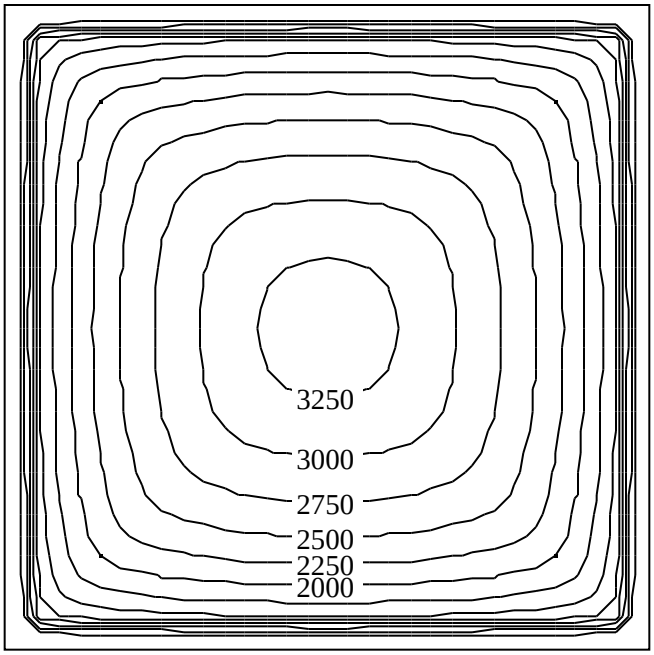

Figure 2.7: Contour map (CI=250 m) of the two-dimensional ice sheet produced after 50,000 years of evolution toward steady state.

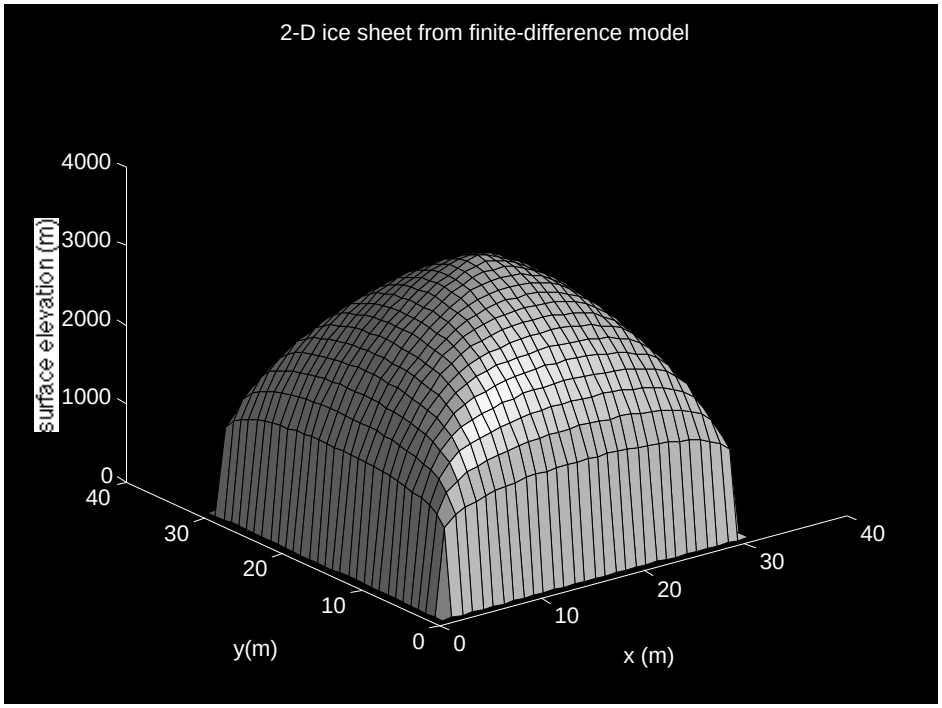

Figure 2.8: Surface of the ice sheet in steady state (50,000 years).

```
L=1500e3/2;
Z=( 5*a*L^ 4/( 2 * Ao * (rho*g)^ 3 ) )^ (1/8);
r=linspace(0,1,N)';
s_exact=( 4 * ( (1/2).^ (4/3) - (r/2).^ (4/3) ) ).^ (3/8);
plot(L*r,Z*s_exact); hold on
pause
nsteps=5000;
dt=10*31556926*a/Z;
sn=zeros(imax,jmax);
s=zeros(nodes,1);
nodes=imax*jmax;
node=zeros(imax,jmax);
row=zeros(4325,1);
col=zeros(4325,1);
value=zeros(4325,1);
delta=2/(imax-1);
R=zeros(nodes,1);
d=zeros(imax,jmax);
%
counter=0;
for j=1:jmax
for i=1:imax
counter=counter+1;
node(i,j)=counter;
end
end
%
for n=1:nsteps
%
for j=1:jmax-1
for i=1:imax-1
d(i,j)=(1/4*(sn(i,j)+sn(i+1,j)+sn(i+1,j+1)+sn(i,j+1)))^ 5 ...
1/(4*delta^ 2)*( ...
(sn(i+1,j)-sn(i,j)+sn(i+1,j+1)-sn(i,j+1))^ 2 ...
+(sn(i,j+1)-sn(i,j)+sn(i+1,j+1)-sn(i+1,j))^ 2);
end
end
```

```
% % Construct the finite-difference stiffness matrix
count=0;
for j=2:jmax-1
for i=2:imax-1
%
count=count+1;
row(count)=node(i,j);
col(count)=node(i,j);
value(count)= 1/dt+1/delta^ 2*(d(i,j)+d(i-1,j)+d(i,j-1)+d(i-1,j-1));
%
count=count+1;
row(count)=node(i,j);
col(count)=node(i,j+1);
value(count)=-1/(2*delta^ 2)*(d(i,j)+d(i-1,j));
%
count=count+1;
row(count)=node(i,j);
col(count)=node(i,j-1);
value(count)=-1/(2*delta^ 2)*(d(i,j-1)+d(i-1,j-1));
%
count=count+1;
row(count)=node(i,j);
col(count)=node(i+1,j);
value(count)=-1/(2*delta^ 2)*(d(i,j)+d(i,j-1));
%
count=count+1;
row(count)=node(i,j);
col(count)=node(i-1,j);
value(count)=-1/(2*delta^ 2)*(d(i-1,j)+d(i-1,j-1));
%
R(node(i,j))=1+sn(i,j)/dt;
end
end
%
j=1;
for i=2:imax-1
count=count+1;
```

```
row(count)=node(i,j);
col(count)=node(i,j);
value(count)=1;
R(node(i,j))=0;
end
%
j=jmax;
for i=2:imax-1
count=count+1;
row(count)=node(i,j);
col(count)=node(i,j);
value(count)=1;
R(node(i,j))=0;
end
%
i=1;
for j=2:imax-1
count=count+1;
row(count)=node(i,j);
col(count)=node(i,j);
value(count)=1;
R(node(i,j))=0;
end
%
i=imax;
for j=2:imax-1
count=count+1;
row(count)=node(i,j);
col(count)=node(i,j);
value(count)=1;
R(node(i,j))=0;
end
count=count+1;
row(count)=node(1,1);
col(count)=node(1,1);
value(count)=1;
R(node(1,1))=0;
```

```
count=count+1;
row(count)=node(1,jmax);
col(count)=node(1,jmax);
value(count)=1;
R(node(1,jmax))=0;
count=count+1;
row(count)=node(imax,jmax);
col(count)=node(imax,jmax);
value(count)=1;
R(node(imax,jmax))=0;
count=count+1;
row(count)=node(imax,1);
col(count)=node(imax,1);
value(count)=1;
R(node(imax,1))=0;
% Construct sparse matrix
A=sparse(row,col,value);
% Cholesky factor and solve (automatic by Matlab)
s=A\ R;
for j=1:jmax
for i=1:imax
sn(i,j)=s(node(i,j));
end
end
if rem(n,100) == 1
plot(L*r,Z*sn(16:31,16));
end
%
end % end time step loop
```

2.3 Finite-Element Model

Our next task is to develop a finite-element model of the same problem completed above. Our motivation for introducing the finite-element method is

twofold. First, we wish to emphasize the accessibility and understandability of finite-element methods. Second, we wish to introduce the finite-element method at a stage where it can readily be compared with the finite-difference method.

The key difference between the finite-element and finite-difference approaches is the starting point. In finite differences, we insist that the partial-differential equation representing mass continuity (Eqn. 2.2) be satisfied at discrete grid points (i, j) which, for the EISMINT exercise, are located on a 31×31 grid. In finite elements, we insist on something different: that the partial-differential equation be satisfied in an *integral sense* everywhere. In other words, we begin the finite-element approach by applying the following condition

$$\int_0^2 \int_0^2 \left\{ \frac{\partial s}{\partial t} - \nabla \cdot (d \nabla s) - 1 \right\} W(x, y) dx dy = 0 \tag{2.13}$$

for *all* arbitrary functions $W(x, y)$ that satisfy the mathematical properties we specify. (For the present, we shall only require the W's to be continuous. As before, we define the effective diffusivity by $d = s^5(\nabla s \cdot \nabla s)$. In future chapters, we may restrict the functional properties of the W's to reflect some need such as "upstream" differencing.) The functions $W(x, y)$ are referred to as "weighting functions" and are unspecified at this stage. The meaning of Eqn. (2.14) is simple. If indeed the condition holds for *all* W's, we are forced to conclude that the portion of the integrand within the curly brackets is zero everywhere. This is just another way of stating Eqn. (2.2).

We next use the divergence theorem and apply the boundary conditions. The divergence theorem gives:

$$\int_0^2 \int_0^2 \left\{ \frac{\partial s}{\partial t} - 1 \right\} W(x, y) dx dy - \oint W d \nabla s \cdot \mathbf{n} ds + \int_0^2 \int_0^2 d \nabla W \cdot \nabla s dx dy = 0 \tag{2.14}$$

where $\mathbf{n}$ is the outward pointing normal vector to the outer boundary of the ice sheet. To enforce boundary conditions $s = 0$ at $x = 0, 2$, $y = 0, 2$, we restrict the otherwise arbitrary W's to be zero at the boundaries. This restriction makes the second term (the integral over the boundary) vanish. We are thus left with the simpler expression (which involves only

single derivatives of W and s):

$$\int_0^2 \int_0^2 \left\{ \frac{\partial s}{\partial t} - 1 \right\} W(x,y) dxdy + \int_0^2 \int_0^2 d \nabla W \cdot \nabla s dxdy = 0 \tag{2.15}$$

We can further simplify the modelling problem by taking advantage of the lines of symmetry which divide the numerical domain. In particular, we restrict our attention to the upper right-hand quadrant of the numerical domain and require that $\nabla s \cdot \mathbf{n} = 0$ on the two lines of symmetry ($x = 1$ and $y = 1$) where the quadrant joins its neighboring quadrants. In this circumstance, we reduce the size of the numerical domain to $\frac{1}{4}$ of that considered in the finite-difference approach above. Thus, our attention now focuses on the following expression:

$$\int_1^2 \int_1^2 \left\{ \frac{\partial s}{\partial t} - 1 \right\} W(x,y) dxdy + \int_1^2 \int_1^2 d \nabla W \cdot \nabla s dxdy = 0 \tag{2.16}$$

We reiterate that satisfaction of the above expression for arbitrary W, with the single restriction that $W = 0$ on the outer margins ($x, y = 2$), assures us that the mass continuity equation is satisfied.

2.3.1 Discretization by the Finite-Element Method

Up to now, we have said nothing about finite-elements. We develop a finite-element solution to Eqn. (2.16) by partitioning the numerical domain (the upper right-hand quadrant of the square ice sheet) into N triangles (as shown in Fig. 2.9) and allow both s and W to vary linearly within each triangle (and be continuous across the boundary of each triangle). (Other kinds of element shapes and interpolation functions can be considered. These are the simplest, and are adequate for most ice-sheet modelling problems.) For simplicity, we require that the effective diffusivity d be a piecewise constant function, *i.e.,* d is constant within each triangular element. (This choice is justified by the fact that the gradient of s on which d depends is constant within each element when s is linear within each element. We will learn in

the accompanying exercises that error in representing d can be a major cause of inaccuracy in finite element methods.) Denoting each triangular element by the index e, the functions s and W within element e are represented by

$$s(x,y,t) = \sum_{l=1}^{3} s_l^e(t)\phi_l^e(x,y) \tag{2.17}$$

$$W(x,y) = \sum_{k=1}^{3} w_k^e \phi_k^e(x,y) \tag{2.18}$$

where the $s_l^e(t)$'s and w_k^e's are the values of $s(x,y,t)$ and $W(x,y)$ at the triangle vertices (called nodes, the index determines which node), and the $\phi_j^e(x,y)$'s are linear interpolation functions of the form

$$\phi_j^e(x,y) = \alpha_j^e x + \beta_j^e y + \gamma_j^e \tag{2.19}$$

for $j = 1,\ldots,3$. The coefficients α_j^e, β_j^e and γ_j^e are determined by the requirement that ϕ_j^e equal 1 at the location of the node who's number is j and 0 at the two other nodes:

$$\begin{bmatrix} x_1^e & y_1^e & 1 \\ x_2^e & y_2^e & 1 \\ x_3^e & y_3^e & 1 \end{bmatrix} \begin{bmatrix} \alpha_j^e \\ \beta_j^e \\ \gamma_j^e \end{bmatrix} = \begin{bmatrix} \delta_{1,j} \\ \delta_{2,j} \\ \delta_{3,j} \end{bmatrix} \tag{2.20}$$

where $\delta_{i,j}$ is the Kroneker delta (one if $i = j$ and zero if $i \neq j$) and the points (x_l^e, y_l^e) are the $l = 1,\ldots,3$ coordinates of the vertices of element e.

Substitution of Eqns. (2.17) and (2.18) into Eqn. (2.16) gives

$$\sum_{e=1}^{N} w_k^e \left\{ \frac{\partial s_l^e}{\partial t} \int\int_e \phi_l^e \phi_k^e dxdy + s_l^e \int\int_e d^e \left[\phi_{l,x}^e \phi_{k,x}^e + \phi_{l,y}^e \phi_{k,y}^e \right] dxdy - \int\int_e \phi_k^e dxdy \right\} = 0 \tag{2.21}$$

where the subscript after the comma denotes the partial derivative of the subscripted variable by the subscripting variable, where nonrepeating indices k and l are summed (summation convention), and where

$$d^e = \left(\frac{s_1^3 + s_2^e + s_3^e}{3} \right)^5 \sum_{i=1}^{3} \sum_{j=1}^{3} \left(s_i^e s_j^e \left(\nabla \phi_i^e \cdot \nabla \phi_j^e \right) \right) \tag{2.22}$$

The above expression may be simplified by recognizing that each of the integrands may be carried out explicitly using the definition of $\phi_j^e(x,y)$ given in Eqns. (2.19) and (2.20):

$$\begin{aligned}
\int\int_e \phi_k^e dxdy &= \frac{a^e}{3} \\
\int\int_e \phi_l^e \phi_k^e dxdy &= \begin{cases} \dfrac{a^e}{6} & l = k \\ \dfrac{a^e}{12} & l \neq k \end{cases} \\
\int\int_e \phi_{l,x}^e \phi_{k,x}^e dxdy &= \alpha_l^e \alpha_k^e a^e \\
\int\int_e \phi_{l,y}^e \phi_{k,y}^e dxdy &= \beta_l^e \beta_k^e a^e
\end{aligned} \tag{2.23}$$

where a^e is the area of element e given by

$$a^e = \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ x_1^e & x_2^e & x_3^e \\ y_1^e & y_2^e & y_3^e \end{vmatrix} \tag{2.24}$$

and where the vertical bars denote the determinant. With these substitutions, Eqn. (2.21) becomes

$$\sum_{e=1}^{N} w_k^e a^e \left\{ \frac{\partial s_l^e}{\partial t} \begin{cases} \frac{1}{6} & l = k \\ \frac{1}{12} & l \neq k \end{cases} + s_l^e d^e \left(\alpha_l^e \alpha_k^e + \beta_l^e \beta_k^e \right) - \frac{1}{3} \right\} = 0 \tag{2.25}$$

At this stage, we discretize the time derivative using an implicit finite-difference time step to obtain:

$$\sum_{e=1}^{N} w_k^e a^e \left\{ [s_l^e]^{n+1} \begin{cases} \frac{1}{6\Delta t} & l = k \\ \frac{1}{12\Delta t} & l \neq k \end{cases} + [s_l^e]^{n+1} [d^e]^n \left(\alpha_l^e \alpha_k^e + \beta_l^e \beta_k^e \right) - \frac{1}{3} - [s_l^e]^n \begin{cases} \frac{1}{6\Delta t} & l = k \\ \frac{1}{12\Delta t} & l \neq k \end{cases} \right\} = 0 \tag{2.26}$$

where superscripts n and $n+1$ denote the time step.

The values of $W(x,y)$ at each of the M node points in the finite-element mesh are arbitrary. We are thus forced to insist that the expression contained

within the curly brackets of Eqn. (2.26) be zero *for each* k, *i.e.*,

$$\sum_{e=1}^{N} a^e \left\{ [s_l^e]^{n+1} \left\{ \begin{array}{ll} \frac{1}{6\Delta t} & l=k \\ \frac{1}{12\Delta t} & l\neq k \end{array} \right\} + [s_l^e]^{n+1}[d^e]^n \left(\alpha_l^e \alpha_k^e + \beta_l^e \beta_k^e\right) - \frac{1}{3} - [s_l^e]^n \left\{ \begin{array}{ll} \frac{1}{6\Delta t} & l=k \\ \frac{1}{12\Delta t} & l\neq k \end{array} \right\} \right\} = 0 \tag{2.27}$$

for $k = 1, 2, 3$.

Given that the M nodes may be numbered consecutively to obtain a unique numbering scheme that is independent of the numbering scheme used to order the vertices of each particular element, we realize that Eqn. (2.27) may be written in matrix notation in a manner similar to Eqn. (2.7) used in the finite-difference formulation:

$$\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R} \tag{2.28}$$

where $\mathbf{s}^n$ is the column vector composed of the values of s_m^n arranged by the order in which the nodes are numbered. Thus, the p'th element of $\mathbf{s}^n$ is the value of s^n at the p'th node. The matrix element A_{pq} describes how the solution at node p depends on the solution at node q.

The algorithm needed to construct $\mathbf{A}$ and $\mathbf{R}$ in Eqn. (2.28) is somewhat tedious to envision because the numbering scheme of nodes in the mesh differs from that that which determines the numbering of vertices within each element. Denoting the node number of the k'th vertex of the e'th element by $p(e,k)$, the matrix $\mathbf{A}$ and the right-hand-side vector $\mathbf{R}$ may be constructed by accumulating the appropriate entries generated by a loop through each element $e = 1, \ldots, N$ and through the two vertex indices $k = 1, 2, 3$ and $l = 1, 2, 3$

$$\left\{ \begin{array}{ll} \frac{a^e}{6\Delta t} & l=k \\ \frac{a^e}{12\Delta t} & l\neq k \end{array} \right\} + a^e[d^e]^n \left(\alpha_l^e \alpha_k^e + \beta_l^e \beta_k^e\right) \quad \rightarrow \quad A_{q(e,k),r(e,l)}$$

$$\sum_{l=1}^{3} [s_l^e]^n \left\{ \begin{array}{ll} \frac{a^e}{6\Delta t} & l=k \\ \frac{a^e}{12\Delta t} & l\neq k \end{array} \right\} + \frac{a^e}{3} \quad \rightarrow \quad R_{q(e,k)} \tag{2.29}$$

An efficient way to keep track of the node-numbering scheme is to define a $N \times 3$ incidence array **Index**. Each row of **Index** corresponds to a particular element. The three column entries within a given row e contain the global node numbers of the three vertices of the particular element e.

From this point on, the finite-element method is exactly the same as the finite-difference method discussed above. To time step s forward in time, it is necessary to construct $\mathbf{A}$ and $\mathbf{R}$, perform the Cholesky factorization (remembering to take advantage of the minimum-degree ordering convention to reduce FLOPS), and then solve for $\mathbf{s}^{n+1}$.

2.3.2 Mesh Generation

One aspect which makes the finite-element method difficult is that the numerical domain must be broken up into (presumably) a collection of irregularly shaped triangles. The fact that this can be done gives the finite-element method an advantage over finite-difference methods in dealing with complex geometries.

The practical difficulty of generating a finite-element mesh should not be underestimated, however. Fortunately, there are software packages becoming available (such as Argus Meshmaker Pro for MACINTOSH and other workstation platforms) which help with this arduous task. Here, we will create the finite-element mesh used to simulate the upper right-hand quadrant of the numerical domain in the EISMINT exercise using the MATLAB script listed below. In circumstances where the ice-sheet geometry might be more complicated, it might be advantageous to use the mesh-generation software provided by ARGUS MESHMAKER PRO . This software produces the arrangement of nodes and triangles according to user-specified node densities. It also creates the incidence array $\mathbf{T}$ which is needed to construct the matrix equation developed in the previous section. Figure (2.9) displays the finite-element mesh used to compare with the finite-difference solution of the governing equations. The silhouette of the adjacency matrix associated with the finite-element mesh is displayed in Fig. (2.10).

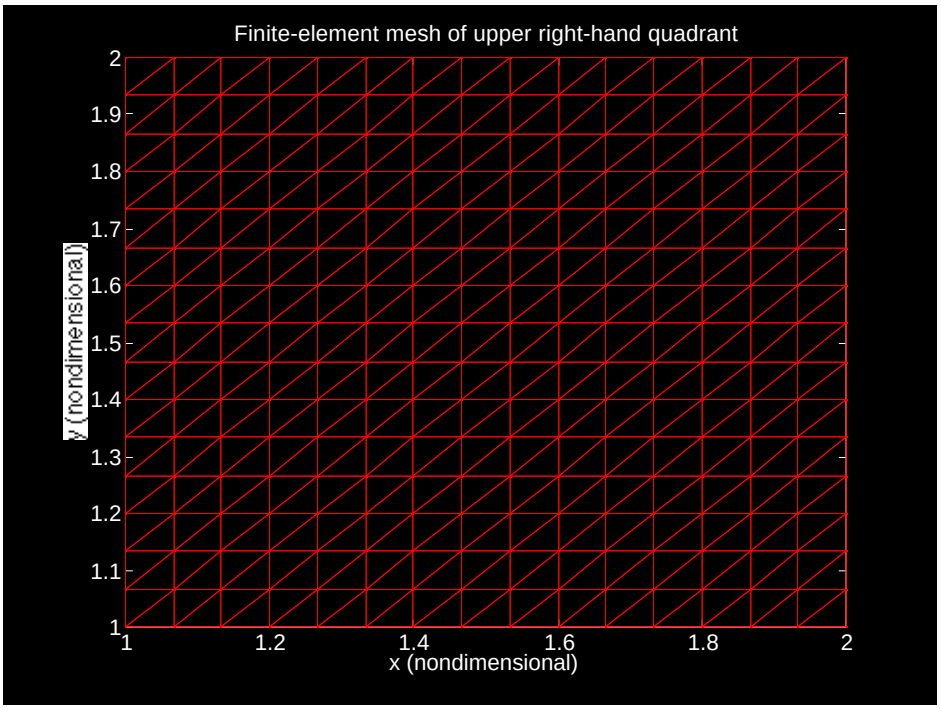

Figure 2.9: Finite-element mesh of upper right-hand quadrant. Zero ice thickness is imposed on two boundaries and zero surface elevation gradient (in direction normal to boundary) is imposed on the other two boundaries. This mesh was constructed using the MATLAB script. In more difficult problems involving complex geometry, it might be advantageous to use a mesh making software package such as ARGUS MESHMAKER PRO .

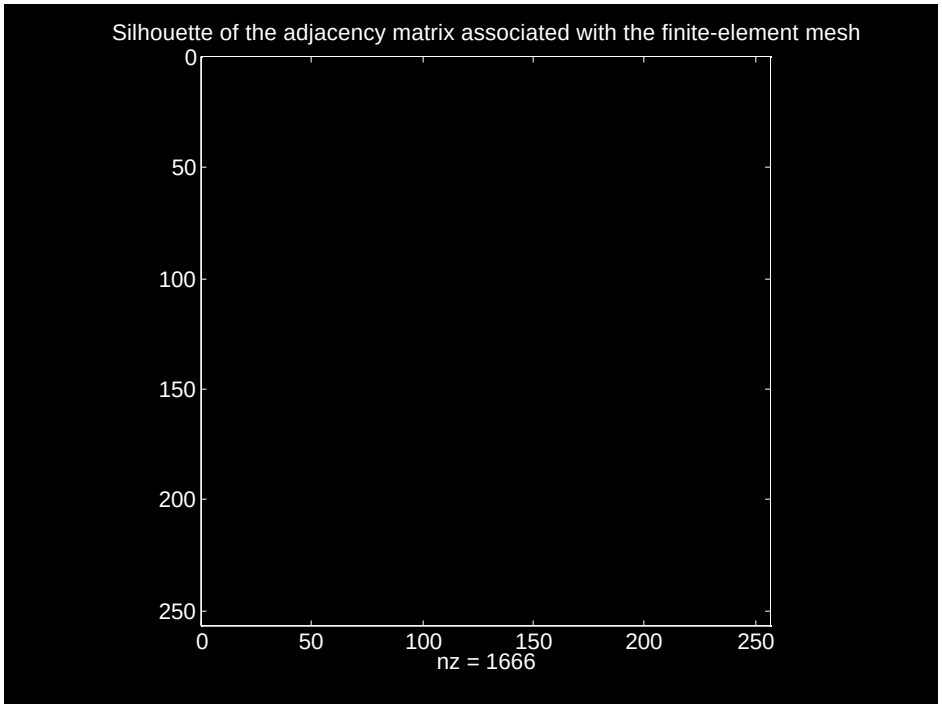

Figure 2.10: Silhouette of the adjacency matrix associated with the finite-element approach to the problem. As with the finite-difference approach, the finite-element approach generates a matrix with sparse structure.

2.3.3 Finite-element simulation: steady state

The MATLAB script which implements the finite-element ice-sheet model is listed below, and was run for a 15,000-year simulation starting from a zero-thickness initial condition. The period of integration was shortened despite the EISMINT exercise instructions because the MACINTOSH CPU is too slow to produce the 200,000-year simulation in a reasonable amount of time. (It would require several days of dedicated computing to complete a 200,000-year run for either the finite-difference or the finite-element model. New runs with the PowerPC MACINTOSH yield the full solution due to a processor speed of approximately 50 times faster than the MacIIfx.) The CPU time required to make one time step is about 35 seconds on the CPU platform used in this exercise (40 Mhz 68030 with FPU). This is slightly longer than that required for the finite-difference version of the model. The time difference is attributed to the fact that it takes more FLOPS to construct the matrix $\mathbf{A}$ with a finite-element approach than to do so with a finite-difference approach. The reason for this disparity is the fact that the do-loop required to construct $\mathbf{A}$ with finite elements is of length $N \approx 2M$, where N is the number of elements and M is the number of nodes (grid points). With a finite-difference approach, the do-loop is of length M. The finite-element time step did not take twice as long as the finite-difference time step, however, because of the fact that the number of nodes was only about $\frac{1}{4}$'th of the number of grid points, and thus $\mathbf{A}$ had smaller dimension (recall that the finite-element approach is able to exploit the symmetry of the ice sheet due to the ease with which zero-gradient boundary conditions are specified). With larger problems involving greater numbers of nodes or grid points, the finite-element method should run about the same speed as the finite-difference method because, in large problems, the Cholesky factorization step (of $\mathbf{A}$) consumes the most CPU time, and the size of the matrix $\mathbf{A}$ is the same for finite-elements and finite-differences.

The result of the 15,000-year finite-element simulation is shown in Fig. (2.11), which displays the approach to steady state. A contour map of the surface elevation in Fig. (2.12) shows the upper right-hand quadrant of the EISMINT ice sheet which, because of symmetry, constitutes the numerical domain in the finite-element model.

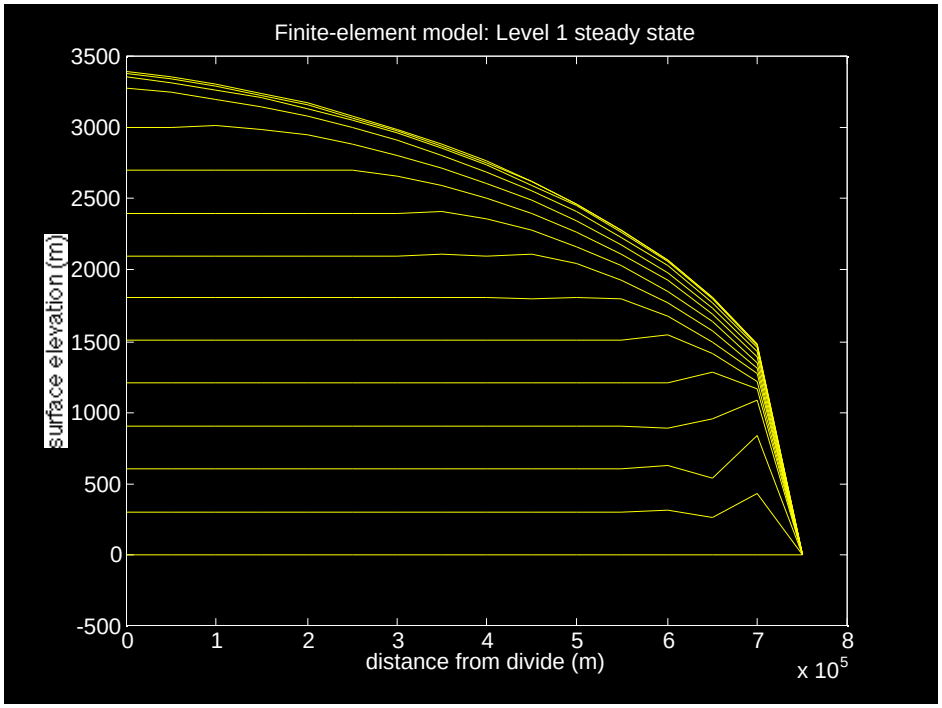

Figure 2.11: Finite-element simulation of growth to steady state (surface profile is shown at 1,000-year increments). The asterisks denote the exact, analytic solution for the azimuthally symmetric ice sheet discussed in the previous chapter. The crosses denote the result of the finite-difference simulation discussed previously in this chapter. Note the rough, grid-to-grid point noise during the initial evolution. This noise can only be overcome by adopting a smaller time step (the time step was 10 years in this particular run).

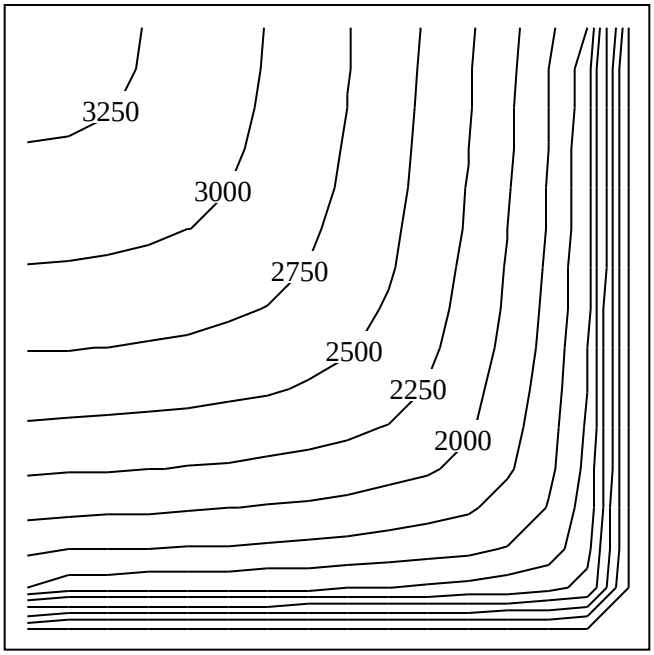

Figure 2.12: Finite-element simulation of growth to steady state (surface profile is shown at 15,000 years after model start-up). Only the upper right-hand quadrant of the EISMINT ice sheet was simulated because of symmetry (boundary conditions of $\nabla z_s \cdot \mathbf{n} = 0$ were applied at junctions where the quadrant joins the neighboring quadrants). This SPYGLASS contour plot rotates the axes of the image so that the region pictured appears as if it were the lower right-hand quadrant (the rows in TRANSFORM are listed from top down instead of from bottom up).

[Note: the following MATLAB script makes use of the fact that a^e is the same for all elements, and can thus be canceled out completely from all terms of the equation to be solved. In circumstances where element size varies, the element areas must be factored back into the script. I wrote this script when I had mistakenly assumed that the areas would cancel out in all circumstances, thus I was unfortunate in not being able to discover my mistake in this particular example, until after the exercise was complete.]

```
% This script represents a finite-element model of an ice sheet
%
% Warning:  element areas have cancelled for this
% particular exercise only.
N=16;
nodes=N^ 2;
nel=(N-1)^ 2*2;
row=zeros(4050,1);
col=zeros(4050,1);
%value=zeros(4050,1);
g=9.81;
rho=910;
phi=zeros(3,3);
Ao=1/31556926 * 1e-16;
a=0.3/31556926;
L=1500e3/2;
Z=( 5*a*L^ 4/( 2 * Ao * (rho*g)^ 3 ) )^ (1/8);
r=linspace(0,1,N)';
s_exact=( 4 * ( (1/2).^ (4/3) - (r/2).^ (4/3) ) ).^ (3/8);
hold off,clg,plot(L*r,Z*s_exact,'g*'); hold on
nsteps=1500;
dt=10*31556926*a/Z;
% Initialize at zero ice thickness
sn=zeros(N,N);
s=zeros(nodes,1);
% Create the interpolation functions.
alpha=zeros(nel,3);
beta=zeros(nel,3);
for n=1:nel
```

```
%
[lowtri uptri]=lu([[xy(index(n,1),1) xy(index(n,2),1) xy(index(n,3),1)]'...
[xy(index(n,1),2) xy(index(n,2),2) xy(index(n,3),2)]' ones(3,1)]);
phi(:,1)=uptri\(lowtri\ [1 0 0]');
phi(:,2)=uptri\(lowtri\ [0 1 0]');
phi(:,3)=uptri\(lowtri\ [0 0 1]');
for k=1:3
alpha(n,k)=phi(1,k);
beta(n,k)=phi(2,k);
end
end
%
for time=1:nsteps
time
%
value=zeros(4050,1);
R=zeros(nodes,1);
d=zeros(nel,1);
count=0;
for n=1:nel
%
% Effective diffusivity
for l=1:3
for k=1:3
d(n)=d(n)+...
s(index(n,l))*s(index(n,k))*(alpha(n,l)*alpha(n,k)+beta(n,l)*beta(n,k));
end
end
d(n)=d(n)*((s(index(n,1))+s(index(n,2))+s(index(n,3)))/3)^ 5;
%
% Load matrix and right-hand-side vector:
%
R(index(n,1))=R(index(n,1))+1/3;
R(index(n,2))=R(index(n,2))+1/3;
R(index(n,3))=R(index(n,3))+1/3;
for l=1:3
for k=1:3
```

```
count=count+1;
row(count)=index(n,k);
col(count)=index(n,l);
if l == k
R(index(n,k))=R(index(n,k))+s(index(n,l))/(6*dt);
value(count)=1/(6*dt);
else
R(index(n,k))=R(index(n,k))+s(index(n,l))/(12*dt);
value(count)=1/(12*dt);
end
value(count)=value(count)+d(n)*(alpha(n,l)*alpha(n,k)+beta(n,l)*beta(n,k));
end
end
% End loop over elements
end
A=sparse(row,col,value);
% Boundary condition at terminus
for i=1:31
A(Bound(i),Bound(i))=1.e12; % is a trick to use large number
R(Bound(i))=0;
end
% Solve the system for new thickness
s=A\R;
if rem(time,20) == 1
for i=1:16
for j=1:16
sn(i,j)=s(gamma(i,j));
end
end
plot(L*r,Z*sn(:,1));
end
% End time-stepping loop
end
```

Before the above MATLAB script can be run, we must generate the initial condition, which is here chosen to be an ice-thickness field that is 9/10'ths that of the steady-state field generated by the finite-difference model. Pre-

suming that the 31×31 array of nondimensional surface elevations `sn` was saved at the end of the finite-difference run, the following MATLAB statements will set up the initial condition for the above code:

```
load sn
snew=sn(16:31,16:31)
sn=snew
```

A comparison of the final, steady-state surface-elevation profiles for the finite-element and finite-difference versions of the ice-sheet model are shown below:

$$z_s^{fem}(t = 15,000) = 10^3 \text{ m } \times \begin{bmatrix} 3.3889 \\ 3.3550 \\ 3.3035 \\ 3.2404 \\ 3.1668 \\ 3.0828 \\ 2.9878 \\ 2.8809 \\ 2.7604 \\ 2.6238 \\ 2.4676 \\ 2.2864 \\ 2.0713 \\ 1.8081 \\ 1.4814 \\ -0.0000 \end{bmatrix} \tag{2.30}$$

$$z_s^{fd}(t = 50,000) = 10^3 \text{ m } \times \begin{bmatrix} 3.4218 \\ 3.3911 \\ 3.3398 \\ 3.2772 \\ 3.2044 \\ 3.1212 \\ 3.0273 \\ 2.9217 \\ 2.8027 \\ 2.6679 \\ 2.5138 \\ 2.3346 \\ 2.1204 \\ 1.8521 \\ 1.4807 \\ 0.0000 \end{bmatrix} \tag{2.31}$$

The finite-element profile is a bit smaller than the finite-difference profile, and this may result partially from the fact that the finite-element run was only 15,000 years instead of 50,000 years. The shorter finite-element run was necessitated by CPU constraints. This intercomparison defect may be fixed at a later date.

A word about time step size

For both the finite-element and finite-difference models constructed here, the time-stepping scheme is implicit (the gradient of s is evaluated at the $n+1$'th time step instead of the n'th time step). The nonlinearity of the flow law is treated by lagging the effective diffusivity by one time step (d is evaluated at the n'th time step for the computation of $\mathbf{s}^{n+1}$). Under many circumstances, implicit time steps are absolutely stable and can be used with arbitrarily large time-step sizes, Δt, provided accuracy is not a concern. Because of nonlinearity in the effective diffusivity, the implicit time stepping scheme used here is not absolutely stable. A Δt size of 10 years (dimensional form) was the largest time step that could be accomplished without introducing grid-point-to-grid-point wiggles in the solution. There is no formula to determine

the largest practical Δt; thus, the researcher will have to perform some trial-and-error experimentation to determine an appropriate time-step size.

2.4 Comparison Between Finite-Difference and Finite-Element Approaches

By using both a finite-difference and a finite-element approach to the EISMINT exercise, we have an opportunity to compare and contrast the two approaches to ice-sheet modelling. The two approaches are similar in that they both generate a sparse matrix equation which must be solved to step through a time step:

$$\mathbf{A}\mathbf{s}^{n+1} = \mathbf{R} \tag{2.32}$$

The matrix $\mathbf{A}$ is slightly less sparse in the finite-element approach because of the fact that finite-elements typically connect a node point with 6 of its neighbors, whereas finite-differences typically connect a grid point with only 4 of its neighbors. The time taken to construct the finite-difference version of $\mathbf{A}$ is generally less than that taken to construct the finite-element counterpart. This is due both to the fact that the finite-difference $\mathbf{A}$ has fewer nonzero elements and to the fact that $\mathbf{A}$ can be constructed on only one pass through the grid points. The construction of the finite-element version of $\mathbf{A}$ requires one pass through the elements, and often there are about twice as many elements as there are grid points or nodes.

The comparison between finite-difference and finite-element approaches is not dependent on whether an iterative relaxation scheme is used to solve the sparse matrix equation listed above. For the present EISMINT exercise, we have chosen to use direct, sparse-matrix solution schemes to obtain $\mathbf{s}^{n+1}$ from the above matrix equation. We have learned that both the finite-difference and finite-element approaches generate symmetric, positive definite matrices $\mathbf{A}$ which are amenable to Cholesky factorization (the physics of the system and not the numerical method is what determines these properties of the matrix). Use of the Cholesky factorization and a minimum-degree node numbering scheme keeps the number of floating point operations (FLOPS) to a minimum in each time step. This computational consideration is taken care

of automatically by MATLAB , the mathematical engine behind the models we have created. In FORTRAN applications on another computer, the factorization of **A** may require careful user intervention to assure that it is done as efficiently as possible.

The finite-element approach is advantageous in circumstances where flux or gradient boundary conditions are specified, and where irregular model geometry might be desired. A personal preference of the author is that the finite-element approach is more elegant to describe and easier to code in MATLAB or FORTRAN.

Above all, it is worth remembering that there is ultimately little difference between the finite-element and finite-difference approaches that should concern the scientific significance of the numerical modelling runs. Most differences are details best appreciated in terms of software engineering.

2.4.1 Exercise 2

Create a finite-element model of the axisymmetric ice sheet of radius 750 km. Compare the solution to the Nye-Vialov solution developed in Chapter 1. Use the MATLAB mat-file (data) `circlemesh.mat` to generate the finite-element mesh.

2.4.2 Exercise 3

Create a finite-element model of Greenland using the mesh data provided in `Greenlandmesh.mat`. Experiment with accumulation rate parameterizations to see if you can reproduce the observed elevation profile of the ice sheet.

Chapter 3

Ice-Shelf Dynamics

Before delving into the numerical modelling associated with the EISMINT ice-shelf exercise, it is useful to review the derivation of some of the governing equations for ice-shelf evolution (as in previous chapters, we postpone considering ice-shelf thermodynamics). These governing equations and their derivation can be intimidating to the newcomer to glaciology. The goal of the present chapter is to increase the "comfort level" associated with ice-shelf dynamics before venturing into the numerical analysis needed to perform the EISMINT exercises.

3.1 What Makes an Ice Shelf Different From an Ice Sheet?

As with a similar question about the sexes, there are a multitude of answers one might give to such a question as that posed in the section heading. The answer that will be given here stems from the technical obstacle that ice-shelf modellers face when trying to adapt an ice-sheet model to an ice-shelf application. For both ice sheets (grounded ice) and ice shelves (floating ice), the mass continuity equation yields the relation which governs the time

evolution of the ice mass:

$$\frac{\partial h}{\partial t} = a - \nabla \cdot \mathbf{q} \tag{3.1}$$

where, as before, h is the ice thickness, t is time, a is the net accumulation rate (which includes both surface and basal mass exchange processes), $\nabla\cdot$ is the two-dimensional divergence operator, and $\mathbf{q}$ is the two-dimensional mass transport vector. The difference between ice-sheet modelling and ice-shelf modelling lies in the definition of $\mathbf{q}$:

ice-sheet model:

$$\mathbf{q} = -2(\rho g)^3 (\nabla z_s \cdot \nabla z_s) \nabla z_s \int_{z_b}^{z_s} \int_{z_b}^{z} A(T(z'))(z_s - z')^3 dz' dz \tag{3.2}$$

ice-shelf model:

$$\mathbf{q} = \int_{\Omega} \int h(x', y') \mathbf{Q}(x, y; x', y') dx' dy' \tag{3.3}$$

where x and y are horizontal coordinates (geographic coordinates could suffice instead), A is a temperature-dependent flow law parameter, ρ is the ice density (assumed constant with depth in this exercise), g is the gravitational acceleration, T is temperature, z is the vertical coordinate, z_s and z_b are the elevations of the ice-sheet surface and base, respectively, and Ω is the plan-view domain of the ice shelf.

Nonlocality

The formula for the mass transport vector $\mathbf{q}$ associated with the ice shelf (Eqn. 3.3) is a formal representation that has little practical value. It does make a point forcefully, however: the flux at position (x, y) depends on the ice thickness at all other points (x', y') within the domain Ω. The vector-valued Green's function $\mathbf{Q}(x, y; x', y')$ is termed the *influence function* that describes how the existence of (presumably straining) ice thickness at one location affects the flow of the ice shelf at another. Buried within $\mathbf{Q}$ is the information about the ice front and ice-stream input boundary conditions.

This additional information could, if desired, be shown explicitly in the formal definition of $\mathbf{q}$ for an ice shelf. We won't bother to do so because we realize that the expression for $\mathbf{q}$ in an ice shelf given by Eqn. (3.3) is too formal to be useful to us in constructing a finite-difference or finite-element model of an ice shelf.

What is important, and what is to be emphasized, is the fact that the ice-shelf mass flux is a *non-local* function of the ice thickness. This is a stark contrast to the case for the grounded ice sheet. As seen in Eqn. (3.2), the ice-sheet mass flux is purely a function of the local ice thickness and surface gradient. Information about the ice thickness and surface gradient at points surrounding the location where the ice flux is desired need not be known to compute $\mathbf{q}$.

3.2 How to Deal with a Non-Local Definition of Mass Flux

In practice, ice-shelf modellers define the ice transport in terms of a horizontal, depth-averaged ice velocity field $\mathbf{u} = (u, v)$:

$$\mathbf{q} = \mathbf{u}h \tag{3.4}$$

where the horizontal velocity field is a solution of the *reduced* [Morland, 1987] or otherwise *simplified* stress-equilibrium equations. These equations are:

$$\frac{\partial}{\partial x}\left(2\bar{\nu}h\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(\bar{\nu}h\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right) = \rho g h\frac{\partial z_s}{\partial x} \tag{3.5}$$

$$\frac{\partial}{\partial y}\left(2\bar{\nu}h\left(2\frac{\partial v}{\partial y}+\frac{\partial u}{\partial x}\right)\right)+\frac{\partial}{\partial x}\left(\bar{\nu}h\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right) = \rho g h\frac{\partial z_s}{\partial y} \tag{3.6}$$

where the depth-averaged effective viscosity $\bar{\nu}$ (overbar denotes depth averaging) is usually defined by

$$\bar{\nu} = \frac{1}{h}\int_{z_b}^{z_s}\frac{B(T(z))\;dz}{2\left[\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2+\frac{1}{4}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2+\frac{\partial u}{\partial x}\frac{\partial v}{\partial y}\right]^{\frac{n-1}{2n}}} \tag{3.7}$$

We shall refer to Eqns. (3.5) and (3.6) as the ice-shelf stress-balance equations, and note the fact that they are second-order, elliptic partial differential equations for u and v. These equations will require boundary conditions to be specified along the margins of the ice shelf defined by the contour $\delta\Omega$. We shall describe these boundary conditions after first deriving Eqns. (3.5) and (3.6) from first principles below.

The above definition for the effective viscosity can be recognized as that viscosity which would yield a power-law constitutive relation between stress and strain rate. The ice-stiffness parameter B is related to the flow-law parameter A referred to in the previous chapters as follows:

$$B = A^{\frac{-1}{n}} \approx A^{\frac{1}{3}} \tag{3.8}$$

The exponent n here does not refer to a time-step index, but rather to the power-law flow exponent which is usually taken to be 3. For the EISMINT exercises, we are asked to assume a constant (temperature independent) flow-law parameter $A_o = 10^{-16}$ Pa$^{\frac{-1}{3}}$ year^{-1} $= 3.1689 \times 10^{-24}$ Pa$^{\frac{-1}{3}}$ s^{-1}. This corresponds to an ice stiffness parameter of $B_o = 1.4688 \times 10^8$ Pa s$^{\frac{1}{3}}$.

Prognostic and diagnostic equations

We see that the way in which ice-shelf modellers get around the problem of a non-local ice flux is to use *two* instead of *one* governing equations. The pair of equations (and their boundary conditions which we don't describe at this point) for u and v given above (Eqns. 3.5 and 3.6) are called *diagnostic* equations because they do not involve time. The diagnostic equations allow us to determine u, v, and thus $\mathbf{q}$ for a given ice-thickness field $h(x, y)$. The solution to the diagnostic equations, once found, is stuffed into Eqn. (3.1), which is referred to as the *prognostic* equation. The prognostic equation is then used to update h through a time step, and then the process is repeated.

In summary, a practical approach to ice-shelf modelling is to employ a two-step time-stepping procedure: First, solve the diagnostic equation for the velocity field. Second, stuff the velocity field into the mass balance equation and solve for a new ice thickness.

3.3 Derivation of the Diagnostic (Velocity) Equations

One might wonder where Eqns. (3.5) and (3.6) come from. Many researchers have independently derived these equations using various approaches and levels of elegance, sophistication and panache. To a person just beginning to learn about ice-shelf modelling, these derivations can sometimes appear intimidating and full of mathematical rhetoric that can obscure the simplicity of the physical system. In an effort to avoid such intimidation here, we shall derive the diagnostic equations using as little formalism as possible. The price we shall pay is that we will have to employ several assumptions at critical stages that cannot be justified in a crisp, elegant manner. Given our interest in getting on with the ice-shelf modelling exercise, we shall accept this price and proceed saying nothing more about the formalism which can be appreciated by looking elsewhere in the literature [*e.g.,* consult the bibliography, and particularly: Weertman, 1957; Van der Veen, 1986; Muszynski and Birchfield, 1987; Morland , 1987; Morland and Shoemaker, 1981; Morland and Zainuddin, 1987; Sanderson and Doake, 1979; Shumskiy and Krass, 1976; Herterich, 1987; Hutter, 1983; MacAyeal and Barcilon, 1988; MacAyeal, 1989].

Viscous Flow Law

For simplicity (since it really doesn't affect the derivation), we shall assume a viscous flow law,

$$\mathbf{T}' = 2\nu\dot{\mathbf{e}} \tag{3.9}$$

Here $\mathbf{T}' = \mathbf{T} + P\mathbf{I}$ is the deviatoric stress, $\mathbf{T}$ is the stress, $P = -\frac{1}{3}T_{ii}$ is the pressure, and $\dot{\mathbf{e}}$ is the strain rate defined by $\dot{e}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$.

Stokes' equations

The stress-equilibrium equations (referred to as the Stokes equations in the case of a viscous fluid) state that the divergence of the stress tensor is equal to the body forces. These equations are written:

$$-\frac{\partial P}{\partial x}+\frac{\partial}{\partial x}\left(2\nu\dot{e}_{xx}\right)+\frac{\partial}{\partial y}\left(2\nu\dot{e}_{xy}\right)+\frac{\partial}{\partial z}\left(2\nu\dot{e}_{xz}\right) = 0 \tag{3.10}$$

$$\ldots = \ldots$$

$$-\frac{\partial P}{\partial z}+\frac{\partial}{\partial x}\left(2\nu\dot{e}_{zx}\right)+\frac{\partial}{\partial y}\left(2\nu\dot{e}_{zy}\right)+\frac{\partial}{\partial z}\left(2\nu\dot{e}_{zz}\right) = \rho g \tag{3.11}$$

For convenience, we have omitted the equation for the y-component of the momentum. All manipulations of this omitted equation proceed as for the x-component equation.

Surface boundary condition

The boundary condition at the upper surface of the ice shelf $z = z_s$ is stress-free, *i.e.*,

$$\mathbf{T}\cdot\mathbf{n}_s = \mathbf{0} \tag{3.12}$$

where $\mathbf{n}_s$ is the outward-pointing unit normal vector given by

$$\mathbf{n}_s = \frac{\mathbf{n}_z - \frac{\partial z_s}{\partial x}\mathbf{n}_x - \frac{\partial z_s}{\partial y}\mathbf{n}_y}{\sqrt{1+\left(\frac{\partial z_s}{\partial x}\right)^2+\left(\frac{\partial z_s}{\partial y}\right)^2}} \tag{3.13}$$

and where $\mathbf{n}_x$, $\mathbf{n}_y$ and $\mathbf{n}_z$ are unit vectors that point along the subscripted coordinates.

Application of the tensor/vector product in Eqn. (3.12) gives three equations which must be satisfied at $z = z_s$:

$$(2\nu\dot{e}_{xx}-P)\frac{\partial z_s}{\partial x}+2\nu\dot{e}_{xy}\frac{\partial z_s}{\partial y}-2\nu\dot{e}_{xz} = 0 \tag{3.14}$$

$$\ldots \;=\; \ldots$$

$$2\nu\dot{e}_{xz}\frac{\partial z_s}{\partial x} + 2\nu\dot{e}_{yz}\frac{\partial z_s}{\partial y} - (2\nu\dot{e}_{zz} - P) \;=\; 0 \tag{3.15}$$

where, again, we avoid writing the second of the three equations for convenience.

Basal boundary condition

The boundary condition at the bottom surface of the ice shelf $z = z_s$ is not stress-free, *i.e.,*

$$\mathbf{T} \cdot \mathbf{n}_s = -\rho g h \mathbf{n}_b \tag{3.16}$$

where $\mathbf{n}_b$ is the outward-pointing unit normal vector to the bottom surface of the ice shelf. Here, we have made the assumption (and it's a pretty good one) that the pressure of the sea water at the bottom of the ice shelf is equal to the hydrostatic pressure necessary to float the ice shelf, *i.e.,* the ice shelf is floating in a medium with a hydrostatic pressure field. Manipulation of the tensor/vector product yields the following three equations which apply at $z = z_b$:

$$(2\nu\dot{e}_{xx} - P)\frac{\partial z_b}{\partial x} + 2\nu\dot{e}_{xy}\frac{\partial z_b}{\partial y} - 2\nu\dot{e}_{zx} \;=\; -\rho g h \frac{\partial z_b}{\partial x} \tag{3.17}$$

$$\ldots \;=\; \ldots$$

$$2\nu\dot{e}_{xz}\frac{\partial z_b}{\partial x} + 2\nu\dot{e}_{yz}\frac{\partial z_b}{\partial y} - (2\nu\dot{e}_{zz} - P) \;=\; \rho g h \tag{3.18}$$

Vertical integration

We anticipate (from observations of ice-shelf geometry and flow) that the horizontal velocities and strain rates are independent of z. A sensible thing

to try in an effort to simplify the Stokes' equations (Eqns. 3.10 and 3.11) is to integrate them over depth. We consider the result of integrating the equation for the x-momentum balance first:

$$-\int_{z_b}^{z_s} \frac{\partial P}{\partial x} dz + \int_{z_b}^{z_s} \frac{\partial}{\partial x} \left(2\nu \dot{e}_{xx}\right) dz + \int_{z_b}^{z_s} \frac{\partial}{\partial y} \left(2\nu \dot{e}_{xy}\right) dz + \int_{z_b}^{z_s} \frac{\partial}{\partial z} \left(2\nu \dot{e}_{xz}\right) dz = 0 \quad (3.19)$$

We can move the partial derivatives that appear inside the above integrals to outside the integrals by using Leibnitz' rule:

$$\int_{z_b}^{z_s} \frac{\partial f(x,z,...)}{\partial x} dz = \frac{\partial}{\partial x} \int_{z_b}^{z_s} f(x,z,...) dz - f(x,z_s,...) \frac{\partial z_s}{\partial x} + f(x,z_b,...) \frac{\partial z_b}{\partial x} \quad (3.20)$$

The result of using Leibnitz' rule on Eqn. (3.19) is:

$$\begin{aligned} -\frac{\partial}{\partial x} \int_{z_b}^{z_s} P dz + \frac{\partial}{\partial x} \int_{z_b}^{z_s} 2\nu \dot{e}_{xx} dz + \frac{\partial}{\partial y} \int_{z_b}^{z_s} 2\nu \dot{e}_{xy} dz + \int_{z_b}^{z_s} \frac{\partial}{\partial z} \left(2\nu \dot{e}_{xz}\right) dz & \\ -(2\nu \dot{e}_{xx} - P) \frac{\partial z_s}{\partial x} - 2\nu \dot{e}_{xy} \frac{\partial z_s}{\partial y} + 2\nu \dot{e}_{xz} & \\ +(2\nu \dot{e}_{xx} - P) \frac{\partial z_b}{\partial x} + 2\nu \dot{e}_{xy} \frac{\partial z_b}{\partial y} - 2\nu \dot{e}_{xz} &= 0 \end{aligned} \quad (3.21)$$

We recognize that the last two lines of the above equation restate the left-hand side of the x component of the surface and basal boundary conditions (Eqns. 3.14 and 3.17). Making use of these boundary conditions, the above equation simplifies to

$$-\frac{\partial}{\partial x} \int_{z_b}^{z_s} P dz + \frac{\partial}{\partial x} \int_{z_b}^{z_s} 2\nu \dot{e}_{xx} dz + \frac{\partial}{\partial y} \int_{z_b}^{z_s} 2\nu \dot{e}_{xy} dz + \int_{z_b}^{z_s} \frac{\partial}{\partial z} \left(2\nu \dot{e}_{xz}\right) dz = 0 \quad (3.22)$$

Zero shear approximation

We now apply the key assumption which gives ice-shelf dynamics its distinctive "flavor". We assume that the horizontal velocities and strain rates are

independent of z. This assumption is justified by the flat and thin geometry of ice shelves, and by the fact that seawater is nearly inviscid when compared to ice. Thus, we assume that u, v, $\dot{e}_{xx}$, $\dot{e}_{yy}$, and $\dot{e}_{xy}$ are independent of z, and that $\dot{e}_{zx} = \dot{e}_{zy} = 0$.

The first consequence of the above assumptions is obtained by integrating the vertical component of the Stokes' equation (Eqn. 3.11), and making use of the boundary condition (Eqn. 3.15) at $z = z_s$. From this integration, we find that the pressure field is found to have a glaciostatic component and a dynamics component as follows:

$$P = \rho g(z_s - z) + 2\nu\dot{e}_{zz} \tag{3.23}$$

This allows us to evaluate the integral of pressure over depth giving:

$$-\frac{\partial}{\partial x}\int_{z_b}^{z_s} P dz = -\frac{\partial}{\partial x}\left[\frac{\rho g h^2}{2} - 2\nu h(\dot{e}_{xx} + \dot{e}_{yy})\right] \tag{3.24}$$

where we have made use of the incompressibility condition, $\dot{e}_{zz} = -(\dot{e}_{xx}+\dot{e}_{yy})$. With these assumptions and simplifications, Eqn. (3.22) becomes:

$$\begin{aligned}\frac{\partial}{\partial x}\left[2\nu h(2\dot{e}_{xx} + \dot{e}_{yy})\right] + \frac{\partial}{\partial y}\left[2\nu h\dot{e}_{xy}\right] &= \rho g h\left(\frac{\partial z_b}{\partial x} + \frac{\partial h}{\partial x}\right) \\ &= \rho g h\frac{\partial z_s}{\partial x}\end{aligned} \tag{3.25}$$

where we have made use of the fact that $h = z_s - z_b$. Making use of the definitions for the strain-rate components in terms of the horizontal velocities, we obtain the equation we desired:

$$\frac{\partial}{\partial x}\left(2\nu h\left(2\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right) + \frac{\partial}{\partial y}\left(\nu h\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right) = \rho g h\frac{\partial z_s}{\partial x} \tag{3.26}$$

The other diagnostic equation follows by a similar derivation to the one above. The flow law for ice may be introduced into the above diagnostic equation without difficulty. (Vertical integration will necessitate that we use the vertical average of the effective viscosity $\bar{\nu}$ owing to the temperature dependence of the ice-stiffness parameter B.)

3.4 Boundary Conditions

The ice-shelf stress-equilibrium equations (Eqns. 3.5 and 3.6) require boundary conditions to be specified along the contour $\delta\Omega$ which defines the boundary to the ice-shelf domain (denoted by Ω). There are two types of boundary conditions: kinematic and dynamic. Kinematic conditions (specification of velocity) are usually applied where ice-shelves abut stagnant, zero slip coastlines (such as where the Ross Ice Shelf abuts the Transantarctic mountains), or where ice streams flow into the ice shelf (in which case the velocity is specified from an examination of ice-stream dynamics). Dynamic conditions (specification of stress) are usually applied at the seaward, iceberg-calving front.

Kinematic conditions

Depth-averaged ice velocity is specified at all junctions with grounded ice or coastlines where the ice shelf shears past stagnant rock. Typically, ice flow into the ice shelf at grounding lines of ice sheets can have z-dependence which is incompatible with the z-independent horizontal flow of the ice shelf. As described by Barcilon and MacAyeal [1988], the z-dependent structure of the input velocity is winnowed out of the net horizontal flow within a narrow transition zone between ice-sheet and ice-shelf flow regimes. This winnowing process is of little interest in most glaciological problems involving ice shelves. Thus, it is sufficient to ignore the winnowing process and simply specify the depth-averaged input velocity as the correct boundary condition.

Dynamic conditions

The balance of forces at the seaward ice front also introduces z-dependent structure in the ice-shelf flow which is winnowed away within a narrow transition zone extending inward from the ice front. As suggested by Morland [1987], this winnowing is of little interest, and may be safely ignored in the specification of boundary conditions for the ice-shelf stress-equilibrium equa-

tions. The balance of forces at the ice front which is relevant as a boundary condition is the depth-integrated balance:

$$\int_{z_b}^{z_s} \mathbf{T} \cdot \mathbf{n} dz = -\frac{\rho_w g}{2} \left(\frac{\rho}{\rho_w} h \right)^2 \mathbf{n} \tag{3.27}$$

where $\mathbf{n}$ is the outward-pointing normal to the portion of $\delta\Omega$ that represents the ice front ($\mathbf{n}$ is restricted to lie in the horizontal plain), and where ρ_w is the average density of seawater. Here we have made use of the assumption that the ice shelf floats in hydrostatic equilibrium with seawater, *i.e.,* that $z_b = -\frac{\rho}{\rho_w} h$.

The integral on the left-hand side of Eqn. (3.27) represents the depth-integrated force transmitted across the ice front due to internal stresses (pressure and deviatoric stress) in the ice shelf. The right-hand side of the above equation represents the integral of the hydrostatic pressure in the seawater beyond the ice front over the face of the ice front. This force balance is summarized in Fig. (3.1).

To render Eqn. (3.27) into a form suitable to be specified as a boundary condition on the ice-shelf stress-balance equations, the flow law must be used to replace the stress $\mathbf{T}$ with the strain rate $\dot{\mathbf{e}}$. The strain rate components may then be written in terms of the derivatives of u and v.

3.4.1 Dynamic condition if $\mathbf{n} = \mathbf{n}_x$

As an illustration of how a dynamic boundary condition would be specified for an ice front that has an outward-pointing normal aligned with the x-axis (the ice front extends along the y-axis, and the ice shelf is to the left and the ocean to the right), *i.e.,* $\mathbf{n} = \mathbf{n}_x$. With this geometry,

$$\mathbf{T} \cdot \mathbf{n}_x = \left(2\nu \dot{e}_{xx} - P \right) \mathbf{n}_x \tag{3.28}$$

and Eqn. (3.27) becomes,

$$\int_{z_b}^{z_s} \mathbf{T} \cdot \mathbf{n} dz = -\frac{\rho_w g}{2} \left(\frac{\rho}{\rho_w} h \right)^2 \mathbf{n}_x \tag{3.29}$$

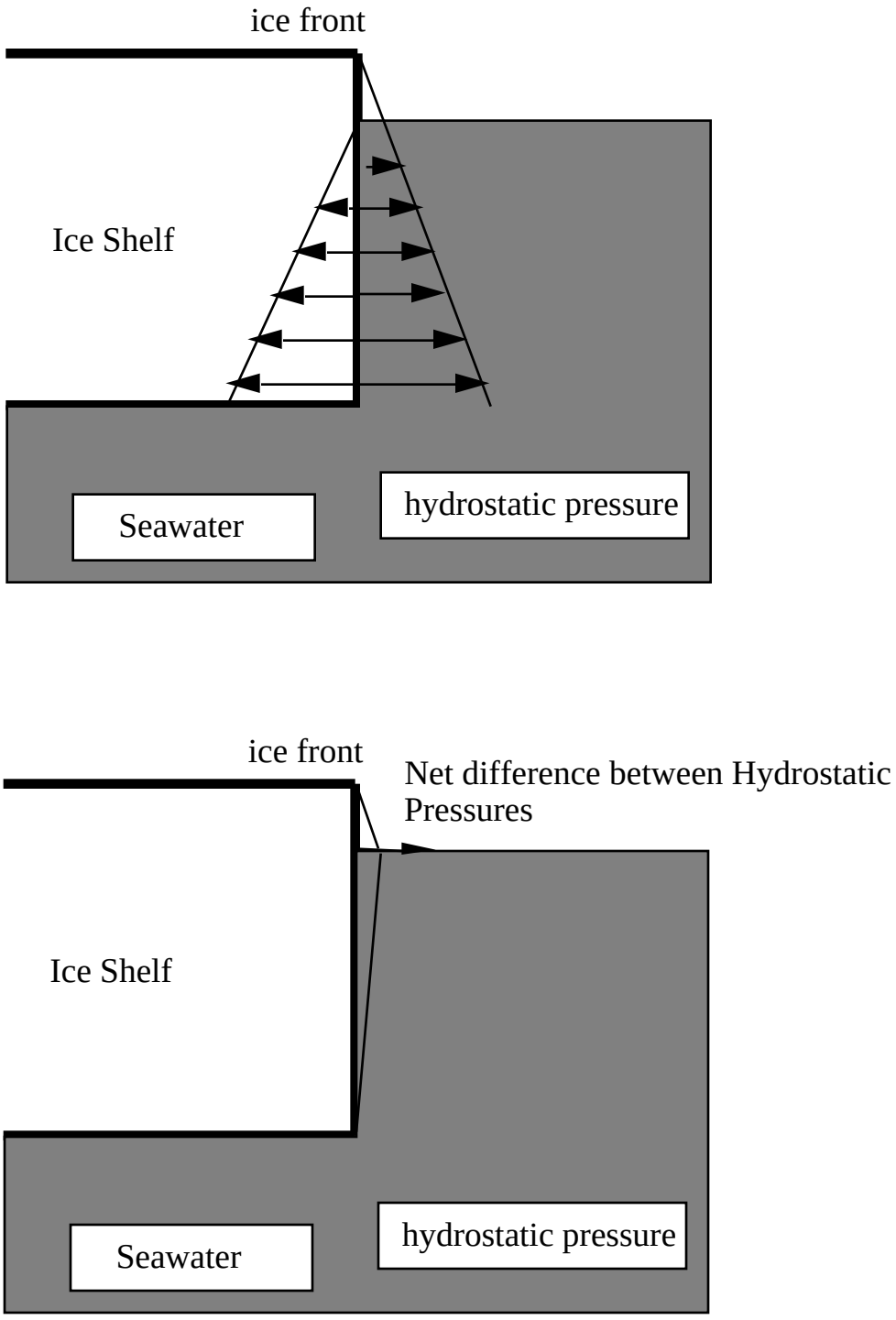

Figure 3.1: Force balance at an ice front which determines the dynamic boundary condition on an ice shelf (after T. Hughes, personal communication).

To write the above dynamic condition in a form which involves u and v, we substitute for $\mathbf{T}$ using the flow law and the definition of the deviatoric stress and then integrate over z. The integral of pressure over depth is evaluated as follows

$$\begin{aligned} \int_{z_b}^{z_s} P dz &= \int_{z_b}^{z_s} \left(\rho g(z_s - z) + 2\nu\dot{e}_{zz}\right) dz \\ &= \rho g \frac{h^2}{2} - 2\bar{\nu}h(\dot{e}_{xx} + \dot{e}_{yy}) \end{aligned} \tag{3.30}$$

where we have made use of the definition of pressure given by Eqn. (3.23) and the z-independence of the strain rates, and the definition $\int_{z_b}^{z_s} \nu dz = h\bar{\nu}$. Combining Eqns. (3.28) and (3.30) with Eqn. (3.27) we obtain:

$$2\bar{\nu}h\left(2\dot{e}_{xx} + \dot{e}_{yy}\right) = \frac{\rho g h^2}{2}\left(1 - \frac{\rho}{\rho_w}\right) \tag{3.31}$$

Making use of the definitions of the strain-rate components, we obtain the boundary condition:

$$2\bar{\nu}h\left(2\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = \frac{\rho g h^2}{2}\left(1 - \frac{\rho}{\rho_w}\right) \tag{3.32}$$

3.5 Weertman's Analytic Solution

A simple and elegant solution for the unidirectional spreading rate of an ice shelf (ice tongue confined in a rectangular embayment with frictionless sides as shown in Fig. 3.2) was derived by Weertman [1957] using the above boundary condition. Inspection of the ice-shelf stress-balance equations indicates that for an ice shelf of uniform thickness (h = constant) allowed to flow in the x-direction only (as if confined by an embayment with parallel sides), the horizontal spreading rate $\dot{e}_{xx}$ will be independent of x. Thus, the value of $\dot{e}_{xx}$ at the ice-front boundary will have the same value as everywhere else.

We can determine $\dot{e}_{xx}$ by considering Eqns. (3.7) and (3.31). The only non-zero term in the denominator of the integrand in the definition of $\bar{\nu}$ (Eqn. 3.7) is the $\left(\frac{\partial u}{\partial x}\right)^2 = \dot{e}_{xx}^2$ term which, due to the z-independence of the flow, can be moved outside of the integral over z. We may thus write $\bar{\nu}$ as follows for this special case:

$$\bar{\nu} = \frac{\bar{B}}{2\left|\dot{e}_{xx}\right|^{\frac{n-1}{n}}} \tag{3.33}$$

where $\bar{B} = \frac{1}{h}\int_{z_b}^{z_s} B dz$. The absolute value of $\dot{e}_{xx}$ appears in the above equation to emphasize the fact that $(\dot{e}_{xx}^2)^{\frac{n-1}{2n}}$ is a positive quantity. Failure to remember this (*e.g.,* what would happen if you just canceled the 2's in the exponent) can cause problems in modelling arbitrary ice-shelf regimes where $\dot{e}_{xx}$ can be both positive and negative. If we assume that $\dot{e}_{xx} > 0$, then the above expression simplifies to:

$$\bar{\nu} = \frac{\bar{B}}{2\dot{e}_{xx}^{\frac{n-1}{n}}} \tag{3.34}$$

Substitution of the above expression into Eqn. (3.31) gives:

$$\dot{e}_{xx}^{\frac{1}{n}} = \frac{\rho g h}{4\bar{B}}\left(1 - \frac{\rho}{\rho_w}\right) \tag{3.35}$$

The spreading rate

$$\dot{e}_{xx} = \left[\frac{\rho g h}{4\bar{B}}\left(1 - \frac{\rho}{\rho_w}\right)\right]^n \tag{3.36}$$

is thus seen to be proportional to $h^n \approx h^3$.

Marine ice-sheet instability

It is important to note the fact that Weertman's solution, $\dot{e}_{xx} \propto h^3$, forms the essence of the *marine ice-sheet instability* he described in 1974 [Weertman, 1974]. The idea is simple. As the grounding line of an ice sheet retreats into deeper water (assuming the bed of the ice sheet slopes downward toward the ice-sheet interior), the spreading rate and the rate at which the

grounding line wants to retreat (if not held in check by snow accumulation and advection) both increase. Thus, once triggered, a grounding line cannot avoid catastrophic, uncontrolled retreat. Of course, this does not happen in practice (thank God!), because real ice shelves (such as those surrounding the West Antarctic ice sheet) are confined in embayments and thus rarely spread at the unconfined limit determined above. This serves to illustrate why ice-shelf dynamics continues to be of interest to glaciologists and other scientists interested in the question of marine ice-sheet stability [*e.g.,* Van der Veen, 1986; Van der Veen, 1985]

3.6 Van der Veen's Exact, Analytic Solution for a Floating Ice Tongue

For use in eventual model testing, we shall derive the exact, analytic solution for the thickness and velocity profiles of a floating ice tongue allowed to spread unidirectionally along the x-axis as originally developed by Van der Veen [1986a, 1986b]. The geometry is essentially the same as that shown in Fig. (3.2) for the Weertman solution. We assume that the ice tongue extends from the origin at $x = 0$ along the positive x-axis. We allow $h(x)$ to be a function of x, and consider the effects of snow accumulation a (assumed constant) and possible input ice flux Q_o at the inland boundary at $x = 0$.

The ice-shelf stress-equilibrium equations (Eqn. 3.5 and 3.6) are simplified by the assumptions that $v = 0$ and that all y-derivatives vanish. This reduces the stress-balance equations to a single equation for $u(x)$:

$$\frac{\partial}{\partial x}\left(4\bar{\nu}\frac{\partial u}{\partial x}\right) = \rho g h \frac{\partial z_s}{\partial x} \tag{3.37}$$

The effective viscosity may be eliminated using Eqn. (3.33) giving:

$$\frac{\partial}{\partial x}\left(2B_o h \left|\frac{\partial u}{\partial x}\right|^{\frac{1-n}{n}} \frac{\partial u}{\partial x}\right) = \rho g h \frac{\partial z_s}{\partial x} \tag{3.38}$$

Noting the fact that $\frac{\partial u}{\partial x} > 0$ in a steady-state ice tongue, the above equation

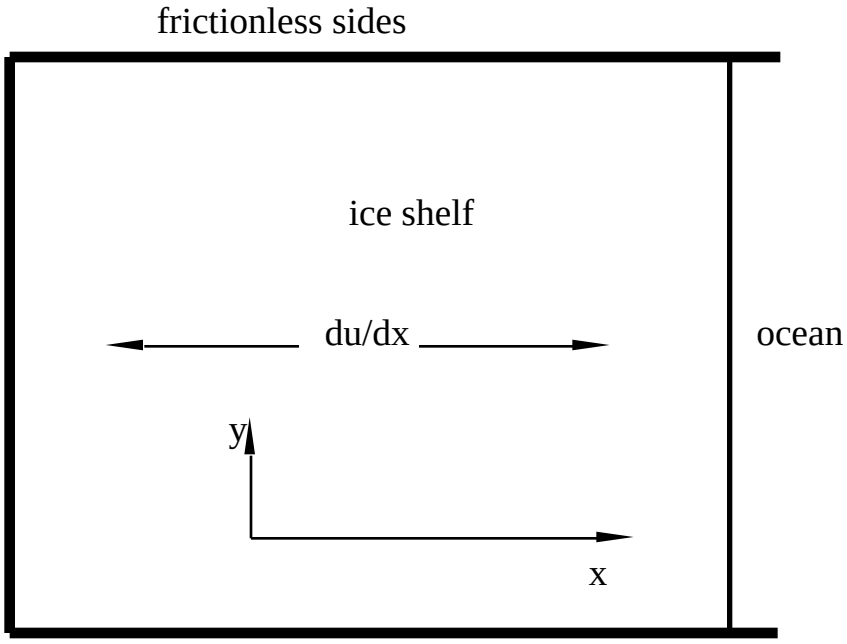

Figure 3.2: Geometry of Weertman's [1957] solution. This geometry is also used to produce Van der Veen's [1986a] exact, analytic solution for a floating ice tongue. For Van der Veen's problem, we assume that the grounding line is at $x = 0$ where an input flux is specified.

simplifies to become

$$\frac{\partial}{\partial x}\left(2B_o h\left(\frac{\partial u}{\partial x}\right)^{\frac{1}{n}}\right) = \rho g h \frac{\partial z_s}{\partial x} \tag{3.39}$$

where we have assumed a constant ice-stiffness parameter $\bar{B} = B_o = 1.4688 \times 10^8$ Pa $\mathrm{s}^{\frac{-1}{3}}$ following the conventions of the EISMINT Level 1 exercises. As a reminder, we cannot be cavalier about the absolute value signs appearing in Eqn. (3.38) when we are dealing with an ice tongue in which $\frac{\partial u}{\partial x}$ can have arbitrary sign.

Nondimensionalization

Before developing the exact, analytic solution to the above equation, we adopt nondimensional variables using the following substitutions:

$$\begin{aligned} u &\rightarrow Uu \\ x &\rightarrow Lx \\ h &\rightarrow Zh \\ z_s &\rightarrow \left(1 - \frac{\rho}{\rho_w}\right) Zh \end{aligned}$$

To simplify Eqn. (3.39), we insist that the scale factors Z, U and L satisfy the following relationship

$$2B_o \left(\frac{U}{L}\right)^{\frac{1}{n}} \frac{Z}{L} = \frac{\rho g Z^2}{2L}\left(1 - \frac{\rho}{\rho_w}\right) \tag{3.40}$$

With this relationship, the nondimensional form of Eqn. (3.39) becomes particularly simple:

$$(h(u')^{\frac{1}{n}})' = (h^2)' \tag{3.41}$$

Here we use primes to denote differentiation by x as a reminder that the governing equation has been reduced from a partial differential equation to an ordinary differential equation of the single variable x.

Velocity profile

Integrating the above equation (Eqn. 3.41) once gives,

$$h(u')^{\frac{1}{n}} = h^2 + c \tag{3.42}$$

The constant of integration c is zero because we require u' to vanish as $h \to 0$. Thus, a bit of rearrangement of exponents gives

$$u' = h^n \tag{3.43}$$

Integration a second time gives the velocity profile of the ice shelf expressed in terms of the ice thickness profile:

$$u(x) = u(0) + \int_0^x h(s)^n ds \tag{3.44}$$

where $u(0)$ is the ice velocity at the inland ice junction $x = 0$ (also referred to as the grounding line). Under most modelling circumstances, $u(0)$ is specified as a kinematic boundary condition.

Mass balance

The mass balance equation (Eqn. 3.1) in nondimensional form, and making use of the relation $q = uh$, is

$$1 = (uh)' \tag{3.45}$$

where we have insisted on a second simplifying relationship between the scales U, L, and Z:

$$\frac{aL}{UZ} = 1 \tag{3.46}$$

Recall that the snow accumulation rate a in the EISMINT Level 1 exercises is assumed constant.

The product rule for differentiation, when applied to the right-hand side of Eqn. (3.45), and use of Eqn. (3.43), gives

$$1 = uh' + h^{n+1} \tag{3.47}$$

Van der Veen's approach

Van der Veen [1986a,b] found an ingenious way to eliminate u from the above equation to obtain an ordinary differential equation for h only. He recognized that mass balance, in integral form, requires that the mass flux $q = hu$ increase linearly with x. This linear relationship follows from the mass continuity equation. This relationship is makes sense because, to accommodate the increasing burden of snow accumulated upstream as one moves their reference point downstream along the x-axis, ice transport must increase with x. In particular,

$$hu = q_o + x \tag{3.48}$$

where q_o is the mass flux at the ice-shelf grounding line (junction with inland ice). Multiplication of Eqn. (3.47) by h, and use of the above relation, gives:

$$h = (q_o + x)h' + h^{n+2} \tag{3.49}$$

or,

$$\int \frac{dx}{q_o + x} = \int \frac{dh}{h - h^{n+2}} \tag{3.50}$$

Integration

The left-hand side of the above equation may be integrated readily to yield

$$\int \frac{dx}{q_o + x} = \ln(q_o + x) \tag{3.51}$$

The right-hand side of Eqn. (3.50) is a bit more laborious to integrate, but is readily done as follows

$$\begin{aligned} \int \frac{dh}{h - h^{n+2}} &= \int \frac{dh}{h} + \int \frac{h^n dh}{1 - h^{n+1}} \\ &= \ln h - \frac{1}{n+1} \ln(1 - h^{n+1}) + c \\ &= \ln \left(\frac{h}{(1 - h^{n+1})^{\frac{1}{n+1}}} \right) + c \end{aligned} \tag{3.52}$$

where c is a constant of integration necessitated by the fact that the integrals in Eqn. (3.50) are indefinite.

Making the above substitutions into Eqn. (3.50) for the integrals, and taking the exponential of both sides to invert the logarithm function, gives

$$(q_o + x) = c \left\{ \frac{h}{(1 - h^{n+1})^{\frac{1}{n+1}}} \right\} \tag{3.53}$$

A bit of rearrangement gives

$$h = \left[1 + c \left(q_o + x \right)^{-n-1} \right]^{\frac{-1}{n+1}} \tag{3.54}$$

In the above expression we have taken the liberty to redefine c as necessary to avoid complication.

The constant of integration c is evaluated by the requirement that, at $x = 0$, $h = h_o = \frac{H_o}{Z}$, where H_o is the dimensional value of h at $x = 0$,

$$c = \left(h_o^{-n-1} - 1 \right) q_o^{n+1} \tag{3.55}$$

Again, by judiciously insisting that the scales of nondimensionalization U, L and Z satisfy a third relation, namely

$$1 = \frac{Q_o^{n+1}}{(UZ)^{n+1}} \left(\frac{H_o^{-n-1}}{Z^{-n-1}} - 1 \right) \tag{3.56}$$

The end result is the exact, analytic solution for the ice-thickness profile in nondimensional form (again, due to Van der Veen):

$$h(x) = \left[1 + \frac{q_o^{n+1} \left(h_o^{-(n+1)} - 1 \right)}{\left(q_o + x \right)^{n+1}} \right]^{\frac{-1}{n+1}} \tag{3.57}$$

It should be noted that if $q_o = 0$, the constant of integration c must be determined another way. We will not consider this alternative nondimensionalization here because the EISMINT exercise involves a nonzero q_o.

Ice-shelf thickness as $x \to \infty$

The ice-shelf thickness profile given by Eqn. (3.57) indicates that $\lim_{x\to\infty} h(x) = 1$. This is a consequence of the fact that the thinning due to horizontal expansion just balances the accumulation rate (presumed constant in this exercise) when $h(x) = 1$. All the "action" in ice-tongue dynamics (unconfined ice shelf dynamics) occurs very close to the grounding line where the ice shelf adjusts from an arbitrary input thickness to the asymptotic far-field thickness of 1.

3.6.1 Evaluation of Scales

In deriving the exact, analytic ice-thickness profile for the floating ice tongue above, we have insisted that the scales of nondimensionalization U, Z and L satisfy two relationships, which are repeated below (using the $n = 3$ flow law):

$$2B_o \left(\frac{U}{L}\right)^{\frac{1}{3}} = \frac{\rho g Z}{2}\left(1 - \frac{\rho}{\rho_w}\right) \tag{3.58}$$

$$\frac{a}{Z}\left(\frac{U}{L}\right)^{-1} = 1 \tag{3.59}$$

Before we can evaluate the exact, analytic solution, or conduct numerical modelling experiments, we must determine the values of the dimensional scales U, L, and Z using the above three equations. Observe that Eqn. (3.58) gives the Weertman [1957] unidirectional spreading rate for an ice thickness Z:

$$\frac{U}{L} = \left(\frac{\rho g Z}{4B_o}\left(1 - \frac{\rho}{\rho_w}\right)\right)^3 \tag{3.60}$$

By using Eqn. (3.58) for U/L in Eqn. (3.59), we can derive an expression for Z:

$$Z = \frac{a^{\frac{1}{4}}\left(4B_o\right)^{\frac{3}{4}}}{\left(\rho g\left(1 - \frac{\rho}{\rho_w}\right)\right)^{\frac{3}{4}}} \tag{3.61}$$

Substitution of the above result for Z into Eqn. (3.60) or Eqn. (3.59), and arbitrarily choosing U to equal the dimensional form of the grounding-line velocity $u(0)$, gives an expression for L, *e.g.,*

$$L = \frac{ZU}{a} \tag{3.62}$$

Evaluation of the above expressions using EISMINT Level 1 constants ($\rho = 910$ kg m^{-3}, $g = 9.81$ m s^{-2}, $\rho_w = 1028$ kg m^{-3}, $a = 0.3$ m year^{-1}, $Q_o = 4 \times 10^5$ m^2 year^{-1}, $H_o = 10^3$ m, and $B_o = 1.4688 \times 10^8$ Pa s$^{\frac{1}{3}}$) gives:

$$\begin{aligned} Z &= 205.7426 \text{ m} \\ U &= 400 \text{ m yr}^{-1} \\ L &= 274.32 \text{ km} \end{aligned}$$

3.6.2 Exact solution

The following MATLAB script was used to evaluate the exact, analytic ice-thickness profile of the ice tongue expressed in Eqn. (3.57).

```
% This program computes the exact, analytic thickness and velocity
% profiles for an ice tongue alowed to spread unidirectionally.
Qo=4e5/31556926;
Ho=1.0e3;
rho=910;
rho_w=1028;
a=0.3/31556926;
Bo=1.4688e8;
g=9.81;
Z=a^ (1/4)*(4*Bo)^ (3/4)/(rho*g*(1-rho/rho_w))^ (3/4);
UonL=(rho*g*Z*(1 - rho/rho_w)/(4*Bo))^ 3;
U=Qo/Ho;
L=Z*U/a;
qo=Qo/(U*Z);
ho=Ho/Z;
```

```
%
x=linspace(0,1,100);
h=(1+ qo^ 4*(ho^ (-4)-1)./(qo+x).^ 4 ).^ (-1/4);
plot(L*x,Z*h)
```

A plot of $h(x)$ over the nondimensional interval $x \in [0,1]$ is presented in dimensional form in Fig. (3.3). What is striking about the thickness profile is the extreme degree of thinning within the first 25 km. Most ice shelves in nature fail to show such extreme thinning near the grounding line, and this can be attributed to the fact that confinement within embayments tends to reduce the horizontal spreading rate well below its unconfined limit.

3.7 Ice-Stream Dynamics

Ice-stream dynamics is the same as ice-shelf dynamics in most respects [MacAyeal, 1989]. The two additional aspects which must be accounted for in determining the flow and mass balance of an ice stream (at the level described here) are basal friction and the lack of flotation. The horizontal velocity field of an ice stream satisfies two equations that are virtually identical to the ice-shelf stress-equilibrium equations. They are listed as follows,

$$\frac{\partial}{\partial x}\left(2\bar{\nu}h\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(\bar{\nu}h\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right) = \rho g h\frac{\partial z_s}{\partial x}-\tau_x \tag{3.63}$$

$$\frac{\partial}{\partial y}\left(2\bar{\nu}h\left(2\frac{\partial v}{\partial y}+\frac{\partial u}{\partial x}\right)\right)+\frac{\partial}{\partial x}\left(\bar{\nu}h\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right) = \rho g h\frac{\partial z_s}{\partial y}-\tau_y \tag{3.64}$$

where τ_x and τ_y are the horizontal components of the basal traction encountered at the ice-stream bed. The lack of flotation (ice-streams sit on the bed) means that the surface elevation is no longer related to the ice thickness by the relation $z_s = (1-\frac{\rho}{\rho_w})h$ (in dimensional form).

The kinship between ice-stream dynamics and ice-shelf dynamics has

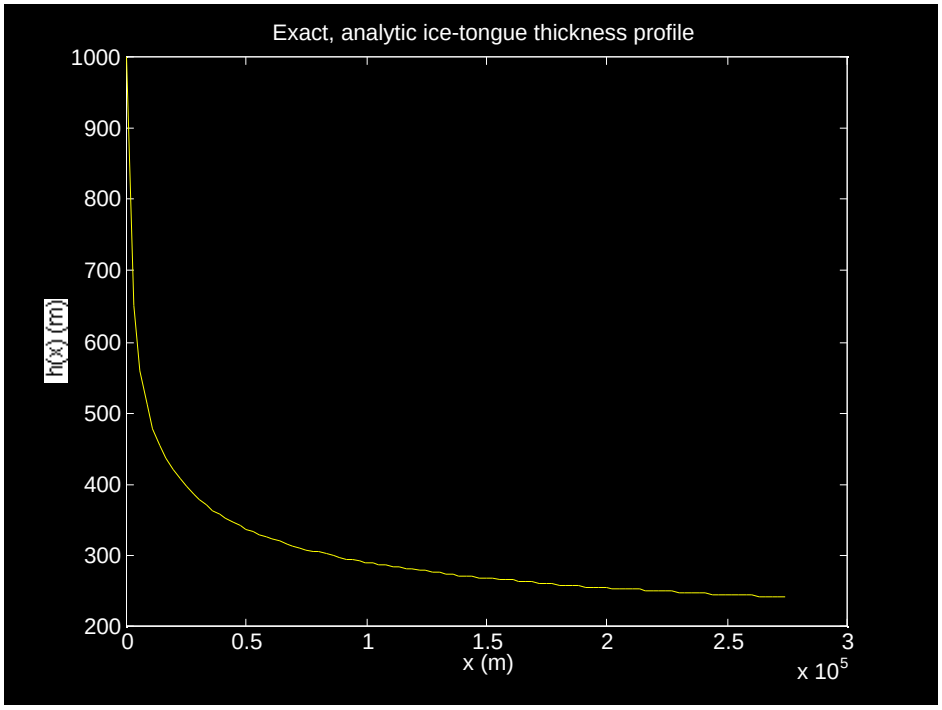

Figure 3.3: Ice thickness profile (exact) of an ice tongue allowed to spread in the longitudinal direction only.

been exploited by numerous workers [*e.g.,* Muzynski and Birchfield, 1987; MacAyeal, 1989; Jenson, 1993] who have approached the problem of specifying τ_x and τ_y with varying degrees of sophistication. At a gross, fundamental level, specification of the basal friction using the assumption that the $\underline{\tau} \propto -\mathbf{u}$ has produced reasonable agreement between model and observation. Efforts to improve this agreement focus primarily on the better specification of τ_x and τ_y rather than improvements in the underlying assumption that Eqns. (3.63) and (3.64) adequately describe ice-stream dynamics.

A recent Phd thesis by John Jenson [1993] shows how the above ice-stream dynamics (modified to account for longitudinal spreading only) can be adapted to simulate the behavior of a marginal lobe of the Laurentide ice sheet that extended along Lake Michigan into Illinois. Sampling of the basal till deposits has suggested a nonlinear basal friction law that is comparable in many ways to what is being discovered from studies of the beds of ice streams in Antarctica.

3.8 Summary

In this chapter we have taken on the onerous task of deriving ice shelf flow dynamics. The results, while tedious compared with the comparable results for grounded ice sheets, are still relatively simple. The two key points to appreciate are: (1) that ice-shelf mass flux is not a local function of the ice thickness and surface elevation, and (2) that ice-stream dynamics is essentially the same as that of ice shelves. The first point, nonlocality, necessitates taking a two-stepped approach to ice-shelf modelling, which we shall do in the next chapter. The first step is to determine via an elliptic partial differential equation the horizontal flow consistent with the current ice thickness field. The second step is to use this flow in the mass continuity equation (which is essentially hyperbolic) to update the ice thickness field through one time step. This two-stepped procedure makes ice-shelf modelling somewhat more awkward and less appealing than ice-sheet modelling.

Chapter 4

Flowline Ice-Shelf Models

In this chapter, we will construct a finite-difference model of a freely expanding ice tongue. Its purpose is to test a numerical method by comparing its solution with the exact, analytic solution developed in the previous chapter.

The ice-shelf models constructed here and in the next chapter (which takes up the issue of two-dimensional ice shelves) will share several key points. First is that the time-stepping procedure is split into a two-step algorithm as suggested in the previous chapter. One step involves solution of the diagnostic equation which determines the velocity field from the ice-thickness field (and boundary conditions). The second step involves solution of the prognostic equation which uses mass continuity to update the ice-thickness distribution. A third point in common between all ice-shelf models is the fact that the nonlinear flow law requires an iterative solution of the diagnostic (velocity) equation.

4.1 Finite-Difference Model of an Ice Tongue

We consider the ice tongue who's exact, analytic solution was obtained in § (3.6). Our goal will be to perform the EISMINT exercise in which the steady-state thickness profile of this ice tongue is achieved after an evolution from an initial condition of zero (actually 1 meter) ice thickness. (Zero thickness is inconvenient because the formula for the effective viscosity of ice is singular when $h = 0$.) The governing equations for the ice tongue, expressed in nondimensional form, are

$$\frac{\partial h}{\partial t} = 1 - \frac{\partial}{\partial x}(uh) \tag{4.1}$$

$$\frac{\partial}{\partial x}\left(h\left|\frac{\partial u}{\partial x}\right|^{\frac{1}{n}-1}\frac{\partial u}{\partial x}\right) = \frac{\partial}{\partial x}\left(h^2\right) \tag{4.2}$$

In the above diagnostic equation (Eqn. 4.2 is henceforth referred to as the diagnostic equation because it does not involve time), we have retained the absolute value in the definition of the effective viscosity because we expect to deal with situations in which $\frac{\partial u}{\partial x}$ has arbitrary sign. A convenient simplification of the above diagnostic equation is made by defining the effective viscosity c:

$$c = h\left|\frac{\partial u}{\partial x}\right|^{\frac{1}{n}-1} \tag{4.3}$$

which renders Eqn. (4.2) into the following form:

$$\frac{\partial}{\partial x}\left(c\frac{\partial u}{\partial x}\right) = \frac{\partial}{\partial x}\left(h^2\right) \tag{4.4}$$

Boundary conditions associated with the ice tongue are applied at the upstream end where the ice tongue joins the inland ice:

$$h = h_o \tag{4.5}$$

$$u = u_o \tag{4.6}$$

at $x = 0$, and at the downstream end where the ice tongue abuts the ocean:

$$\frac{\partial u}{\partial x} = h^3 \tag{4.7}$$

at $x = 1$. (We assume arbitrarily that our interest in modelling the ice tongue is restricted to the nondimensional interval $x \in [0, 1]$.) The initial condition applied at $t = 0$ is

$$h(x) = \frac{1}{Z} \tag{4.8}$$

which is 1 meter, in dimensional units.

4.2 Finite Differencing

Both the prognostic (Eqn. 4.1) and diagnostic (Eqn. 4.4) equations are discretized by the finite-difference method using a staggered-grid convention shown in Fig. (4.1). The purpose of adopting this scheme is to avoid mass conservation problems and numerical noise that can be generated if the velocity is defined at the same grid locations as the ice thickness.

4.2.1 Prognostic equation

The finite-difference form of the prognostic equation (mass continuity) is

$$\frac{h_i^{j+1}}{\Delta t} - \frac{h_i^j}{\Delta t} = 1 - \frac{1}{2\Delta x}\left(u_i^j\left(h_{i+1}^{j+1} + h_i^{j+1}\right) - u_{i-1}^j\left(h_i^{j+1} + h_{i-1}^{j+1}\right)\right) \tag{4.9}$$

for $i = 2, \ldots, N-1$ where N is the number of grid points. For $i = 1$ we use the boundary condition $h_1^{j+1} = h_o$. For $i = N$ we use the above equation, but replace h_{N+1}^{j+1} with h_N^{j+1}. This short-cut at the seaward margin of the ice tongue is not a boundary condition *per se*, but rather a numerical short-cut (that is reasonably accurate) which eliminates the need to specify the ice thickness at grid point $N + 1$ using the method of characteristics. In the above equation subscripts denote the grid-point index and superscripts

denote the time-step level, Δx is the grid spacing, and Δt is the time-step size. Slight rearrangement of terms in the above equation gives:

$$h_{i-1}^{j+1}\left(\frac{-u_{i-1}^{j}}{2\Delta x}\right)+h_{i}^{j+1}\left(\frac{1}{\Delta t}+\frac{u_{i}^{j}-u_{i-1}^{j}}{2\Delta x}\right)+h_{i+1}^{j+1}\left(\frac{u_{i}^{j}}{2\Delta x}\right)=1+\frac{h_{i}^{j}}{\Delta t} \quad (4.10)$$

Eqn. (4.10) is easily recognized as a tridiagonal system. Thus we express the entire finite-difference discretization of the prognostic equation as a single matrix equation:

$$\mathbf{A}\mathbf{h}^{j+1}=\mathbf{R} \quad (4.11)$$

where

$$\mathbf{h}^{j+1}=\begin{bmatrix} h_1^{j+1} \\ h_2^{j+1} \\ \vdots \\ h_N^{j+1} \end{bmatrix} \quad (4.12)$$

the tridiagonal matrix $\mathbf{A}$ is composed of the following nonzero elements:

$$\begin{aligned} \frac{1}{\Delta t}+\frac{u_{i}^{j}-u_{i-1}^{j}}{2\Delta x} &\rightarrow A_{i,i} \\ \frac{-u_{i-1}^{j}}{2\Delta x} &\rightarrow A_{i,i-1} \\ \frac{u_{i}^{j}}{2\Delta x} &\rightarrow A_{i,i+1} \end{aligned}$$

for $i=2,\ldots,N-1$,

$$1 \rightarrow A_{1,1}$$

and,

$$\begin{aligned} \frac{1}{\Delta t}+\frac{2u_{N}^{j}-u_{N-1}^{j}}{2\Delta x} &\rightarrow A_{N,N} \\ \frac{-u_{N-1}^{j}}{2\Delta x} &\rightarrow A_{N,N-1} \end{aligned}$$

and the right-hand-side vector $\mathbf{R}$ is given by:

$$1 + \frac{h_i^j}{\Delta t} \rightarrow R_i \tag{4.13}$$

for $i = 2, \ldots, N$, and

$$h_o \rightarrow R_1 \tag{4.14}$$

4.2.2 Diagnostic equation

The finite-difference discretization of the diagnostic equation is accomplished using the same staggered-grid convention as that used for the prognostic equation. In the notation shown below, we drop the superscript j which indicates the time-step level because all variables are evaluated at the same time-step level. The effective viscosity c is defined as follows:

$$c_i^{[k]} = h_i \left| \frac{u_i^{[k-1]} - u_{i-1}^{[k-1]}}{\Delta x} \right|^{\frac{1}{n}-1} \tag{4.15}$$

for $i = 2, \ldots, N$. The value of $c_i^{[k]}$ at grid point $i = 1$ is not referenced in the finite-difference model, hence it is not defined. The superscript $[k]$ is used to denote iteration number in the successive approximation algorithm for enforcing the nonlinear flow law. We shall postpone for now the discussion which motivates the need for the superscript $[k]$.

The definition of the effective viscosity shown above can cause difficulty when either $h = 0$ or $\left| \frac{u_i^{[k-1]} - u_{i-1}^{[k-1]}}{\Delta x} \right| = 0$. When the latter of these conditions are met, the definition of $c_i^{[k]}$ will trigger a divide-by-zero. One way to overcome this potential difficulty is to add a "tid bit" to the velocity gradient to keep the effective viscosity positive definite:

$$c_i^{[k]} = h_i \left(\left| \frac{u_i^{[k-1]} - u_{i-1}^{[k-1]}}{\Delta x} \right| + \epsilon \right)^{\frac{1}{n}-1} \tag{4.16}$$

where ϵ is a small number on the order of 10^{-25} (nondimensional) or so.

The discretization of the diagnostic equation is made simple by the above definitions, and is listed as follows:

$$\frac{1}{\Delta x^2}\left(c_{i+1}^{[k]}\left(u_{i+1}^{[k]}-u_i^{[k]}\right)-c_i^{[k]}\left(u_i^{[k]}-u_{i-1}^{[k]}\right)\right)=\frac{h_{i+1}^2-h_i^2}{\Delta x} \tag{4.17}$$

for $i = 2, \ldots, N-1$. Two boundary conditions are required. At the upstream end of the ice tongue, the velocity is specified:

$$u_1^{[k]} = u_o + \frac{\Delta x}{h_1 + h_2} \tag{4.18}$$

The second term in the above equation represents the snow accumulation between the grounding line and the location of the grid point where u_1 is specified. At the downstream end of the ice tongue, a gradient condition is specified:

$$\frac{u_N^{[k]} - u_{N-1}^{[k]}}{\Delta x} = h_N^3 \tag{4.19}$$

As with the prognostic equation, the finite-difference form of the diagnostic equation represents a tridiagonal matrix equation:

$$\mathbf{D}^{[k]}\mathbf{u}^{[k]} = \mathbf{F} \tag{4.20}$$

where,

$$\mathbf{u}^{[k]} = \begin{bmatrix} u_1^{[k]} \\ u_2^{[k]} \\ \vdots \\ u_N^{[k]} \end{bmatrix} \tag{4.21}$$

To define the matrix $\mathbf{D}^{[k]}$, we must make use of the effective viscosity vector:

$$\mathbf{c}^{[k]} = \begin{bmatrix} c_1^{[k]} \\ c_2^{[k]} \\ \vdots \\ c_N^{[k]} \end{bmatrix} \tag{4.22}$$

The matrix **D** is tridiagonal. It's nonzero elements, and the elements of **F** are summarized as follows:

$$\begin{aligned}
\frac{-1}{\Delta x^2}\left(c_i^{[k]} + c_{i+1}^{[k]}\right) &\rightarrow D_{i,i}^{[k]} \\
\frac{c_i^{[k]}}{\Delta x^2} &\rightarrow D_{i,i-1}^{[k]} \\
\frac{c_{i+1}^{[k]}}{\Delta x^2} &\rightarrow D_{i,i+1}^{[k]} \\
\frac{1}{\Delta x}\left(h_{i+1}^2 - h_i^2\right) &\rightarrow F_i
\end{aligned}$$

for $i = 2, \ldots, N-1$, and

$$\begin{aligned}
1 &\rightarrow D_{1,1}^{[k]} \\
u_o &\rightarrow F_1 \\
\frac{1}{\Delta x} &\rightarrow D_{N,N}^{[k]} \\
\frac{-1}{\Delta x} &\rightarrow D_{N,N-1}^{[k]} \\
h_N^3 &\rightarrow F_N
\end{aligned}$$

4.2.3 Viscosity iteration

At this stage, we must explain the iteration index represented by the superscript $[k]$. The diagnostic equation for the velocity **u** is nonlinear because **c** depends on **u**. A simple way to treat this nonlinearity (*i.e.,* to solve the nonlinear ordinary-differential equation) is to use the method of successive approximation. (There are better, more efficient methods, but I have not investigated them at length.)

The method of successive approximation works as follows. First, we start with a guess $\mathbf{u}^{[0]}$ (say, the result of the previous time step), and we use this guess to generate $\mathbf{c}^{[1]}$ using Eqn. (4.16). We then use $\mathbf{c}^{[1]}$ to construct $\mathbf{D}^{[1]}$, which is then used with $\mathbf{F}$ to solve for $\mathbf{u}^{[1]}$. The $\mathbf{u}^{[1]}$ is then used to generate $\mathbf{c}^{[2]}$, and the process is repeated, so on and so forth, until convergence occurs. Convergence is deemed satisfactory when the following condition is met:

$$\max_{i=2,\ldots,N} \left| \frac{c_i^{[k+1]} - c_i^{[k]}}{c_i^{[k]}} \right| < \delta \tag{4.23}$$

where δ is a small number of the order of 0.01.

A sequence of linear equations

To reiterate, the sequence of linear equations described above converge to the nonlinear equation that we ultimately wish to solve, *i.e.,* the solution of

$$\lim_{k\to\infty} \mathbf{D}^{[k]}\mathbf{u}^{[k]} = \mathbf{F} \tag{4.24}$$

is the solution of the nonlinear diagnostic equation.

Numerical flow chart

The two-step numerical procedure used to perform a time step is summarized by the flow chart shown in Fig. (4.2). Experience suggests that without the internal iteration loop on the index $[k]$, the routine is not sufficiently accurate.

4.3 Simulation of an Ice Tongue: Comparison Between Numerical and Exact Solutions

The above description of the finite-difference ice-tongue model was implemented with the MATLAB script listed below to simulate the evolution to

a steady-state from a zero-thickness initial condition. (The initial condition was actually 1 meter, for the technical reasons cited above.) The evolution was as expected. After model start-up, the grounding-line boundary condition introduced a ice-thickness jump which propagated downstream as time evolved. In the wake of the ice-thickness jump, or shock as it can be called, the ice tongue is in steady state. This correspondence between the region where the ice tongue is in steady state and the location of the ice-thickness jump is expected from the hyperbolic nature of the prognostic equation [MacAyeal and Barcilon, 1988]. One of the disappointments of the numerical scheme is the fact that the ice-thickness jump tends to broaden as time evolves. In nature, the ice-thickness jump would propagate without changing it's steep initial form. This broadening is a defect common to many finite-difference schemes.

The comparison between the exact, analytic thickness profile and that generated by the model is shown in Fig. (4.3). The comparison is satisfactory and suggests that the finite-difference model has performed up to expectation.

```
% This program models the thickness and velocity
% profiles for an ice tongue using a finite-difference approach.
N=100;
Qo=4e5/31556926;
Ho=1.0e3;
rho=910;
rho_w=1028;
a=0.3/31556926;
Bo=1.4688e8;
g=9.81;
Z=a^ (1/4)*(4*Bo)^ (3/4)/(rho*g*(1-rho/rho_w))^ (3/4);
UonL=(rho*g*Z*(1 - rho/rho_w)/(4*Bo))^ 3;
U=Qo/Ho;
L=Z*U/a;
qo=Qo/(U*Z);
ho=Ho/Z;
%
length=1.0;
c=zeros(N,1);
```

```
x=[linspace(0,length,N)]';
dx=length/(N-1);
dt=1*U/L*31556926;
h_exact=(1+ qo^ 4*(ho^ (-4)-1)./(qo+x).^ 4 ).^ (-1/4);
%
nsteps=500;
h=1/Z*ones(N,1); % start at 1 m initial condition
u=ones(N,1); % start with zero strain solution
hold off,clg,subplot(2,1,1); plot(x*L, u*U*31556926);
hold on;subplot(2,1,2);plot(x*L,Z*h);hold on
subplot(2,1,2);plot(x*L,Z*h_exact,'r+');
%
% Enter time step:
%
for j=1:nsteps
%
% Ice velocity:
%
c=ones(N,1);
c_old=zeros(N,1);
count=0;
while max((abs(c-c_old)./c)) > 0.01
c_old=c;
count=count+1;
% Construct and solve Du=F:
%
for i=2:N
c(i)=h(i)*( abs((u(i)-u(i-1)))/dx + 1.e-25)^ (-2/3); % Caution:  u(i)
must never equal u(i-1)
end
DD=zeros(N,1);
DU=zeros(N,1);
DL=zeros(N,1);
F=zeros(N,1);
DD(1)=1;
DU(2)=0;
DD(N)=1/dx;
```

```
DL(N-1)=-1/dx;
F(1)=(2*qo+dx)/(h_exact(1)+h_exact(2)); % upstream condition
F(N)=h(N)^ 3; % downstream condition
%
for i=2:N-1
DD(i)= (-c(i+1)-c(i))/dx^ 2;
DL(i-1)= c(i)/dx^ 2;
DU(i+1)= c(i+1)/dx^ 2;
F(i)=(h(i+1)^ 2-h(i)^ 2)/dx;
end
D=spdiags([DL DD DU],[-1 0 1],N,N);
u=D\F;
end
%
% Now do ice-thickness update:
%
% Construct and solve Ah=R:
%
AD=zeros(N,1);
AU=zeros(N,1);
AL=zeros(N,1);
R=zeros(N,1);
AD(1)=1;
AU(2)=0;
R(1)=Ho/Z;
AD(N)=1/dt + 1/(2*dx)*(2*u(N)-u(N-1));
AL(N-1)= -u(N-1)/(2*dx);
R(N)=1+h(N)/dt;
%AD(N)=1/dx; %Alternative ice front
%AL(N-1)=-1/dx; % Boundary condition
%R(N)=0;
%
for i=2:N-1
AD(i)=1/dt + 1/(2*dx)*(u(i)-u(i-1));
AL(i-1)= -u(i-1)/(2*dx);
AU(i+1)= u(i)/(2*dx);
R(i)=h(i)/dt +1;
```

```
end
A=spdiags([AL AD AU],[-1 0 1],N,N);
h=A\R;
if rem(j,25) == 1
subplot(2,1,1); plot(x*L, u*U*3.155e7);subplot(2,1,2);plot(x*L,Z*h)
end
%
end % end time loop
```

In the next chapter, we shall investigate how to put the approach to ice-shelf modelling developed here to use in circumstances where the plan-view geometry of the ice shelf is arbitrary.

4.4 A Question of Mass Balance

Not all plausible finite-difference schemes perform satisfactorily in the simulation of the ice tongue. The scheme developed above (the staggered-grid scheme) has the virtues of reproducing the exact, analytic solution and of mass conservation; it fails, however, to capture the shock-like nature of the ice front as it passes down the grid after start-up from a zero-thickness initial condition. This inadequacy of the above scheme is displayed in Fig. (4.3), and in greater detail in Fig. (4.4).

Conservation of mass is demonstrated by the staggered-grid scheme by comparing the flux at the downstream end of the grid with its exact counterpart. The flux determined at the downstream end is determined by the following MATLAB code fragment:

```
   % Flux diagnostics
flux=zeros(N,1);
for i=1:N-1
flux(i)=u(i)*(h(i)+h(i+1))/2;
end
flux(N)=flux(N-1)+dx;
```

The computed value is 4.0139×10^5 m^2 a^{-1}. The exact value is 4.0273×10^5 m^2 a^{-1}. A non-staggered finite-difference scheme gives a flux of 4.0907×10^5 m^2 a^{-1}.

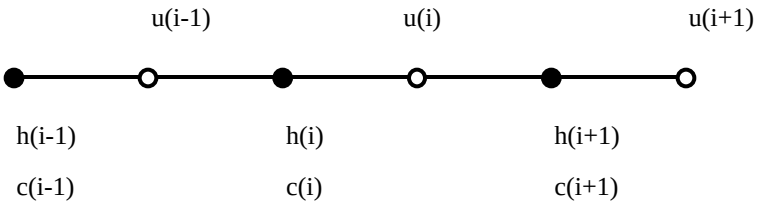

Figure 4.1: staggered grid convention used to model the ice tongue.

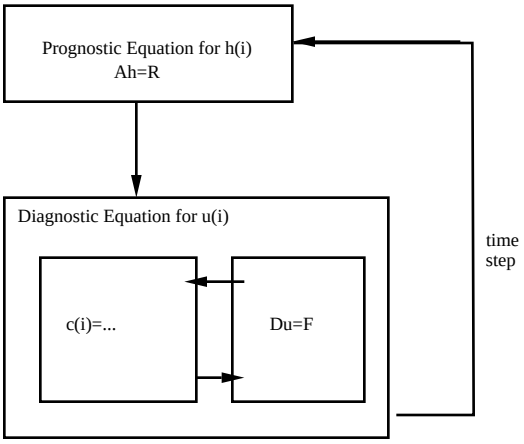

Figure 4.2: Two-step time-stepping procedure for ice-shelf models.

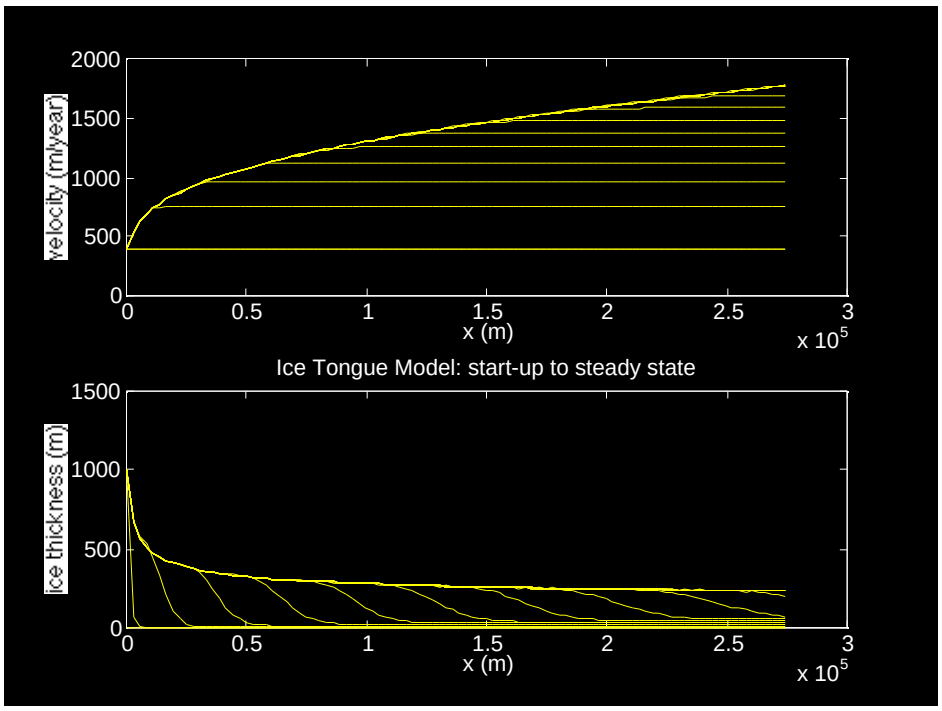

Figure 4.3: Ice tongue evolution from a 1-meter initial condition to steady state (at 1500 years). Ice thickness profile $h(x)$ (lower, dimensional form) and horizontal velocity profile $u(x)$ (upper, dimensional form) are plotted at 25-year intervals. The length of the ice tongue is 1 nondimensional unit (equivalent to 274 km). The exact, analytic steady-state ice-thickness profile determined in the previous chapter [Van der Veen, 1986a,b] is denoted in the lower plot by the crosses. Notice that an ice-thickness shock propagates down the ice tongue as the flux across the grounding line (which starts instantaneously at $t = 0$) makes it's presence known to the ice tongue. Due to numerical diffusion, this shock broadens with time. In nature, such a shock would propagate without a change in form (*i.e.,* it would remain steep). The ice velocity ahead of the shock remains relatively constant because the small ice thickness ahead of the shock introduces little horizontal spreading in the ice tongue.

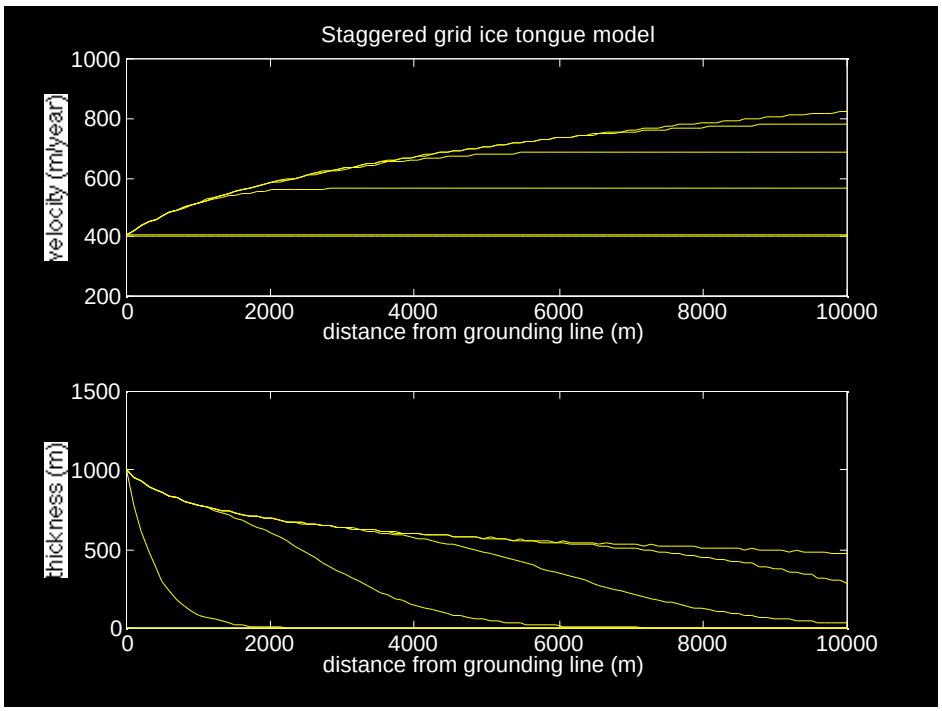

Figure 4.4: Ice tongue evolution from a 1-meter initial condition to steady state using a staggered grid finite-difference scheme (see Fig. **??**). Ice thickness profile $h(x)$ (lower, dimensional form) and horizontal velocity profile $u(x)$ (upper, dimensional form) are plotted at 5-year intervals. The length of the ice tongue is 10km. The exact, analytic steady-state ice-thickness profile determined in the previous chapter [Van der Veen, 1986a,b] is denoted in the lower plot by the dots (to which the steady state profile closely resembles). Notice that an ice-thickness shock propagates down the ice tongue. Due to numerical diffusion, this shock broadens with time. In nature, such a shock would propagate without a change in form (*i.e.,* it would remain steep). The ice velocity ahead of the shock remains relatively constant because the small ice thickness ahead of the shock introduces little horizontal spreading in the ice tongue.

Chapter 5

Two-Dimensional (Plan View) Ice-Shelf Models

In this chapter we will construct a finite-element model of a two dimensional (plan view) ice shelf of possibly arbitrary geometry. Our effort will be directed towards simulating the idealized ice-shelf geometry suggested by the EISMINT Level 1 exercises: an ice shelf confined by a rectangular (plan view) embayment, into which an ice stream discharges. This geometry will be modelled using only the finite-element approach developed in previous chapters. A finite-difference approach is similar to the finite-element approach, so will be left as an exercise for the interested reader.

The construction of a numerical model capable of simulating ice-shelf flow in an arbitrary, confined geometry is tedious because of the two-step time-stepping procedure. We shall attempt to alleviate the tedium of ice-shelf model construction by breaking up the tasks into smaller units. The first task we will consider is the solution of the diagnostic equations needed to obtain the ice-shelf velocity field from the ice shelf's current thickness. After considering the solution of the diagnostic equations, we will focus on the prognostic (mass balance) equation; and again implement a finite-element approach. Following this, we will tackle the problem of hooking the two parts of the model (prognostic and diagnostic) together to allow a time-stepping

simulation of ice-shelf evolution.

5.1 Nondimensional Form of the Diagnostic Equations

The nondimensional forms of the two diagnostic equations which represent stress-equilibrium in an ice shelf are constructed by using the definition of scales U, L, and Z developed in § (3.6) on Eqns. (3.5) - (3.7).

$$\frac{\partial}{\partial x}\left(\nu h\left(\dot{e}_{xx}+\frac{1}{2}\dot{e}_{yy}\right)\right)+\frac{\partial}{\partial y}\left(\frac{\nu h}{2}\dot{e}_{xy}\right)-\frac{\partial h^2}{\partial x} = 0 \tag{5.1}$$

$$\frac{\partial}{\partial y}\left(\nu h\left(\dot{e}_{yy}+\frac{1}{2}\dot{e}_{xx}\right)\right)+\frac{\partial}{\partial x}\left(\frac{\nu h}{2}\dot{e}_{yx}\right)-\frac{\partial h^2}{\partial y} = 0 \tag{5.2}$$

where $\dot{e}_{xx}=\frac{\partial u}{\partial x}$, $\dot{e}_{yy}=\frac{\partial v}{\partial y}$, $\dot{e}_{xy}=\dot{e}_{yx}=\frac{1}{2}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)$, and where the nondimensional effective viscosity ν is defined by

$$\nu=\left[\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2+\frac{1}{4}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2+\frac{\partial u}{\partial x}\frac{\partial v}{\partial y}\right]^{\frac{1-n}{2n}} \tag{5.3}$$

Dynamic boundary conditions in nondimensional form

The boundary condition to be applied at ice-shelf/ocean boundaries is, in nondimensional form, given by

$$\nu h\left(\dot{\mathbf{e}}\cdot\mathbf{n}+\left(\dot{e}_{xx}+\dot{e}_{yy}\right)\mathbf{n}\right)=2h^2\mathbf{n} \tag{5.4}$$

where $\mathbf{n}$ is the outward pointing normal vector to the boundary $\delta\Omega$. Each component of the above vector-valued equation is given by

$$\nu h\left(\left(2\dot{e}_{xx}+\dot{e}_{yy}\right)n_x+\dot{e}_{xy}n_y\right)=2h^2n_x \tag{5.5}$$

$$\nu h\left(\left(2\dot{e}_{yy}+\dot{e}_{xx}\right)n_y+\dot{e}_{yx}n_x\right)=2h^2n_y \tag{5.6}$$

$$\tag{5.7}$$

where n_x and n_y are the x and y components of $\mathbf{n}$.

5.2 Galerkin form of the Diagnostic Equations

Following the approach described in § (2.3), we use the Galerkin method (method of weighted residuals) to define what we consider to be a solution of Eqns. (5.1) - (5.3). For the x component of the momentum balance (Eqn. 5.1), we require that

$$\int\limits_{\Omega}\int \psi(x,y)\left\{\frac{\partial}{\partial x}\left(\nu h\left(\dot{e}_{xx}+\frac{1}{2}\dot{e}_{yy}\right)\right)+\frac{\partial}{\partial y}\left(\frac{\nu h}{2}\dot{e}_{xy}\right)-\frac{\partial h^2}{\partial x}\right\}dxdy=0 \tag{5.8}$$

where ψ is an arbitrary weighting function and Ω is the plan-view domain of the ice shelf. Use of the divergence theorem converts the above equation to the following form:

$$\begin{aligned}\int\limits_{\Omega}\int &\left\{\nu h\left(\dot{e}_{xx}+\frac{1}{2}\dot{e}_{yy}\right)\frac{\partial\psi}{\partial x}+\frac{\nu h}{2}\dot{e}_{xy}\frac{\partial\psi}{\partial y}-h^2\frac{\partial\psi}{\partial x}\right\}dxdy\\ &-\oint\limits_{\delta\Omega}\psi\left\{\nu h\left(\dot{e}_{xx}+\frac{1}{2}\dot{e}_{yy}\right)n_x+\frac{\nu h}{2}\dot{e}_{xy}n_y-h^2n_x\right\}ds\\ &=0\end{aligned} \tag{5.9}$$

where $\delta\Omega$ is the boundary of the ice shelf.

We recognize the boundary contour integral in Eqn. (5.9) as nothing more than a restatement of the dynamic boundary condition Eqn. (5.5). (Where kinematic conditions are applied, we assume that ψ is no longer arbitrary; thus we may enforce $\psi = 0$ on portions of $\delta\Omega$ where kinematic conditions are applied.) We are thus left with the following statement of what the solution to the diagnostic equations (the x component) should satisfy:

$$\int\limits_{\Omega}\int \left\{\nu h\left(\dot{e}_{xx}+\frac{1}{2}\dot{e}_{yy}\right)\frac{\partial\psi}{\partial x}+\frac{\nu h}{2}\dot{e}_{xy}\frac{\partial\psi}{\partial y}-h^2\frac{\partial\psi}{\partial x}\right\}dxdy=0 \tag{5.10}$$

A similar analysis of the y component of the diagnostic equations yields

$$\int_{\Omega} \int \left\{ \nu h \left(\dot{e}_{yy} + \frac{1}{2} \dot{e}_{xx} \right) \frac{\partial \chi}{\partial y} + \frac{\nu h}{2} \dot{e}_{yx} \frac{\partial \chi}{\partial x} - h^2 \frac{\partial \chi}{\partial y} \right\} dxdy = 0 \tag{5.11}$$

where $\chi(x, y)$ is another arbitrary weighting function similar to ψ.

5.2.1 Finite-Element Algorithm: Diagnostic Equations

Following the approach described previously in § (2.3), we write the finite-element form of the diagnostic equations (Eqns. 5.10 and 5.11 subject to the nonlinear flow law given by 5.3) as a matrix equation:

$$\lim_{k \to \infty} \mathbf{D}^{[k]} \mathbf{u}^{[k]} = \mathbf{F} \tag{5.12}$$

where the vector of nodal velocities $\mathbf{u} \in \mathcal{R}^{2M}$ is defined by

$$\mathbf{u} = \begin{bmatrix} u_1^{[k]} \\ v_1^{[k]} \\ u_2^{[k]} \\ v_2^{[k]} \\ \vdots \\ u_M^{[k]} \\ v_M^{[k]} \end{bmatrix} \tag{5.13}$$

where M is the number of nodes in the finite-element mesh, and where $[k]$ denotes the iteration index of the iteration scheme used to ensure satisfaction of the nonlinear flow law. Using the linear interpolation functions defined in Eqn. (2.19), the matrix-stuffing conventions are listed as follows

$$a^e c_e^{[k]} \left(\alpha_i \alpha_j + \frac{1}{4} \beta_i \beta_j \right) \quad \rightarrow \quad D_{2\gamma(j)-1, 2\gamma(i)-1}$$

$$a^e c_e^{[k]} \left(\frac{1}{2} \beta_i \alpha_j + \frac{1}{4} \alpha_i \beta_j \right) \quad \rightarrow \quad D_{2\gamma(j)-1, 2\gamma(i)}$$

$$
\begin{aligned}
a^e c_e^{[k]} \left(\frac{1}{2}\alpha_i\beta_j + \frac{1}{4}\beta_i\alpha_j \right) &\rightarrow D_{2\gamma(j),2\gamma(i)-1} \\
a^e c_e^{[k]} \left(\beta_i\beta_j + \frac{1}{4}\alpha_i\alpha_j \right) &\rightarrow D_{2\gamma(j),2\gamma(i)} \\
a^e (h^e)^2 \alpha_j &\rightarrow F_{2\gamma(j)-1} \\
a^e (h^e)^2 \beta_j &\rightarrow F_{2\gamma(j)}
\end{aligned}
$$

(summation is required over all elements $e = 1, \ldots, N$, and over $i = 1, 2, 3$ and $j = 1, 2, 3$ within each element, where i and j are indices representing the local, intra-element number of the 3 vertices of each triangular element), where a^e is the area of element e, where

$$
h^e = \frac{h_{\gamma(1)} + h_{\gamma(2)} + h_{\gamma(3)}}{3} \tag{5.14}
$$

is the average thickness in element e, and

$$
c_e^{[k]} = h^e \left(\left(u_{\gamma(i)}^{[k-1]} \alpha_i \right)^2 + \left(v_{\gamma(i)}^{[k-1]} \beta_i \right)^2 + \frac{1}{4} \left(u_{\gamma(i)}^{[k-1]} \beta_i + v_{\gamma(i)}^{[k-1]} \alpha_i \right)^2 + u_{\gamma(i)}^{[k-1]} v_{\gamma(j)}^{[k-1]} \alpha_i \beta_j \right)^{\frac{-1}{3}} \tag{5.15}
$$

(summation for each elemental value of $c_e^{[k]}$ is over vertex indices $i = 1, 2, 3$ and $j = 1, 2, 3$) and where $\gamma(i)$ is the global node number of the i'th vertex of element e. The above equation for $c_e^{[k]}$ represents the recursion formula that is used to generate $\mathbf{D}^{[k]}$ from the velocity field $\mathbf{u}^{[k-1]}$ at iteration number $[k-1]$.

As done with the ice-tongue model in § (4.1), an initial guess of the velocity field $\mathbf{u}^{[0]}$ is used to generate $c^{[1]}$ for each element. The matrix equation $\mathbf{D}^{[1]}\mathbf{u}^{[1]} = \mathbf{F}$ is then solved for $\mathbf{u}^{[1]}$. The resulting $\mathbf{u}^{[1]}$ is used to generate $c^{[2]}$ and so on until the sequence of $c^{[k]}$'s converges. The convergence criterion is

$$
\max_{e=2,\ldots,N} \left| \frac{c_e^{[k+1]} - c_e^{[k]}}{c_e^{[k]}} \right| < \delta \tag{5.16}
$$

where δ is a small number of the order of 0.01, and N is the number of elements.

5.3 Velocity Solution (Fixed Ice Thickness)

The MATLAB script listed below is used to implement the above finite-element algorithm. Figure (5.1) displays the finite-element mesh used to simulate the EISMINT Level 1 test. The resolution of this mesh is 5 km, and the adjacency matrix associated with this mesh (which reveals the sparseness of the matrix $\mathbf{D}^{[k]}$) is displayed in Fig. (5.2).

Figures (5.3) - (5.6) display the velocity field of the ice shelf produced by the finite-element method. As mentioned previously, the velocity field displayed in Figs. (5.3) - (5.6) represents the instantaneous flow associated with an ice thickness of 1000 m. (Time-dependent evolution of the velocity field in concert with a changing ice thickness field is considered in a later section.) The ice velocity at the ice front reaches a maximum of over 12 km/year. This very large velocity reflects the fact that ice shelves of 1000 m thickness have a very large spreading rate unless confined by ice rises or longer embayments. The effect of the input ice stream is seen to be relatively minor.

MatLab script for solution of velocity equations

This script takes advantage of the fact that the ice-shelf thickness is uniform (1000 m). In circumstances where the ice thickness is not uniform, the elemental interpolation functions for ice thickness must be accounted for in the area integrations which discretize the problem. Also note that the computations necessary to determine the row and column index ordering (vectors `row` and `col`) of the sparse matrix $\mathbf{D}^{[k]}$ are done once at the beginning of the script.

```
% This script represents a finite-element model of an ice shelf
%
% EISMINT Level 1 ice shelf test.
%
Qo=4e5/31556926;
Ho=1.0e3;
```

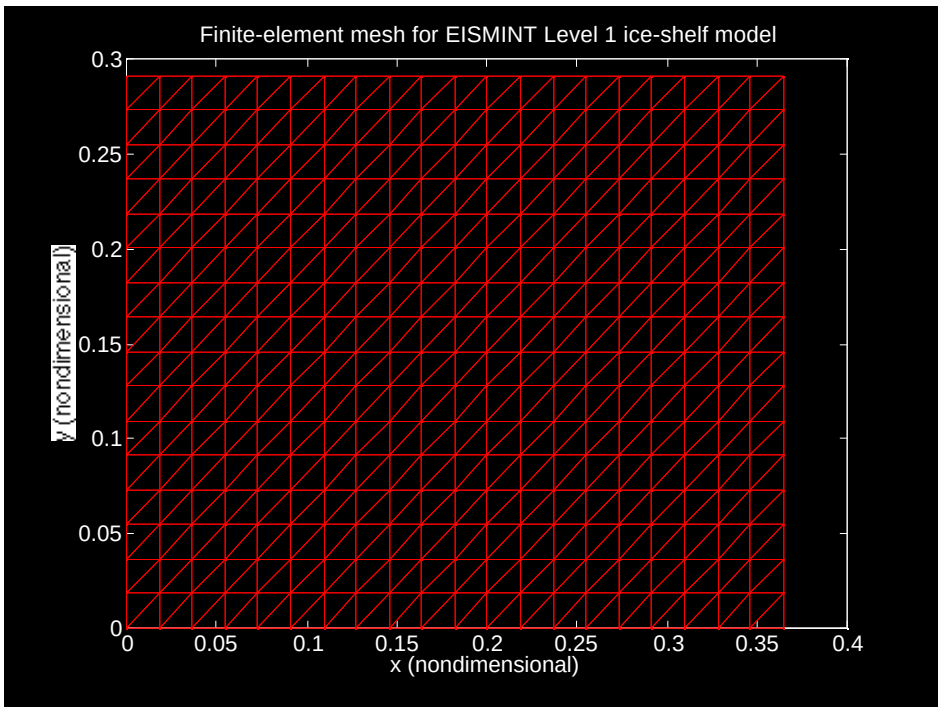

Figure 5.1: The finite-element mesh (shown using nondimensional x and y coordinates) for the EISMINT Level 1 ice-shelf test. Mesh resolution corresponds to 5 km. The kinematic boundary condition associated with ice-stream input (1000 m thickness, 400 m/year velocity) is specified on the right-most 4 nodes of the top boundary. The ice front corresponds to the lower boundary. The right boundary is an axis of symmetry across which there are no gradients in longitudinal velocity. The left boundary, and portion of the top boundary not corresponding to the inflowing ice stream, have zero velocity (no slip, no normal flow) boundary conditions specified.

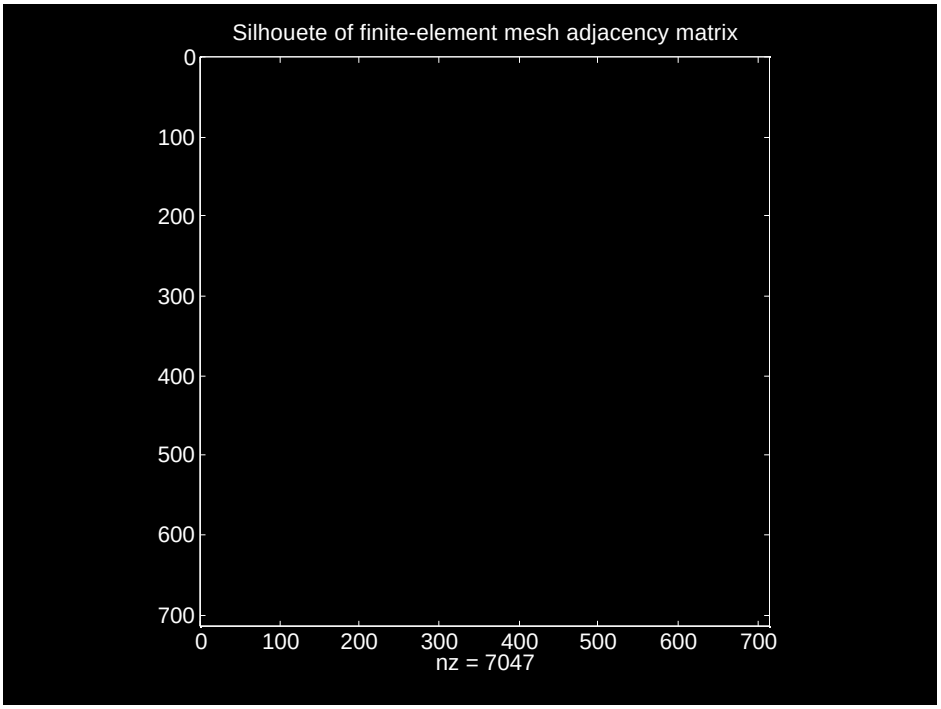

Figure 5.2: The silhouette of the adjacency matrix associated with the finite-element mesh of the ice shelf displayed in Fig. (5.1). As can be readily appreciated, the matrix form of the finite-element problem involves a sparse matrix having relatively few nonzero elements that are clustered along the diagonal.

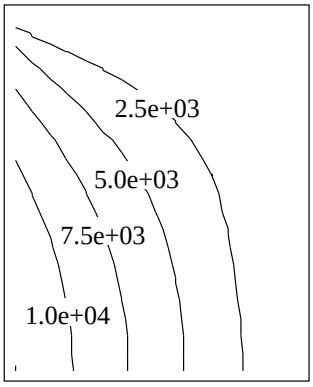

Figure 5.3: Velocity magnitude (m/year) for the 1000 m thick ice shelf. The SPYGLASS contour plot has rotated the image of the domain by 90-degrees, thus the top boundary of Fig. (5.1) corresponds with the right boundary displayed here. (The left boundary is thus the ice front, and the lower boundary is the axis of symmetry.)

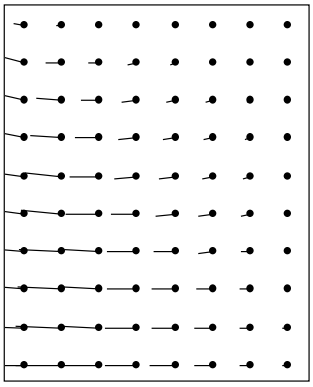

Figure 5.4: Velocity vectors for the 1000 m thick ice shelf. The SPYGLASS contour plot has rotated the image of the domain by 90-degrees, thus the top boundary of Fig. (5.1) corresponds with the right boundary displayed here. (The left boundary is thus the ice front, and the lower boundary is the axis of symmetry.)

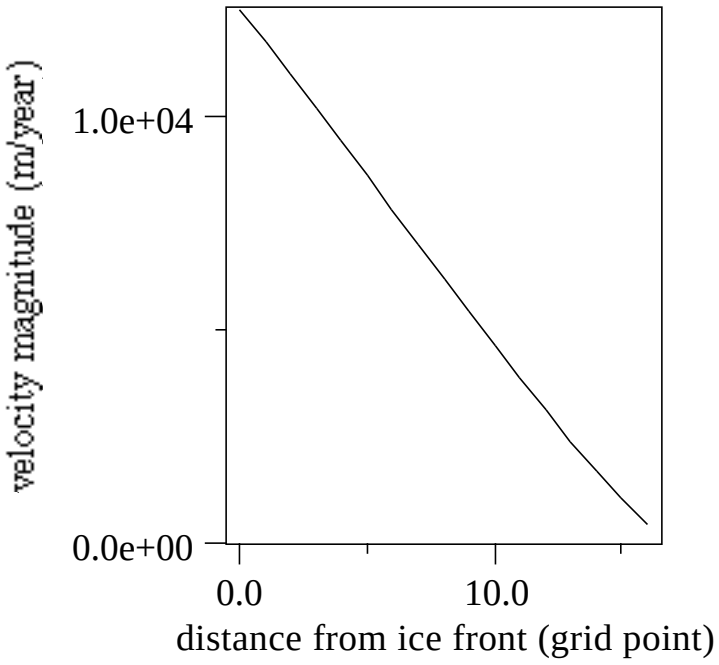

Figure 5.5: Velocity magnitude (m/year) along the longitudinal cross section that extends down the centerline of the ice-shelf domain (corresponding to the axis of symmetry).

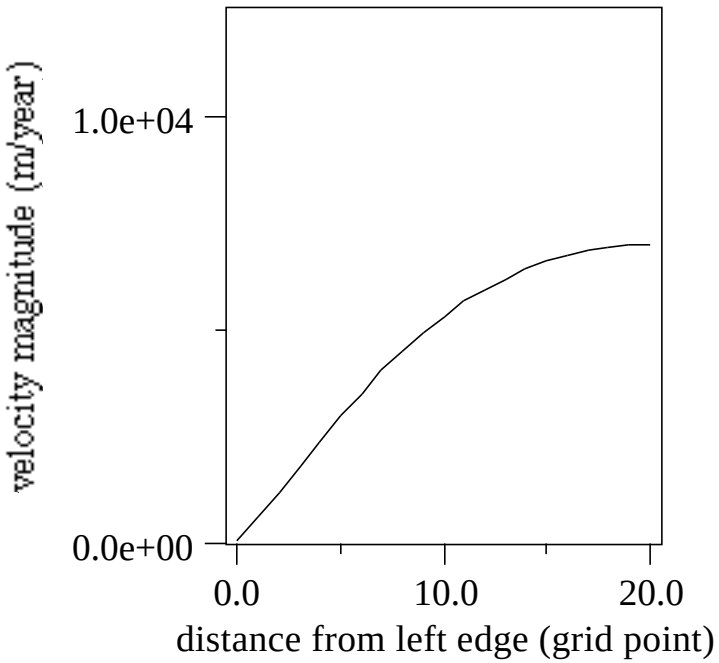

Figure 5.6: Velocity magnitude (m/year) along the transverse cross section that extends across the ice-shelf domain at the midline.

```
rho=910;
rho_w=1028;
a=0.3/31556926;
Bo=1.4688e8;
g=9.81;
Z=a^ (1/4)*(4*Bo)^ (3/4)/(rho*g*(1-rho/rho_w))^ (3/4);
UonL=(rho*g*Z*(1 - rho/rho_w)/(4*Bo))^ 3;
U=400/31556926;
L=Z*U/a;
qo=Qo/(U*Z);
ho=Ho/Z;
imax=21;
jmax=17;
nodes=imax*jmax;
nel=(imax-1)*(jmax-1)*2;
nrows=23040;
row=zeros(nrows,1);
col=zeros(nrows,1);
value=zeros(nrows,1);
phi=zeros(3,3);
% Initialize at zero ice velocity
ugrid=zeros(imax,jmax);
vgrid=zeros(imax,jmax);
u=zeros(nodes*2,1);
% Initialize
h=10^ 3/Z*ones(nodes,1);
havg=10^ 3/Z;
% Create the interpolation functions.
area=zeros(nel,1);
alpha=zeros(nel,3);
beta=zeros(nel,3);
count=0;
%
for n=1:nel
%
[lowtri uptri]=lu([[xy(index(n,1),1) xy(index(n,2),1) xy(index(n,3),1)]'...
[xy(index(n,1),2) xy(index(n,2),2) xy(index(n,3),2)]' ones(3,1)]);
```

```
phi(:,1)=uptri\(lowtri\ [1 0 0]');
phi(:,2)=uptri\(lowtri\ [0 1 0]');
phi(:,3)=uptri\(lowtri\ [0 0 1]');
for k=1:3
alpha(n,k)=phi(1,k);
beta(n,k)=phi(2,k);
end
%
area(n)=abs(.5*det([1 1 1
xy(index(n,1),1) xy(index(n,2),1) xy(index(n,3),1)
xy(index(n,1),2) xy(index(n,2),2) xy(index(n,3),2)]));
% Perform loading of row and col arrays
for i=1:3
for j=1:3
row(count+1)=index(n,j)*2-1;
col(count+1)=index(n,i)*2-1;
row(count+2)=index(n,j)*2-1;
col(count+2)=index(n,i)*2;
row(count+3)=index(n,j)*2;
col(count+3)=index(n,i)*2-1;
row(count+4)=index(n,j)*2;
col(count+4)=index(n,i)*2;
count=count+4;
end
end
%
end
%
c=ones(nel,1);
c_old=zeros(nel,1);
loop=0;
while max((abs(c-c_old)./c)) > 0.01
loop=loop+1
%
value=zeros(nrows,1);
F=zeros(2*nodes,1);
count=0;
```

```
%
for n=1:nel
%
% Effective diffusivity:
%
if loop  =1
c_old(n)=c(n);
ux=0;
uy=0;
vx=0;
vy=0;
for i=1:3
ux=ux+u(index(n,i)*2-1)*alpha(n,i);
uy=uy+u(index(n,i)*2-1)*beta(n,i);
vx=vx+u(index(n,i)*2)*alpha(n,i);
vy=vy+u(index(n,i)*2)*beta(n,i);
end
if (ux^ 2+vy^ 2+((uy+vx)^ 2)/4 +ux*vy ) > 10^ (-15)
c(n)=havg*(ux^ 2+vy^ 2+((uy+vx)^ 2)/4 +ux*vy )^ (-1/3);
else
c(n)=havg*10^ 5;
end
end
% Load right-hand-side vector:
%
for k=1:3
F(2*index(n,k)-1)=F(2*index(n,k)-1)+area(n)*havg^ 2*alpha(n,k);
F(2*index(n,k))=F(2*index(n,k))+area(n)*havg^ 2*beta(n,k);
end


Notice the mistake in the preceeding two statements
It is better to use
for m=1:3
for n=1:nel
F(index(n,m)*2-1)=F(index(n,m)*2-1)+alpha(n,m)*rho*g/2*(1-rho/rhowater)*hsq(n);
% x
```

```
F(index(n,m)*2)=F(index(n,m)*2)+beta(n,m)*rho*g/2*(1-rho/rhowater)*hsq(n);
% y
end
end
where
hsq=area(:).*(h(index(:,1)).^2+h(index(:,2)).^2+h(index(:,3)).^2)/6
...
+ h(index(:,1)).*(h(index(:,2))+h(index(:,3)))/12 ...
+ h(index(:,2)).*(h(index(:,1))+h(index(:,3)))/12 ...
+ h(index(:,3)).*(h(index(:,1))+h(index(:,2)))/12;
```

The above mistake was caught after the results of this chapter were printed
DRM, August 10, 1995

```
%
% Load matrix:
%
for i=1:3
for j=1:3
value(count+1)=area(n)*c(n)*(alpha(n,i)*alpha(n,j)+beta(n,i)*beta(n,j)/4);
value(count+2)=area(n)*c(n)*(beta(n,i)*alpha(n,j)/2+alpha(n,i)*beta(n,j)/4);
value(count+3)=area(n)*c(n)*(alpha(n,i)*beta(n,j)/2+beta(n,i)*alpha(n,j)/4);
value(count+4)=area(n)*c(n)*(beta(n,i)*beta(n,j)+alpha(n,i)*alpha(n,j)/4);
count=count+4;
end
end
% End loop over elements
end
D=sparse(row,col,value);
% Boundary conditions
bxcount=0;
bycount=0;
for j=1:(jmax-1) % left side
bxcount=bxcount+1;
D(Boundu(bxcount)*2-1,Boundu(bxcount)*2-1)=10^ 12;
```

```
F(Boundu(bxcount)*2-1)=0;
bycount=bycount+1;
D(Boundv(bycount)*2,Boundv(bycount)*2)=10^ 12;
F(Boundv(bycount)*2)=0;
end
for i=1:imax % top side
bxcount=bxcount+1;
bycount=bycount+1;
D(Boundu(bxcount)*2-1,Boundu(bxcount)*2-1)=10^ 12;
F(Boundu(bxcount)*2-1)=0;
D(Boundv(bycount)*2,Boundv(bycount)*2)=10^ 12;
F(Boundv(bycount)*2)=0;
end
for j=1:(jmax-1) % right side
bxcount=bxcount+1;
D(Boundu(bxcount)*2-1,Boundu(bxcount)*2-1)=10^ 12;
F(Boundu(bxcount)*2-1)=0;
end
F(gamma(18,17)*2)=-10^ 12;
F(gamma(19,17)*2)=-10^ 12;
F(gamma(20,17)*2)=-10^ 12;
F(gamma(21,17)*2)=-10^ 12;
% Solve the system for new velocity
u=D\F;
for i=1:imax
for j=1:jmax
ugrid(i,j)=u(gamma(i,j)*2-1);
vgrid(i,j)=u(gamma(i,j)*2);
end
end
end % end While statement
```

MatLab script used to generate the finite-element mesh

For completeness, we list the MATLAB script used to generate the finite-element mesh used in the previous test of the ice-shelf diagnostic equation solver as follows:

```
% This script creates the mesh for the EISMINT Level 1
% ice shelf test.
nodes=17*21;
imax=21;
jmax=17;
nrows=17*21*2;
row=zeros(nrows,1);
col=zeros(nrows,1);
value=zeros(nrows,1);
xy=zeros(nrows,2);
gamma=zeros(imax,jmax);
Boundu=zeros(53,1); % zero x-velocity nodes
Boundv=zeros(37,1); % zero y-velocity nodes
count=0;
dx=100e3/L; % L is the nondimensional length scale
dy=80e3/L;
for i=1:imax
for j=1:jmax
count=count+1;
xy(count,1)=(i-1)/(imax-1)*dx;
xy(count,2)=(j-1)/(jmax-1)*dy;
gamma(i,j)=count;
end
end
% Create triangulation
%
nel=(imax-1)*(jmax-1)*2;
index=zeros(nel,3);
count=0;
%
```

```
for i=1:imax-1
for j=1:jmax-1
count=count+1;
index(count,1)=gamma(i,j);
index(count,2)=gamma(i+1,j);
index(count,3)=gamma(i+1,j+1);
count=count+1;
index(count,1)=gamma(i,j);
index(count,2)=gamma(i+1,j+1);
index(count,3)=gamma(i,j+1);
end
end
bxcount=0;
bycount=0;
for j=1:(jmax-1) % left side
bxcount=bxcount+1;
Boundu(bxcount)=gamma(1,j);
bycount=bycount+1;
Boundv(bycount)=gamma(1,j);
end
for i=1:imax % top side
bxcount=bxcount+1;
bycount=bycount+1;
Boundu(bxcount)=gamma(i,jmax);
Boundv(bycount)=gamma(i,jmax);
end
for j=1:(jmax-1) % right side
bxcount=bxcount+1;
Boundu(bxcount)=gamma(imax,j);
end
% Create adjacency matrix and plot mesh and silhouette of mesh
count=0;
pause
for n=1:nel
for i=1:3
for j=1:3
count=count+1;
```

```
row(count)=index(n,i)*2-1;
col(count)=index(n,j)*2-1;
value(count)=1;
count=count+1;
row(count)=index(n,i)*2-1;
col(count)=index(n,j)*2;
value(count)=1;
count=count+1;
row(count)=index(n,i)*2;
col(count)=index(n,j)*2-1;
value(count)=1;
row(count)=index(n,i)*2;
col(count)=index(n,j)*2;
value(count)=1;
end
end
end
% Construct mesh plot
Adj=sparse(row,col,value);
gplot(Adj,xy);
pause
spy(Adj)
```

5.4 Nondimensional Form of the Prognostic Equation

The above representation of the diagnostic (stress-equilibrium) equations yields the z-independent horizontal velocity field $\mathbf{u} = (u, v)$ that corresponds to an instantaneous *snap shot* of the ice-thickness field, $h(\mathbf{x}, t)$. The time evolution of the ice shelf is governed by the prognostic equation which specifies how h changes with time as a result of the divergence of the ice transport associated with nonzero $\mathbf{u}$ and the surface and basal accumulation rates (here lumped into one term). The nondimensional prognostic equation (Eqn. 3.1)

is written

$$\frac{\partial h}{\partial t} + \nabla \cdot (\mathbf{u}h) - 1 = 0 \tag{5.17}$$

where the surface and basal accumulation rates are assumed uniform (and of nondimensional magnitude 1).

The method of weighted residuals (Galerkin method) requires that

$$\int_{\Omega} \int \psi \left(\frac{\partial h}{\partial t} + \nabla \cdot (\mathbf{u}h) - 1 \right) dxdy = 0 \tag{5.18}$$

where ψ is an arbitrary weighting function. Integration by parts and the divergence theorem gives

$$\int_{\Omega} \int \left(\psi \frac{\partial h}{\partial t} - uh \frac{\partial \psi}{\partial x} - vh \frac{\partial \psi}{\partial y} - \psi \right) dxdy + \oint_{\delta\Omega} \psi \, \mathbf{u} \cdot \mathbf{n} ds = 0 \tag{5.19}$$

where $\mathbf{n}$ is the outward-pointing normal vector to the boundary of the ice-shelf domain $\delta\Omega$.

Boundary conditions are described as follows. At the inland boundaries where the ice shelf abuts stagnant inland ice or ice-free coastline,

$$h\mathbf{u} \cdot \mathbf{n} = \mathbf{q} \cdot \mathbf{n} = 0 \tag{5.20}$$

At the inland boundaries where the ice shelf joins with an ice stream, either a flux condition can be specified, *e.g.,*

$$\mathbf{q} \cdot \mathbf{n} = q_o \tag{5.21}$$

or a fixed ice thickness can be specified, *e.g.,* [1]he boundary integral on the right-hand side of Eqn. (6.3) allows the possibility of two types of boundary conditions at boundaries where ice streams flow into the ice shelf. This point will be emphasized in a later chapter where the discussion of mass conservation of the EISMINT tests is given. Either the ice thickness or the mass flux can be specified, not both. In the former case, the mass flux becomes an unknown flux of constraint (output of the model) that is needed

[1] T

to enforce the fixed ice thickness. In the latter case, the ice thickness varies in response to the mass flux.

$$h = h_o \tag{5.22}$$

At the ice front, where the ice shelf calves into the ocean, a free-radiation condition must be applied to avoid having ice "pile up" at the ice front, *e.g.*,

$$\mathbf{q} \cdot \mathbf{n} = (h\mathbf{u})\Big|_{\_} \cdot \mathbf{n} \tag{5.23}$$

where $(h\mathbf{u})\Big|_{\_}$ represents the ice transport just upstream of the ice front.

5.4.1 Finite-Element Algorithm: Prognostic Equation

Following the methods outlined in § (2.3), we express Eqn. (6.3) as a matrix equation

$$\mathbf{A}\mathbf{h}^{n+1} = \mathbf{R} \tag{5.24}$$

where, $\mathbf{h}^{n+1}$ is a column vector containing the nodal values of the ice thickness at time step $n+1$:

$$\mathbf{h}^{n+1} = \begin{bmatrix} h_1^{n+1} \\ h_2^{n+1} \\ \vdots \\ h_M^{n+1} \end{bmatrix} \tag{5.25}$$

and where M is the number of nodes (recall that N is the number of elements). The matrix-stuffing conventions are listed as follows

$$\frac{1}{\Delta t}\begin{Bmatrix} \frac{a^e}{6} & k=l \\ \frac{a^e}{12} & k\neq l \end{Bmatrix} - (\alpha_k u_j + \beta_k v_j)\begin{Bmatrix} \frac{a^e}{6} & l=j \\ \frac{a^e}{12} & l\neq j \end{Bmatrix} \;\rightarrow\; A_{\gamma(k)\gamma(l)}$$

$$\frac{a^e}{3} + \frac{h_l^n}{\Delta t}\begin{Bmatrix} \frac{a^e}{6} & k=l \\ \frac{a^e}{12} & k\neq l \end{Bmatrix} - (q_o)_j \Delta l \begin{Bmatrix} \frac{1}{4} & k=l=j; k,l,j\neq s_3 \\ 0 & k=s_3, \text{ or } l=s_3, \text{ or } j=s_3 \\ & k\neq l=j \\ \frac{1}{12} & k=l\neq j \;; k,l,j\neq s_3 \\ & k=j\neq l \end{Bmatrix} \;\rightarrow\; R_{\gamma(k)}$$

where a^e is the area of element e, and $\Delta l = \sqrt{(x_{\gamma(s_1)} - x_{\gamma(s_2)})^2 + (y_{\gamma(s_1)} - y_{\gamma(s_2)})^2}$ is the length of the boundary line segment connecting nodes s_1 and s_2 of element e. In the above conventions, it is understood that summation over j and e is necessary for the creation of $\mathbf{A}$, and that summation over j, e, and l is necessary for the creation of $\mathbf{R}$. The indices k, l and j denote the local vertex numbers of triangular element e. The indices s_1, s_2 and s_3 also denote the local vertex numbers of triangular element e with the additional understanding that vertices s_1 and s_2 lie on the boundary $\delta\Omega$ whereas vertex s_3 does not. The number $\gamma(j)$ represent the global node number of the vertex j. The last part of the expression specifying the construction of $\mathbf{R}$ takes care of the boundary flux conditions where necessary (recall that, at the ice front, $(q_o)_j$ is equal to $h^n[\mathbf{u}]_j \cdot \mathbf{n}$). At boundary nodes where $h = h_o$ is specified, appropriate changes to the matrix $\mathbf{A}$ and vector $\mathbf{R}$ are made, *e.g.*, $A_{\gamma(k),\gamma(k)} = 1$ and $R_{\gamma(k)} = h_o$.

An alternative construction of $\mathbf{A}$ and $\mathbf{R}$ in which the ice-front radiation condition is evaluated implicitly is listed as follows:

$$\frac{1}{\Delta t}\begin{Bmatrix} \frac{a^e}{6} & k=l \\ \frac{a^e}{12} & k\neq l \end{Bmatrix} - (\alpha_k u_j + \beta_k v_j)\begin{Bmatrix} \frac{a^e}{6} & l=j \\ \frac{a^e}{12} & l\neq j \end{Bmatrix}$$

$$+[\mathbf{u}]_j \cdot \mathbf{n}\Delta l \begin{Bmatrix} \frac{1}{4} & k=l=j; k,l,j\neq s_3 \\ 0 & k=s_3, \text{ or } l=s_3, \text{ or } j=s_3 \\ & k\neq l=j \\ \frac{1}{12} & k=l\neq j \ ; k,l,j\neq s_3 \\ & k=j\neq l \end{Bmatrix} \rightarrow A_{\gamma(k)\gamma(l)}$$

$$\frac{a^e}{3} + \frac{h_l^n}{\Delta t}\begin{Bmatrix} \frac{a^e}{6} & k=l \\ \frac{a^e}{12} & k\neq l \end{Bmatrix} - (q_o)_j \Delta l \begin{Bmatrix} \frac{1}{4} & k=l=j; k,l,j\neq s_3 \\ 0 & k=s_3, \text{ or } l=s_3, \text{ or } j=s_3 \\ & k\neq l=j \\ \frac{1}{12} & k=l\neq j \ ; k,l,j\neq s_3 \\ & k=j\neq l \end{Bmatrix} \rightarrow R_{\gamma(k)}$$

where the q_o in the construction of $\mathbf{R}$ denotes boundary fluxes not specified

at the ice front.

In developing the matrix-stuffing conventions listed above, we have made use of the formulae for $\int_e \phi_k \phi_l dxdy$ developed in § (2.3.1). For the integrals over the ice front boundaries where freely radiating flux is specified, we must use the formula:

$$\oint_{\delta\Omega^e} \phi_k \phi_l \phi_j ds = \begin{cases} \frac{1}{4} & k = j = l;\ k, j, l \neq s_3 \\ 0 & k = s_3 \text{ or } j = s_3 \text{ or } l = s_3 \\ \frac{1}{12} & k = j \neq l, \text{ etc.} \end{cases} \tag{5.26}$$

where s_3 is the local index number of the one vertex of element e that does *not* form an endpoint of the boundary segment (side of triangle) $\delta\Omega^e$. These formulae are relatively simple to derive using the definitions for the linear interpolation functions ϕ_i, $i = 1, 2, 3$ found in § (2.3.1).

5.5 Thickness Solution (Fixed Velocity)

A MATLAB script (listed below) is used to step the prognostic (mass balance) equation through 150 years of evolution holding the ice velocity $\mathbf{u}$ constant. (This represents an artificial test because the ice velocity is not updated to reflect the changing ice-thickness conditions.) By holding the ice velocity fixed, we are able to test the portion of the ice-shelf model that is responsible for time evolution of the ice thickness. We shall construct and test a real ice shelf model, in which the diagnostic equations and prognostic equation are coupled together, in a section which follows the discussion here.

Several technical simplifications associated with this fixed-velocity test deserve mention. First is that the matrix $\mathbf{A}$ need only be constructed and LU-factored once no matter how many time steps are to be taken. (We note the fact that, unlike $\mathbf{D}$, the matrix associated with the diagnostic equations, $\mathbf{A}$ is not positive definite. We must thus use the less efficient LU-decomposition, with reversed Cuthill-McKee node ordering, instead of the Cholesky decomposition.) The matrix $\mathbf{A}$ involves (see the matrix-stuffing conventions summarized above) the velocity field $\mathbf{u}$ which, for this test only, does not change with changing ice thickness. The time-stepping procedure

thus involves merely the construction of the right-hand-side vector $\mathbf{R}$ and the application of the triangular factors of $\mathbf{A}$ in a forward substitution and back substitution to obtain $\mathbf{h}^{n+1}$. The MATLAB script which performs the time-dependent, fixed-velocity simulation is listed below:

```
% This script represents a finite-element model of an ice shelf.
% (with it's velocity held fixed!)
% EISMINT Level 1
% ice shelf test.
nrows=5920;
row=zeros(nrows,1);
col=zeros(nrows,1);
% Initialize
h=10^ 3/Z*ones(nodes,1);
count=0;
value=zeros(nrows,1);
for n=1:nel
%
% Perform loading of row and col arrays
%
% Load matrix:  This matrix doesn't change due to
% fixed ice velocity (this example only)
%
for l=1:3
for k=1:3
count=count+1;
row(count)=index(n,k);
col(count)=index(n,l);
if k == l
value(count)=area(n)/(6*dt);
else
value(count)=area(n)/(12*dt);
end
for j=1:3
if j == l
value(count)=value(count)-(alpha(n,k)*u(index(n,j)*2-1)+beta(n,k)*u(index(n,j)*2))
area(n)/6;
```

```
else
value(count)=value(count)-(alpha(n,k)*u(index(n,j)*2-1)+beta(n,k)*u(index(n,j)*2))
area(n)/12;
end
end
end
end
%
end % end loop over elements
%
% Boundary conditions
% Ice front contour:  Implicit case
%
for n=1:imax-1
for k=1:2
for l=1:2
for j=1:2
count=count+1;
row(count)=icefnt(n,k);
col(count)=icefnt(n,l);
if l==k & j==k
value(count)=-u(icefnt(n,j)*2)*dl/4;
else
value(count)=-u(icefnt(n,j)*2)*dl/12;
end
end
end
end
end
%
A=sparse(row,col,value);
% Fix the input ice thickness to 1
%
A(gamma(18,17),gamma(18,17))=10^ 15;
A(gamma(19,17),gamma(19,17))=10^ 15;
A(gamma(20,17),gamma(20,17))=10^ 15;
A(gamma(21,17),gamma(21,17))=10^ 15;
```

```
%
% Perform LU decomposition once (since velocity
% doesn't change, the matrix A doesn't change)
%
[LowTri,UpTri]=lu(A);
%
dl=xy(icefnt(1,2),1)-xy(icefnt(1,1),1);
nsteps=1500;
dt=.1*31556926*a/Z;
history=zeros(nsteps,1);
%
for time=1:nsteps
history(time)=h(gamma(21,1));
time
%
R=zeros(nodes,1);
%
for n=1:nel
%
% Load right-hand-side vector:
%
R(index(n,1))=R(index(n,1))+area(n)/3;
R(index(n,2))=R(index(n,2))+area(n)/3;
R(index(n,3))=R(index(n,3))+area(n)/3;
for k=1:3
for j=1:3
if j == k
R(index(n,k))=R(index(n,k))+area(n)*h(index(n,j))/(6*dt);
else
R(index(n,k))=R(index(n,k))+area(n)*h(index(n,j))/(12*dt);
end
end
end
% End loop over elements
end
% Ice stream input condition:
R(gamma(18,17))=10^ 3/Z*10^ 15;
```

```
R(gamma(19,17))=10^ 3/Z*10^ 15;
R(gamma(20,17))=10^ 3/Z*10^ 15;
R(gamma(21,17))=10^ 3/Z*10^ 15;
% Solve the system for new thickness
h=UpTri\(LowTri\R);
%
end % end timestep loop
for i=1:imax
for j=1:jmax
hgrid(i,j)=h(gamma(i,j))*Z;
end
end
```

The ice-front index array `icefnt` is dimensioned `imax-1` by 2, and holds the global node numbers for the endpoints of each of the line segments that form the ice-front boundary. This index array was created in the mesh-generation program (the same as that used to generate the mesh used to solve the diagnostic equations in the previous sections) with the following fragment of additional MATLAB code:

```
% Create ice front boundary contour;
ifcount=0
for i=1:imax-1
ifcount=ifcount+1;
icefnt(ifcount,1)=gamma(i,1);
icefnt(ifcount,2)=gamma(i+1,1);
end
```

5.5.1 Unsatisfactory Results

The above listed MATLAB script was run to represent the time-evolution of the fixed-velocity ice shelf through 150 years using 0.1-year time steps (in dimensional units) starting with a uniform 1000-m ice thickness. (A comment

about the choice of time step size is given below.) The equilibration of the thickness at the ice-front node corresponding to the point where the axis of symmetry (the ice shelf's longitudinal centerline) intersects the ice front is demonstrated in Fig. (5.7). The ice-thickness field, h, at the end of the 150-year evolution is displayed as a contour map in Fig. (5.8), and in both a longitudinal section down the axis of symmetry (Fig. 5.9) and a transverse section across the midline of the ice shelf (Fig. 5.10).

A problem is clearly evident in the solution displayed in Figs. (5.8) - (5.10). The ice thickness is less than zero in some locations, and is unacceptably wiggly in others. (We expect the solution to be smooth, and not to exhibit node-to-node oscillations in any of the fields.) Another, minor defect is the fact that the thickness has grown unacceptably large at the node point corresponding to the corner where the two stagnant, no-flux boundaries meet (corresponding to the upper right-hand corner in Fig. 5.8). This minor defect results from the mesh discretization which places a triangle in this corner who's three vertices have zero velocity (thus the accumulation in the corner has now way to get out).

The unsatisfactory results of this test serve to illustrate the difficult nature of simulating hyperbolic partial differential equations with a technique that does not make use of the characteristics of the solution (the trajectories along which information is propagated). There are many approaches to overcoming the difficulties illustrated here, and we shall investigate several.

5.5.2 Positive-Definite Ice Thickness

The first remedy we will try to correct the defects of the finite-element solution illustrated above is to insist that the ice thickness never be less than zero. Thus we require that the prognostic equation

$$\frac{\partial h}{\partial t} + \nabla \cdot (\mathbf{u}h) - 1 = 0 \tag{5.27}$$

be satisfied subject to the constraint

$$h \geq 0 \tag{5.28}$$

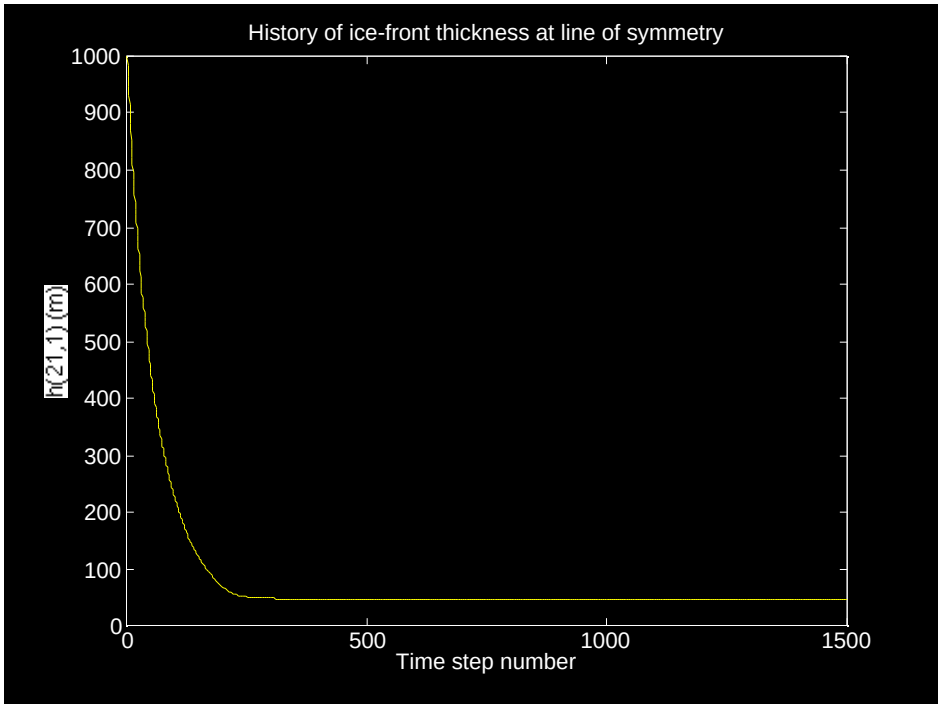

Figure 5.7: Ice thickness (dimensional units) at the ice front at the point where the ice front intersects the midline (axis of symmetry) of the ice shelf as a function of time. Equilibration is complete after about 250 time steps, corresponding to 25 years.

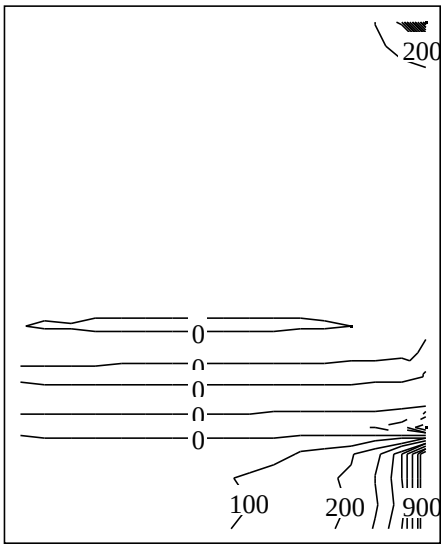

Figure 5.8: Ice thickness contours (100 m C.I.) after 150 years of evolution. The ice front is on the left-hand side of the diagram; the ice-stream input is on the lower right-hand side.

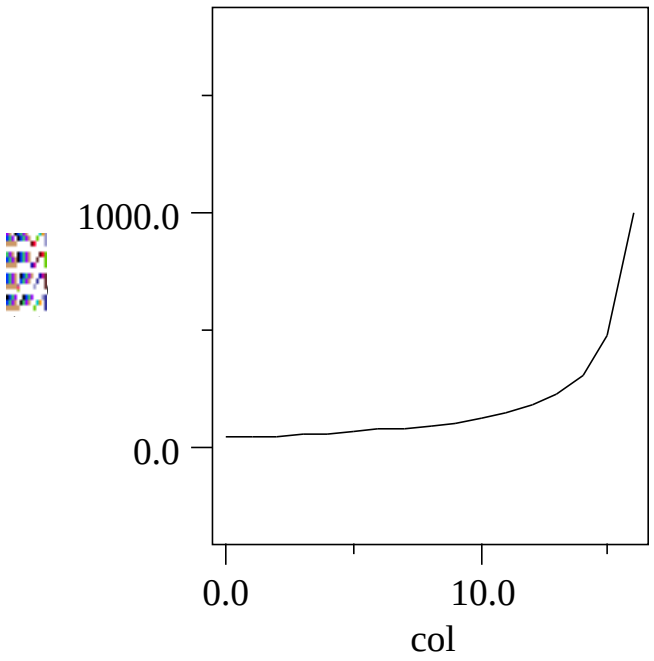

Figure 5.9: Ice thickness (m) along the axis of symmetry after 150 years of evolution.

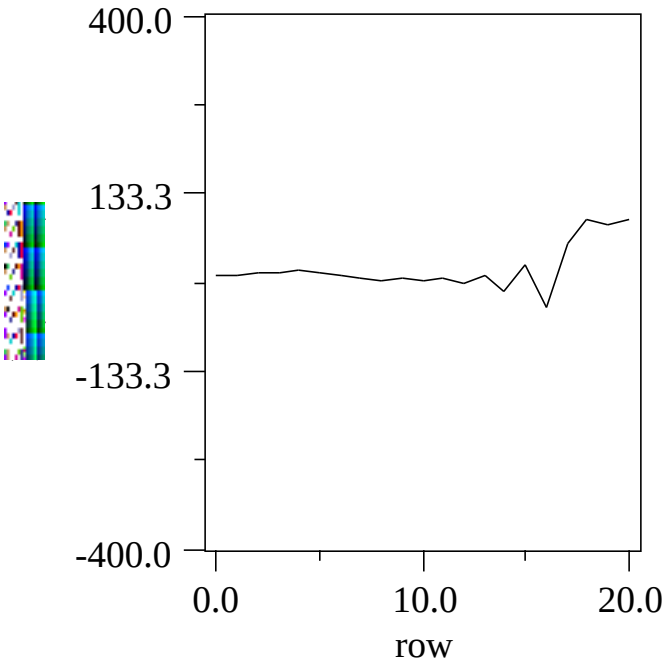

Figure 5.10: Ice thickness (m) along the transverse, midline axis of the ice shelf after 150 years of evolution. Notice that the field is negative at some nodes, and displays node-to-node wiggles that are unphysical. These aspects of the ice-thickness field lead us to reject the simple approach to the prognostic equation which does not explicitly enforce a positive-definite ice thickness.

The introduction of the constraint means that we can no longer deal strictly with a conservative solution of Eqn. (5.27). If an implicit time step is used, nodes surrounding a node where the ice thickness would normally become negative after the time step is taken "think" that the ice thickness is indeed negative while their own ice thickness is updated despite the fact that the constraint will be used at the end of the time step to ensure $h \geq 0$.

We implement the positive-definite constraint by adding the following fragment of MATLAB code to the above-listed script:

```
% Solve the system for new thickness
h=UpTri\(LowTri\R);
for n=1:nodes
if h(n) < 0
h(n)=0;
end
end
%
```

The results of an abbreviated test of the finite-element model of the prognostic equation with the positive-definite constraint are displayed in Figs. (5.11) - (5.13). The model was stepped through only 25 years of evolution (over 150 years are required to achieve a steady state in some portions of the ice-shelf domain) to reduce the computer time required to simulate the evolution (it takes approximately 15 seconds per time step on a MACINTOSH with a 40Mhz 68030 computer with FPU). The solution has improved (there are no negative ice thicknesses), but some of the undesirable node-to-node wiggles remain in the field.

The approach to modelling the prognostic equation presented up to this point is satisfactory for some ice-shelf modelling applications (once the technical hurdle of coupling the prognostic and diagnostic equations has been overcome). The time-stepping algorithm is stable and reflects a solution to the mass balance equation which is accurate over large scales. (The stability of the numerical algorithm in the present fixed-velocity exercise is unconditional, *i.e.,* not dependent on Δt. When nonlinear aspects of ice-shelf dynamics are introduced, such as when the velocity field is allowed to evolve freely

with the changing ice topography, stability may require Δt to be smaller than some empirically determined threshold.) In the next section, we will review a technique that adds *artificial damping* to small scale variations in the ice-thickness field to give a more cosmetically pleasing field that lacks the node-to-node wiggles that may be problematical in some applications.

5.6 Upwind Differencing & Artificial Diffusion

A common approach to filtering unwanted grid-point-to-grid-point noise (wiggles) is to employ an *upwind differencing* scheme in finite difference models. This scheme may be described in the context of the simple, one-dimensional advection equation for an arbitrary (unspecified) scalar tracer θ:

$$\frac{\partial \theta}{\partial t} + u\frac{\partial \theta}{\partial x} = 0 \tag{5.29}$$

(boundary conditions and initial conditions are irrelevant to our discussion, so are not specified). For a non-staggered finite-difference grid such as that shown in Fig. (5.14), the centered difference version of the $u\frac{\partial \theta}{\partial x}$ term is:

$$u\frac{\partial \theta}{\partial x}\Big|_{\text{centered}} \rightarrow u_i \frac{\theta_{i+1} - \theta_{i-1}}{2\Delta x} \tag{5.30}$$

where Δx is the grid spacing. The upwind difference version of the same term is (assuming $u > 0$):

$$u\frac{\partial \theta}{\partial x}\Big|_{\text{upwind}} \rightarrow u_i \frac{\theta_i - \theta_{i-1}}{\Delta x} \tag{5.31}$$

The difference between the two schemes is that the derivative of θ in the upwind scheme is no longer centered on the grid point i, but is instead, centered on the point located $\frac{1}{2}\Delta x$ upstream of i.

The advantage of the upwind scheme is that it damps wiggles. The reason the upwind scheme damps wiggles is that it is formally equivalent to what

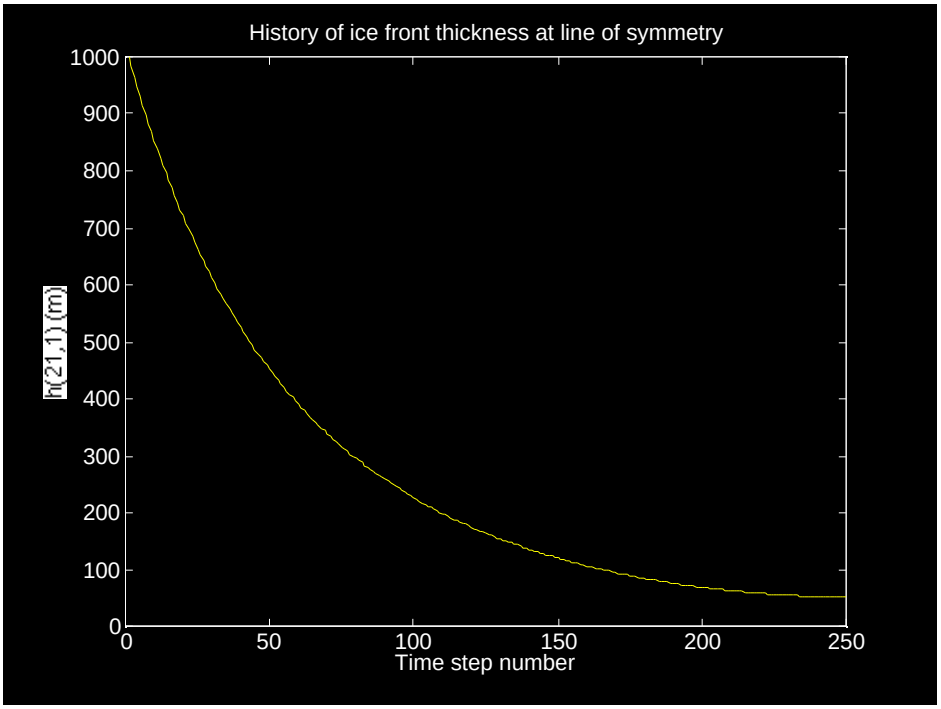

Figure 5.11: Ice thickness (dimensional units) at the ice front at the point where the ice front intersects the midline (axis of symmetry) of the ice shelf as a function of time. A positive-definite constraint on ice thickness was imposed on this particular solution. Equilibration is complete after about 250 time steps, corresponding to 25 years.

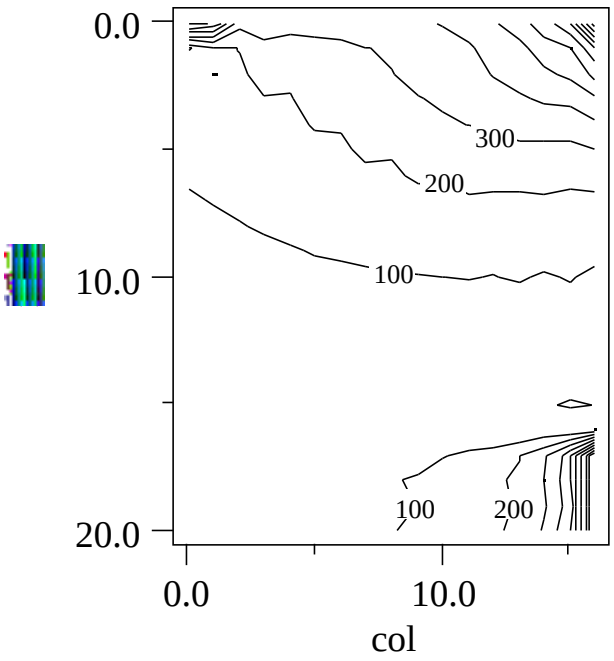

Figure 5.12: Ice thickness contours (100 m C.I.) after 25 years of evolution of a prognostic equation subject to the constraint that $h \geq 0$. (Comparison with the solution shown in Fig. 5.8 suggests that the upper right-hand-corner of the domain has not yet reached equilibrium. This is not important, however, because the purpose of this simulation is to demonstrate the maintenance of positive definite ice thickness.) The ice front is on the left-hand side of the diagram; the ice-stream input is on the lower right-hand side.

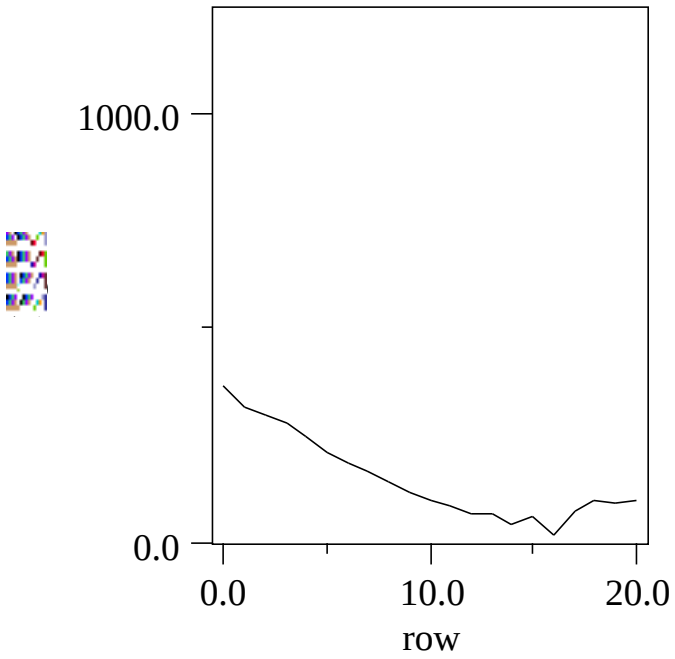

Figure 5.13: Ice thickness (m) along the transverse, midline axis of the ice shelf after 25 years of evolution with an algorithm that ensures positive ice thickness. Notice that the negative thickness which appeared in Fig. (5.10) at some nodes has been eliminated. The solution still displays some undesirable node-to-node wiggles.

you would get if you used the centered scheme with an additional artificial diffusion with a diffusivity $\kappa_{\text{art}} = \frac{u_i \Delta x}{2}$. In other words:

$$u\frac{\partial \theta}{\partial x}\Big|_{\text{centered}} - \frac{\partial}{\partial x}\left(\kappa_{\text{art}}\frac{\partial \theta}{\partial x}\right) = u\frac{\partial \theta}{\partial x}\Big|_{\text{upwind}} \tag{5.32}$$

This can be proven by substituting the finite-difference formula given in Eqns. (5.30) - (5.31) and the finite-difference formula for $\frac{\partial}{\partial x}\left(\kappa_{\text{art}}\frac{\partial \theta}{\partial x}\right)$ into the above expression and allowing terms to cancel.

5.6.1 A Finite-Element Implementation of Artificial Diffusion

We can achieve the same benefits as an upwind finite-difference scheme by introducing a diffusive term into the mass continuity (prognostic) equation:

$$\frac{\partial h}{\partial t} + \nabla \cdot (\mathbf{u}h) - 1 - \nabla \cdot (\underline{\kappa}\nabla h) = 0 \tag{5.33}$$

where $\underline{\kappa}$ is a 2×2 diffusivity tensor designed to diffuse along flowlines and minimize diffusion across flowlines

$$\underline{\kappa} = \left(\frac{\sqrt{2a^e}}{6}\right)\begin{bmatrix} |u_1 + u_2 + u_3| & 0 \\ 0 & |v_1 + v_2 + v_3| \end{bmatrix} = \begin{bmatrix} \kappa_{xx} & 0 \\ 0 & \kappa_{yy} \end{bmatrix} \tag{5.34}$$

where u_1, u_2, u_3, etc. are the nodal values of the horizontal velocity components. The artificial diffusivity tensor is designed to give about the same damping as that which would be associated with upwind differencing if the finite-element mesh consisted of a regular array of nodes similar to a finite-difference grid. (The absolute value of the local velocity is used in defining $\underline{\kappa}$ to ensure the diffusivity will always be positive.)

The Galerkin method requires the solution of the artificially damped prognostic equation to satisfy

$$\int_{\Omega}\int \left(\psi\frac{\partial h}{\partial t} - uh\frac{\partial \psi}{\partial x} - vh\frac{\partial \psi}{\partial y} - \psi + \frac{\partial \psi}{\partial x}\kappa_{xx}\frac{\partial h}{\partial x} + \frac{\partial \psi}{\partial y}\kappa_{yy}\frac{\partial h}{\partial y}\right) dxdy$$

$$+ \oint_{\delta\Omega} \cancel{h}\ \mathbf{u}\cdot\mathbf{n}ds - \oint_{\delta\Omega}\psi(\underline{\kappa}\nabla h)\cdot\mathbf{n}ds = 0 \tag{5.35}$$

The addition of artificial diffusion bumps the prognostic equation up to a higher order (two spatial derivatives instead of one are now involved), and this necessitates specifying additional boundary conditions. For our purposes, we might as well take $\nabla h \cdot \mathbf{n} = 0$ on all portions of $\delta\Omega$.

The matrix-stuffing conventions developed above for the prognostic equation receive the following additional contributions to account for the artificial diffusion:

$$a^e \left(\kappa_{xx}\alpha_k\alpha_l + \kappa_{yy}\beta_k\beta_l\right) \rightarrow A_{\gamma(k),\gamma(l)} \tag{5.36}$$

if an implicit treatment of artificial diffusion is desired, or

$$-a^e h_l \left(\kappa_{xx}\alpha_k\alpha_l + \kappa_{yy}\beta_k\beta_l\right) \rightarrow R_{\gamma(k)} \tag{5.37}$$

if an explicit treatment is desired. In practice, the implicit treatment of diffusion is better because the explicit treatment introduces an additional numerical stability concern (which is alleviated when Δt is chosen to be sufficiently small).

5.6.2 Ice Thickness Solution with Artificial Damping

The above implicit treatment of artificial diffusion was implemented by adding the two following fragments of MATLAB code to the script listed above:

```
% Compute artificial diffusivity terms
%
kxx=sqrt(2*area(n))/6*abs(u(index(n,1)*2-1) + u(index(n,2)*2-1)...
+u(index(n,3)*2-1) );
kyy= sqrt(2*area(n))/6*abs(u(index(n,1)*2) + u(index(n,2)*2)...
+u(index(n,3)*2) );
%
```

and

```
value(count)=value(count)+area(n)*(kxx*alpha(n,k)*alpha(n,l) ...
+kyy*beta(n,k)*beta(n,l));
```

The single line of MATLAB code listed above is inserted in the loop which constructs the matrix **A**.

The results of an abbreviated test of the finite-element model of the prognostic equation with both the positive-definite constraint and artificial diffusion are displayed in Figs. (5.15) and (5.16). The model was stepped through only 25 years of evolution (over 150 years are required to achieve a steady state in some portions of the ice-shelf domain) to reduce the computer time required to simulate the evolution. The solution suggests that many of the undesirable node-to-node wiggles have been eliminated.

5.7 Putting It All Together: Fully Coupled Prognostic/Diagnostic Ice-Shelf Model

Finally, we come to the point where we have tested the two main ingredients of a finite-element ice-shelf model, and we are ready to couple them in a realistic simulation of ice-shelf evolution. There is no new theory to be considered at this stage, we must simply become good programmers and fit the two portions of MATLAB code developed in the previous sections together. We must keep in mind that because ice velocity changes with each time step, the matrix **A** used in the prognostic portion of the model must be reconstructed at each time step. Considerably more LU-factorizations of **A** are thus required, and the coupled model will thus take longer to run. Another point to keep in mind is the fact that the entire code for solving the nonlinear diagnostic equations must be imbedded within the loop that does the time step. A flow chart showing the sequence of events leading to a successful time step is displayed in Fig. (5.17).

A comment about Δt

The implicit time-stepping scheme suggests that the solution of the prognostic (mass balance) equation should be stable for any time-step size. In practice, the time-step size should be chosen so that all imaginary passive-

tracer particles initially released at the nodal points (except those starting at ice-front nodes) remain confined by one single element throughout the time span $[t, t + \Delta t]$. In other words, it is best to choose a Δt small enough so that the distance traveled by one of these imaginary passive-tracer particles will not exceed the mesh spacing. In circumstances where the velocity field of the ice shelf changes substantially through the time, it may be wise to consider adaptively modifying Δt during the simulation. In the EISMINT Level 1 test, where an ice shelf that is initially 1000 m thick is allowed to evolve towards a substantially thinner (and slower flowing) steady state, the 0.1-year time-steps needed in the beginning may be unnecessarily short towards the end of the simulation. With this consideration in mind, we reset the time-step size in the EISMINT simulation according to the following rule

$$\Delta t = \frac{\frac{\Delta L}{\sqrt{2}}}{2 \max(\mathbf{u})} \tag{5.38}$$

where ΔL is the nondimensional size of the finite-element mesh spacing (shortest distance between node points in the regular mesh shown in Fig. 5.1). The idea of the above equation is that Δt is just large enough so that the fastest moving passive-tracer particle will move half the distance along a diagonal trajectory across a typical triangular element.

Results: Evolution to steady state

The coupled ice-shelf model was run through 1084 years of evolution using 658 time steps of variable Δt (as described in Eqn. 5.38). The initial condition corresponded to the EISMINT Level 1 exercise, and represented a uniform ice thickness of 1000 m. The input ice thickness and velocity at the ice-stream boundary were held fixed though time at 1000 m and 400 m/year, respectively. (The 658 time steps took approximately 60 hours of CPU time on a Macintosh IIfx with 40Mhz 68030 CPU and 68882 FPU.) Contour maps of the ice thickness and velocity magnitude after 1084 years of evolution are displayed in Figs. (5.18) and (5.19). Cross sections of ice thickness along the longitudinal axis of symmetry and transverse midline after 95 years of evolution are shown in Figs. (5.20) and (5.21). Time series of total (dimensional) volume and average velocity magnitude at each time

step are shown in Figs. (5.22) and (5.23). A comparison of the ice-thickness cross section along the logitudinal axis of symmetry and the exact, analytic ice-tongue profile (steady state) developed in Chapter (3) is shown in Fig. (5.24).

One curious feature that was not expected is the low ice-thickness near the ice front on the longitudinal axis (Fig. 5.24). The analytic ice tongue solution is thicker at the ice front than the finite-element solution for the ice shelf. This is unexpected because the ice tongue, under usual circumstances, should exhibit the most rapid spreading rates, and thus define a "lower bound" to the ice thickness in a longitudinal section of the ice shelf. The explanation of why the ice-shelf thickness is *less* than the ice-tongue thickness in this circumstance lies in the fact that the ice shelf displays a *two-dimensional* ice-thickness pattern. On close inspection, the ice shelf *thins* away from the centerline axis of symmetry (which is the center of the ice lobe debouched from the ice-stream inlet boundary). The ice shelf thus exhibits *two-dimensional* spreading (*i.e.,* transverse spreading as well as longitudinal spreading); and is thus able to achieve a lower thickness at the ice front than the ice tongue which is only allowed to spread longitudinally.

5.7.1 MATLAB Script for the Coupled Ice-Shelf Model

Below is a listing of the MATLAB script used to create the above ice-shelf simulation. Notice that the script has been broken down into sub-scripts (MATLAB 's version of subroutines) to make the code easier to read and understand.

Driver code

```
% This script represents a finite-element model of an ice shelf.
%
init
nsteps=1500 % Note:  only 658 time steps were actually completed
in %         a single weekend.
```

```
history=zeros(nsteps,5);
year=0;
for time=1:nsteps
dt = (5.0e3/L)/(2^ (3/2) * max(abs(u)));
year=year+dt*Z/a/31556926
time
timeseries
diagnost
prog
end
output
```

Init code

```
% This script fragment does the initialization
%
nrowsA=5920;
nrowsD=23040;
rowA=zeros(nrowsA,1);
colA=zeros(nrowsA,1);
rowD=zeros(nrowsD,1);
colD=zeros(nrowsD,1);
c_old=zeros(nel,1);
h=10^ 3/Z*ones(nodes,1);
count=0;
countD=0;
%
for n=1:nel
%
% Perform loading of row and col arrays
%
for i=1:3
for j=1:3
countD=countD+1;
```

```
rowD(countD)=index(n,j)*2-1;
colD(countD)=index(n,i)*2-1;
countD=countD+1;
rowD(countD)=index(n,j)*2-1;
colD(countD)=index(n,i)*2;
countD=countD+1;
rowD(countD)=index(n,j)*2;
colD(countD)=index(n,i)*2-1;
countD=countD+1;
rowD(countD)=index(n,j)*2;
colD(countD)=index(n,i)*2;
end
end
for l=1:3
for k=1:3
count=count+1;
rowA(count)=index(n,k);
colA(count)=index(n,l);
end
end
%
end % end loop over elements
%
% Boundary conditions
% Ice front contour:  Implicit case
%
for n=1:imax-1
for k=1:2
for l=1:2
for j=1:2
count=count+1;
rowA(count)=icefnt(n,k);
colA(count)=icefnt(n,l);
end
end
end
end
```

```
%
%
```

Timeseries code

```
% This script fills the history (bookeeping)
% time-series array
history(time,1)=year;
history(time,2)=h(gamma(21,1));
volume=0;
speed=0;
for n=1:nel
volume=volume+area(n)*(h(index(n,1))+h(index(n,2)) ...
+h(index(n,3)))/3;
speed=speed+area(n)*( ...// sqrt( u(index(n,1)*2-1)^ 2+u(index(n,1)*2)^
2 ) ...
+ sqrt( u(index(n,2)*2-1)^ 2+u(index(n,2)*2)^ 2 ) ...
+ sqrt( u(index(n,3)*2-1)^ 2+u(index(n,3)*2)^ 2 ) )/3;
end
history(time,3)=volume;
history(time,4)=speed*U/sum(area)*31556926;
history(time,5)=sqrt(u(gamma(21,1)*2-1)^ 2 ...
+u(gamma(21,1)*2)^ 2)*U*31556926;
```

Diagnostic code

```
% This script represents the Diagnostic solver
% sequence of Du=F prolbems
%
loop=0;
c_old=zeros(nel,1);
while max((abs(c-c_old)./c)) > 0.01
```

```
loop=loop+1;
%
value=zeros(nrowsD,1);
F=zeros(2*nodes,1);
countD=0;
%
for n=1:nel
havg=(h(index(n,1))+h(index(n,2))+h(index(n,3)))/3;
%
% Effective diffusivity:
%
if loop > 1
c_old(n)=c(n);
end
ux=0;
uy=0;
vx=0;
vy=0;
for i=1:3
ux=ux+u(index(n,i)*2-1)*alpha(n,i);
uy=uy+u(index(n,i)*2-1)*beta(n,i);
vx=vx+u(index(n,i)*2)*alpha(n,i);
vy=vy+u(index(n,i)*2)*beta(n,i);
end
if (ux^ 2+vy^ 2+((uy+vx)^ 2)/4 +ux*vy ) > 10^ (-15)
c(n)=havg*(ux^ 2+vy^ 2+((uy+vx)^ 2)/4 +ux*vy )^ (-1/3);
else
c(n)=havg*10^ 5;
end
% Load right-hand-side vector:
%
for k=1:3
F(2*index(n,k)-1)=F(2*index(n,k)-1)+area(n)*havg^ 2*alpha(n,k);
F(2*index(n,k))=F(2*index(n,k))+area(n)*havg^ 2*beta(n,k);
end
%
% Load matrix:
```

```
%
for i=1:3
for j=1:3
countD=countD+1;
valueD(countD)=area(n)*c(n)*(alpha(n,i)*alpha(n,j)+beta(n,i)*beta(n,j)/4);
countD=countD+1;
valueD(countD)=area(n)*c(n)*(beta(n,i)*alpha(n,j)/2+alpha(n,i)*beta(n,j)/4);
countD=countD+1;
valueD(countD)=area(n)*c(n)*(alpha(n,i)*beta(n,j)/2+beta(n,i)*alpha(n,j)/4);
countD=countD+1;
valueD(countD)=area(n)*c(n)*(beta(n,i)*beta(n,j)+alpha(n,i)*alpha(n,j)/4);
end
end
% End loop over elements
end
D=sparse(rowD,colD,valueD);
% Boundary conditions
bxcount=0;
bycount=0;
for j=1:(jmax-1) % left side
bxcount=bxcount+1;
D(Boundu(bxcount)*2-1,Boundu(bxcount)*2-1)=10^ 12;
F(Boundu(bxcount)*2-1)=0;
bycount=bycount+1;
D(Boundv(bycount)*2,Boundv(bycount)*2)=10^ 12;
F(Boundv(bycount)*2)=0;
end
for i=1:imax % top side
bxcount=bxcount+1;
bycount=bycount+1;
D(Boundu(bxcount)*2-1,Boundu(bxcount)*2-1)=10^ 12;
F(Boundu(bxcount)*2-1)=0;
D(Boundv(bycount)*2,Boundv(bycount)*2)=10^ 12;
F(Boundv(bycount)*2)=0;
end
for j=1:(jmax-1) % right side
bxcount=bxcount+1;
```

```
D(Boundu(bxcount)*2-1,Boundu(bxcount)*2-1)=10^ 12;
F(Boundu(bxcount)*2-1)=0;
end
F(gamma(18,17)*2)=-10^ 12;
F(gamma(19,17)*2)=-10^ 12;
F(gamma(20,17)*2)=-10^ 12;
F(gamma(21,17)*2)=-10^ 12;
% Solve the system for new ice velocity
u=D\F;
end % end While statement
```

Prog code

```
% This script does the Prognostic part
% by solving Ah=R
%
%
valueA=zeros(nrowsA,1);
R=zeros(nodes,1);
count=0;
% loop over elements
for n=1:nel
% Compute artificial diffusivity terms
%
kxx=sqrt(2*area(n))/6*abs(u(index(n,1)*2-1) + u(index(n,2)*2-1)...
+u(index(n,3)*2-1) );
kyy= sqrt(2*area(n))/6*abs(u(index(n,1)*2) + u(index(n,2)*2)...
+u(index(n,3)*2) );
%
%
% Load right-hand-side vector:
%
R(index(n,1))=R(index(n,1))+area(n)/3;
R(index(n,2))=R(index(n,2))+area(n)/3;
```

```
R(index(n,3))=R(index(n,3))+area(n)/3;
for k=1:3
for j=1:3
if j == k
R(index(n,k))=R(index(n,k))+area(n)*h(index(n,j))/(6*dt);
else
R(index(n,k))=R(index(n,k))+area(n)*h(index(n,j))/(12*dt);
end
end
end
% Construct A using updated velocity:
for l=1:3
for k=1:3
count=count+1;
if k == l
valueA(count)=area(n)/(6*dt);
else
valueA(count)=area(n)/(12*dt);
end
for j=1:3
if j == l
valueA(count)=valueA(count)-(alpha(n,k)*u(index(n,j)*2-1)...
+beta(n,k)*u(index(n,j)*2))...
area(n)/6;
else
valueA(count)=valueA(count)-(alpha(n,k)*u(index(n,j)*2-1)...
+beta(n,k)*u(index(n,j)*2))...
area(n)/12;
end
end
valueA(count)=valueA(count)+area(n)*(kxx*alpha(n,k)*alpha(n,l)...
+kyy*beta(n,k)*beta(n,l));
end
end
%
end % End loop over elements
% Boundary conditions
```

```
% Ice front contour:  Implicit case
%
for n=1:imax-1
for k=1:2
for l=1:2
for j=1:2
count=count+1;
if l==k & j==k
valueA(count)=-u(icefnt(n,j)*2)*dl/4;
else
valueA(count)=-u(icefnt(n,j)*2)*dl/12;
end
end
end
end
end
%
A=sparse(rowA,colA,valueA);
% Fix the input ice thickness to 1
%
A(gamma(18,17),gamma(18,17))=10^ 15;
A(gamma(19,17),gamma(19,17))=10^ 15;
A(gamma(20,17),gamma(20,17))=10^ 15;
A(gamma(21,17),gamma(21,17))=10^ 15;
%
% Ice stream input condition:
R(gamma(18,17))=10^ 3/Z*10^ 15;
R(gamma(19,17))=10^ 3/Z*10^ 15;
R(gamma(20,17))=10^ 3/Z*10^ 15;
R(gamma(21,17))=10^ 3/Z*10^ 15;
% Solve the system for new thickness
h=A\R;
for n=1:nodes
if h(n) < 0
h(n)=0;
end
end
```

Output code

```
% This script writes the output
% (if any)
for i=1:imax
for j=1:jmax
ugrid(i,j)=u(gamma(i,j)*2-1)*U*31556926;
vgrid(i,j)=u(gamma(i,j)*2)*U*31556926;
hgrid(i,j)=h(gamma(i,j))*Z;
end
end
```

Note: The above code has been modified since the EISMINT model-intercomparison meeting in Bremerhaven, Germany (June 1994). For an updated copy of the code, please contact the author.

5.8 Summary

In this chapter, we have constructed a two-dimensional (plan view) model of an ice shelf. The difficulty we encountered stemmed from the fact that a time step in an ice shelf model requires solution of two problems: the diagnostic for the ice velocity, and the prognostic for the new ice thickness. Although the work is tedious, and the code can become lengthy, the process of constructing an ice-shelf model is essentially simple.

5.8.1 Exercise 1

Using data provided in `Rossmesh.mat` construct a snap-shot of the present flow of the Ross Ice Shelf using a finite-element model of the diagnostic stress equilibrium equations (suitably simplified). Experiment with values of

$\bar{B}$, the flow-law rate constant, to tune your model to the observed ice flow field (maximum ice front velocity about 1010 m per year).

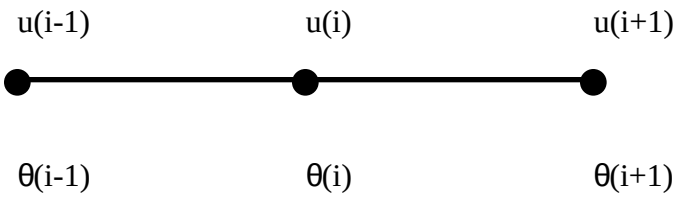

Figure 5.14: Nonstaggered finite-difference graph where θ and u are defined at the same locations on the grid. This grid scheme is used to illustrate upwind differencing.

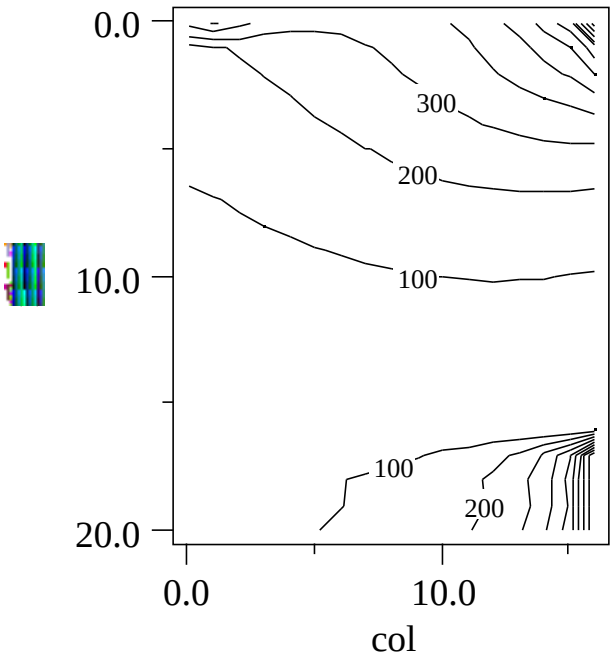

Figure 5.15: Ice thickness contours (100 m C.I.) after 25 years of evolution of a prognostic equation subject to the constraint that $h \geq 0$ and to an artificial diffusion to smooth wiggles. (Comparison with the solution shown in Fig. 5.8 suggests that the upper right-hand-corner of the domain has not yet reached equilibrium. This is not important, however, because the purpose of this simulation is to demonstrate the maintenance of positive definite ice thickness.) The ice front is on the left-hand side of the diagram; the ice-stream input is on the lower right-hand side.

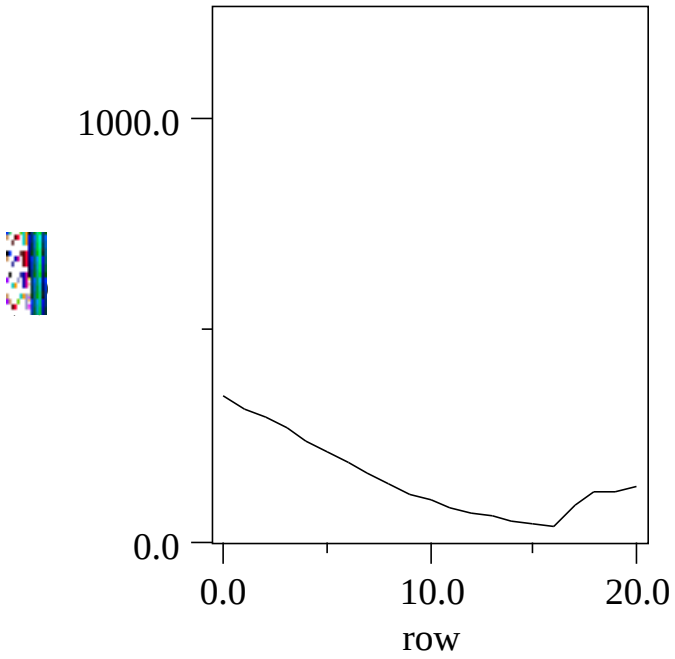

Figure 5.16: Ice thickness (m) along the transverse, midline axis of the ice shelf after 25 years of evolution with an algorithm that preserves positive definite ice thickness and dissipates wiggles with artificial diffusion. Notice that the amplitude of undesirable node-to-node wiggles have been reduced over those displayed in Fig. (5.13).

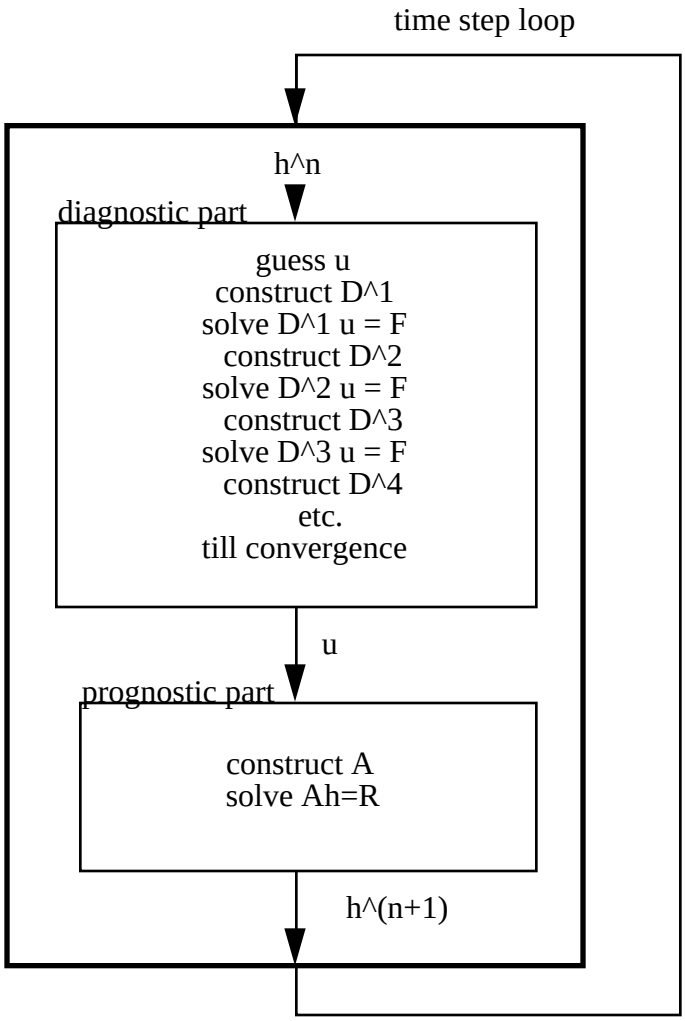

Figure 5.17: Flow chart indicating how diagnostic and prognostic parts of the ice shelf model are coupled together to achieve an algorithm that steps through time.

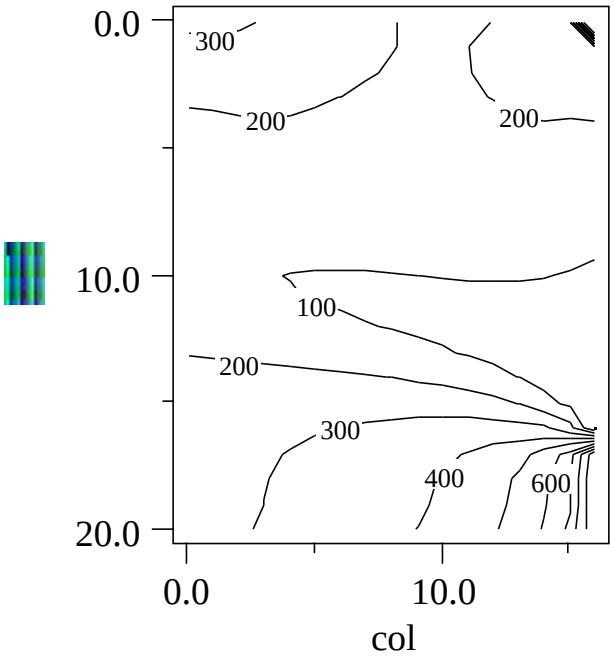

Figure 5.18: Contour map of ice thickness (C.I. = 100 m) after 1084 years of evolution towards a steady state from an initial condition of a uniform 1,000 m ice thickness. The ice-front boundary is the left side of the diagram. The lower right side of the diagram represents the location where ice-stream input was specified as a boundary condition.

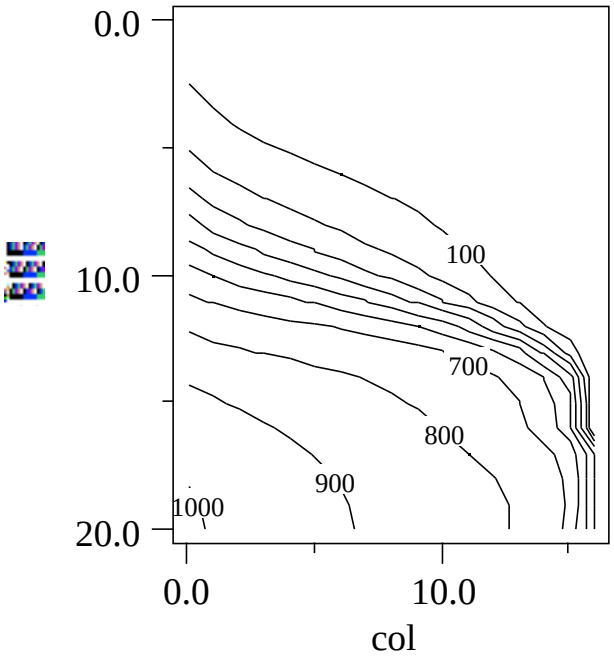

Figure 5.19: Contour map of ice velocity magnitude (C.I. = 100 m/year) after 1084 years of evolution towards a steady state from an initial condition of a uniform 1,000 m ice thickness. The ice-front boundary is the left side of the diagram. The lower right side of the diagram represents the location where ice-stream input was specified as a boundary condition.

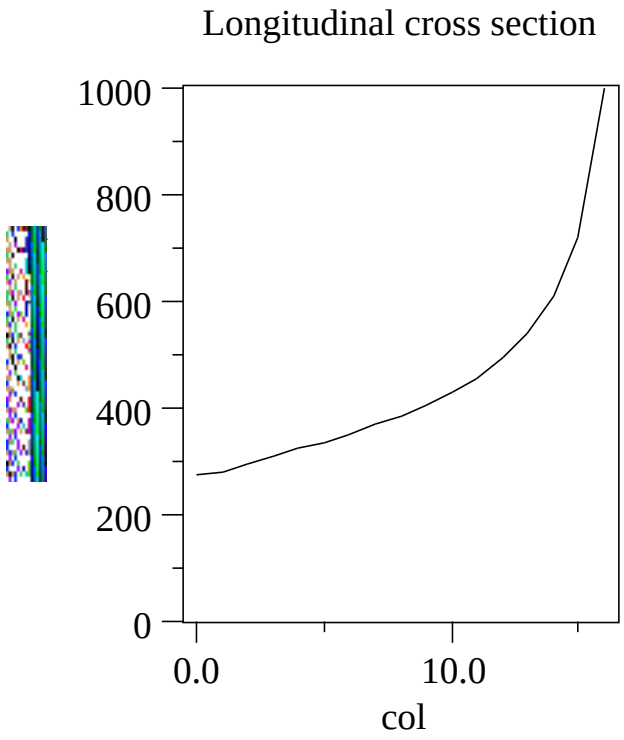

Figure 5.20: Longitudinal cross section of ice thickness along the axis of symmetry after 1084 years of evolution towards steady state.

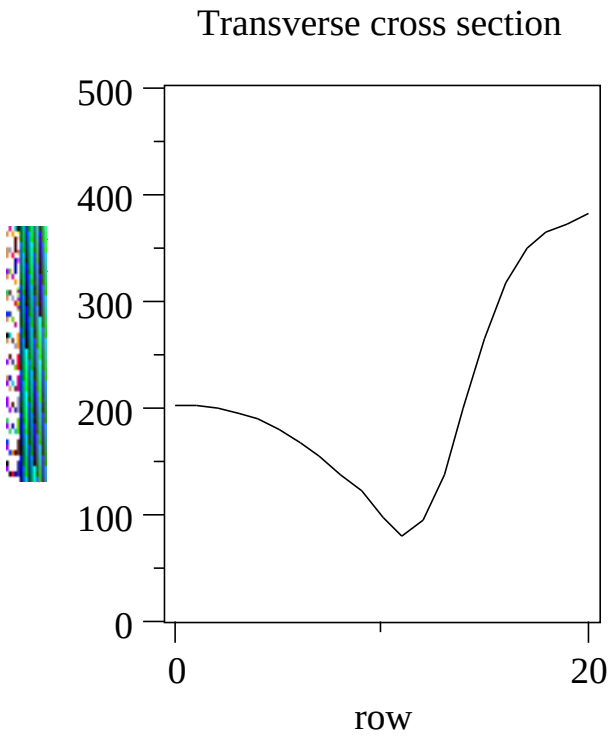

Figure 5.21: Transverse cross section of ice thickness along the midline after 1084 years of evolution towards steady state.

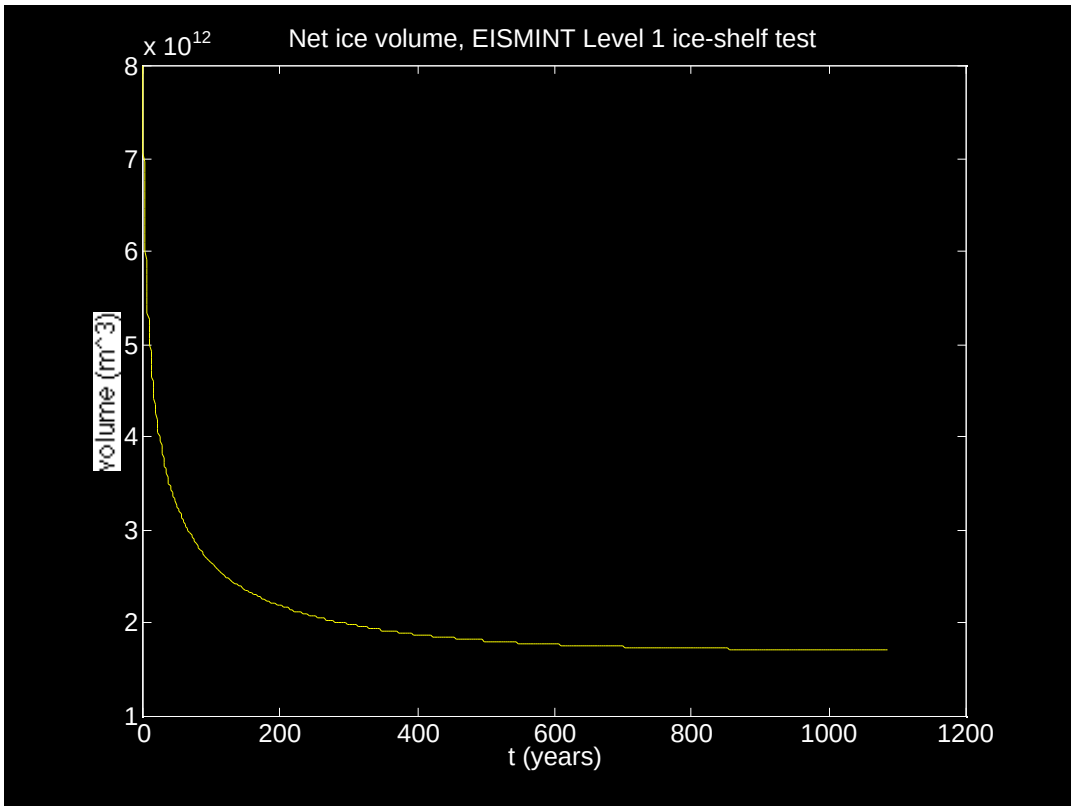

Figure 5.22: Ice volume (m^3) as a function of time. Steady state was not achieved after 1084 years, but the ice-shelf was close to steady state.

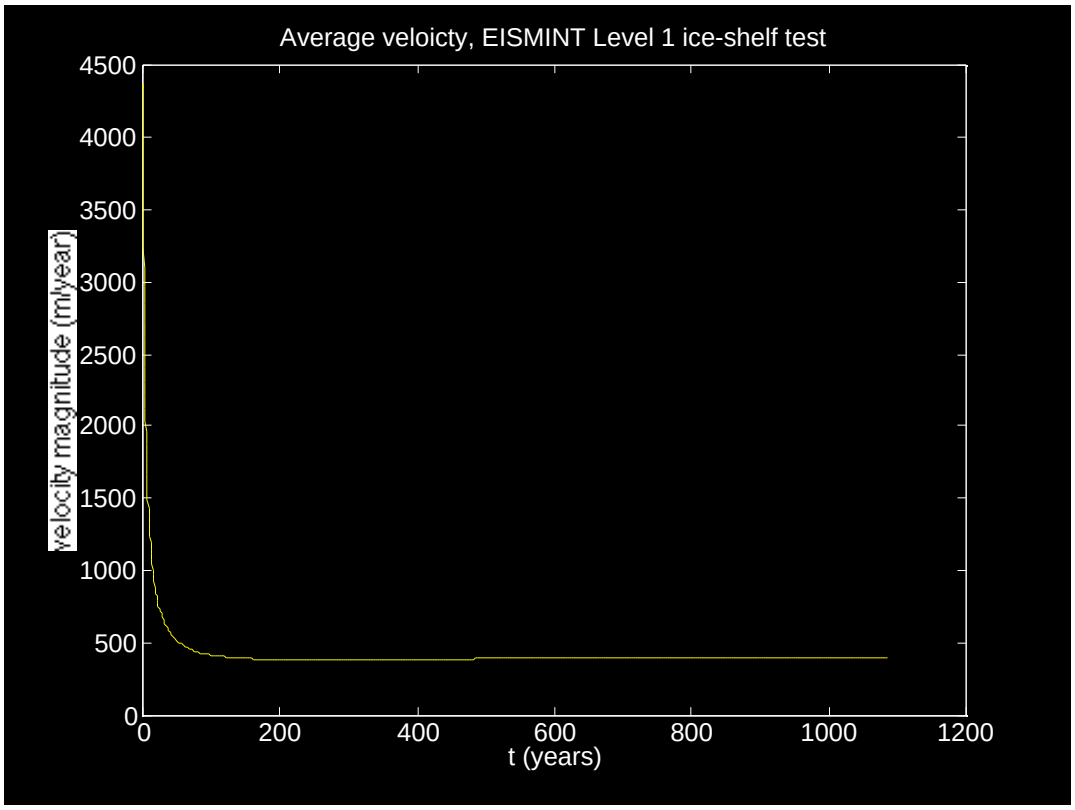

Figure 5.23: Area-averaged velocity magnitude (m/year) as a function of time.

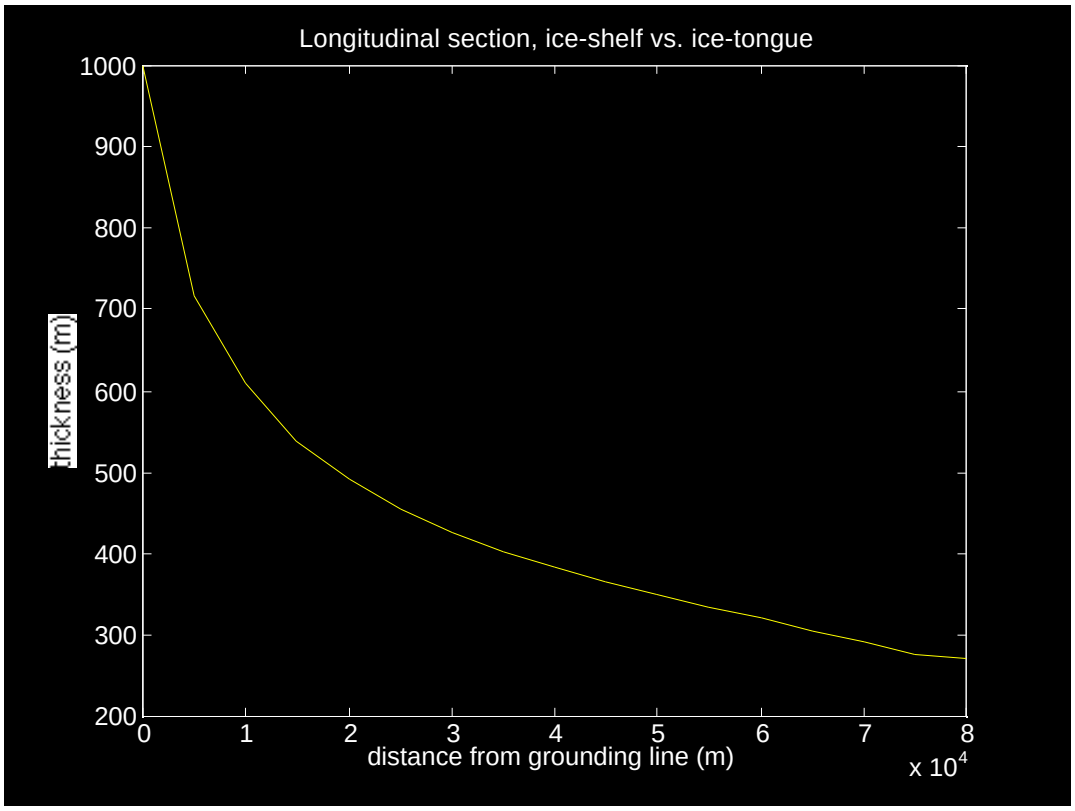

Figure 5.24: Comparison of the longitudinal section thickness along the axis of symmetry for the ice shelf after 1084 years of evolution towards steady state (line) and for the unconfined ice tongue (exact, analytic solution, asterisks). The ice-shelf thickness is greater than that of the ice tongue near the grounding line, and this reflects the fact that confinement by the embayment can substantially reduce longitudinal spreading rates near the grounding line. The ice-shelf thickness is less than that of the ice tongue near the ice front, and this reflects the fact that the ice shelf exhibits two-dimensional (longitudinal and transverse) spreading near the ice front, whereas the ice tongue is restricted to longitudinal spreading only.

Chapter 6

EISMINT Workshop Epilogue: Mass Conservation Problems

Several unforeseen inadequacies of the solution to the ice-shelf modelling exercise described in the previous chapter came to light at the EISMINT model-intercomparison workshop held in June, 1994, in Bremerhaven, Germany. In particular, the flux of ice through the ice front when in steady state exceeded the input of ice to the ice shelf (via snow accumulation and ice-stream discharge) by about 27% . This mass imbalance prompted several additional tests described here. The results of these additional tests suggest that the ice-stream influx conditions specified in the EISMINT tests are a source of numerical error (discretization error) which ultimately manifests itself in one of two ways: gross mass imbalance, or gross ice-thickness error. When the model developed in the previous chapter is run without an ice-stream input condition, mass-balance is accurate to better than 0.1 %.

6.1 Computation of Mass Balance

To compute the net mass balance associated with the ice-shelf model, three integrations of model data must be performed. The first is an area integral

to compute the net ice input due to snow accumulation. (Basal melting is assumed zero.) The second is a line integral along the ice-stream inlet to compute the net flux of ice delivered to the ice shelf by the ice stream. The third is a line integral along the ice front to compute the net flux of ice into the ocean. Care must be exercised on the two line integrals to ensure that the integral of $h\mathbf{u} \cdot \mathbf{n}$, where $\mathbf{n}$ is the outward pointing unit vector to the boundary, is performed in a manner consistent with the finite-element discretization. The velocity directed normal to the boundary $\mathbf{u} \cdot \mathbf{n}$ and the ice thickness h are linear functions between the two nodes which define a boundary segment. Their product is thus a quadratically varying function. (If, for example, one were to compute the flux through a boundary segment by multiplying the length of the segment by the average ice thickness and average outward directed ice velocity, one would be computing an erroneous mass flux that is inconsistent with the numerical method used to generate the solution.)

The second line integral mentioned above, *i.e.,* the integral along the ice-stream inlet boundary, is optional. Recall that Eqn. (6.3) suggests that there are two types of boundary conditions: one where flux is specified and the other where ice thickness is specified. In the circumstance where ice thickness is specified, the flux needed to maintain that fixed ice thickness becomes a *flux of constraint*, *i.e.,* is undetermined until after the model produces its solution. Thus, for this circumstance, the line integral must be evaluated *after the fact*, *i.e.,* after the model produces the solution, to find out what the flux of constraint actually was. When h is specified in the finite-element model developed in the previous chapter, rows of the matrix $\mathbf{A}$ in Eqn. (5.24) are modified or eliminated. (One way to do this is to replace the diagonal element with a one and all off diagonal elements with zeroes for rows referencing a nodal value of h to be specified as a boundary condition. This replacement effectively *erases* all information about the velocity at the boundary; thus, the model regards the flux to be that which enforces the specified h, not that which represents $h\mathbf{u}\cdot\mathbf{n}$.) This point is often misunderstood by modellers who have not worked with finite-element or Galerkin methods before.

In the second circumstance, where ice flux is specified at the open boundary, the line integral along the ice-stream inlet is no longer necessary. (Actually, it is necessary, but it's value is simply the specified flux times the

length of the boundary in question.) There is no need to reference model data to determine this flux because the model produces its solution with the firm constraint that the mass flux is indeed what it was specified to be. Inaccuracy of the model solution h at the open boundary may result from the numerical discretization error, as will be shown below to be the case for the EISMINT test. This inaccuracy, however, does not affect the mass balance. (This means that the inaccuracy in h and $h\mathbf{u} \cdot \mathbf{n}$ for the given $\mathbf{u}$ offset each other in a manner consistent with a correct rendition of the mass flux.)

Here is the MATLAB code used to examine the mass balance characteristics of the solution found in the previous chapter. Note the optional statements which account for the different circumstances of boundary condition specification at the ice-stream inlet.

```
% This program computes the net mass budget (input vs output)
% for the end-state of the ice-shelf model run:

% Definitions:
% Input = ice flowing through the icestream boundary
% + snow accumulation over entire area
%
% Output = ice flowing out of the ice front
%

% Units are in m^ 3 per year (velocities must be specified in m/a)

% Input:

totalarea= 80.e3 * 100.e3;
accum=0.3;
Lside=5.0e3; % length of grid spacing

Inputsnow= totalarea*accum

Inputstream=0;
nodecount=0;
for k=1:20
nodecount=nodecount+1;
h1=hgrid(nodecount,17);
h2=hgrid(nodecount+1,17);
v1=vgrid(nodecount,17);
v2=vgrid(nodecount+1,17);

Inputstream= Inputstream - h1*v1*Lside - (v1*(h2-h1) + h1*(v2-v1))*Lside/2
...
- (h2-h1)*(v2-v1)*Lside/3;

% note , if flux is specified, the previous statement must be replaced
% with Inputstream=Inputstream*1000*400*Lside
% see statement below:
```

```
% note minus sign because v<0 means "in" to the ice shelf.
end

% Use the following statement if flux is specified (here we have
an example
% where the width of the inlet is 3xLside
% Inputstream=400*1000*Lside*3;
Inputstream

Input=Inputsnow+Inputstream

% Output through ice front (other sides are no flux):

Output=0;
nodecount=0;
for k=1:20
nodecount=nodecount+1;
h1=hnpdgrid(nodecount,1);
h2=hnpdgrid(nodecount+1,1);
v1=vgrid(nodecount,1);
v2=vgrid(nodecount+1,1);

Output= Output + h1*v1*Lside + (v1*(h2-h1) + h1*(v2-v1))*Lside/2
...
+ (h2-h1)*(v2-v1)*Lside/3;
end

Output

net= Input+Output
```

The result of applying the above mass-balance diagnostic to the EISMINT intercomparison test results described in the previous chapter (and shown in Figs. (**??**) and (5.19)) is listed as follows:

Input	9.0667×10^9 m^3/a
Output	-11.502×10^9
net	-2.44×10^9
percent	26.9 %

As is clearly shown by the above table, the steady-state solution transmits about 27 % more ice through the ice front than flows into the ice shelf.

What could be the cause of this mass imbalance? How could the ice-shelf model have achieved a steady state with so much extra mass outflow?

These anxiety-inducing questions were not easy to answer. It took me several weeks of effort to finally develop an understanding of what was going wrong. The process by which I developed an understanding of the answers to these questions involved re-running the ice-shelf model with different boundary conditions to determine the circumstances where it would or would not conserve mass.

6.2 Test With Zero Ice-Stream Input

The first model diagnosis test I considered was to examine the mass balance of a steady-state solution produced for a model domain which lacked the ice-stream inlet, *i.e.,* for the case where $\mathbf{u} = \mathbf{0}$ for the entire inland boundary of the ice shelf. The results of this test are shown in Figs. (6.1) - (6.2). The mass balance summary is listed as follows:

Input	2.4000×10^9 m^3/a
Output	-2.3975×10^9
net	0.0025×10^9
percent	0.1 %

When the ice stream inlet is eliminated, the finite-element ice-shelf model appears to conserve mass. This suggests that the cause of the mass imbalance in the EISMINT test of the previous chapter is associated with some aspect of the ice-stream inlet.

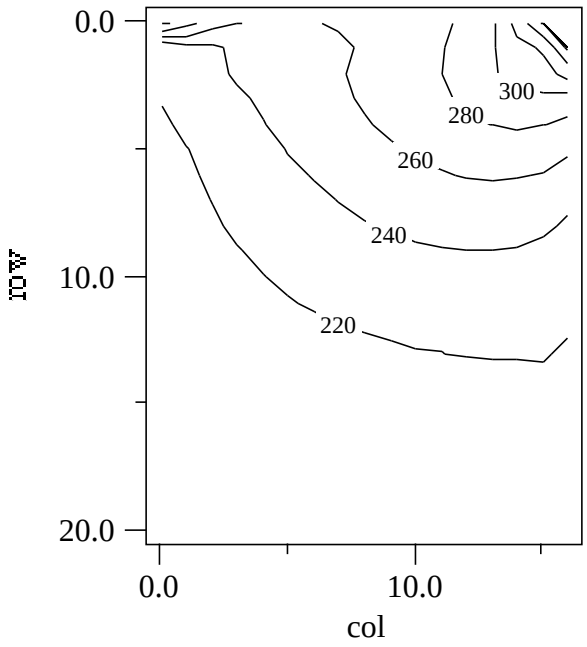

Figure 6.1: Steady-state ice thickness (dimensional units) for a for a model run which lacked ice-stream input. Contour interval is 20 m.

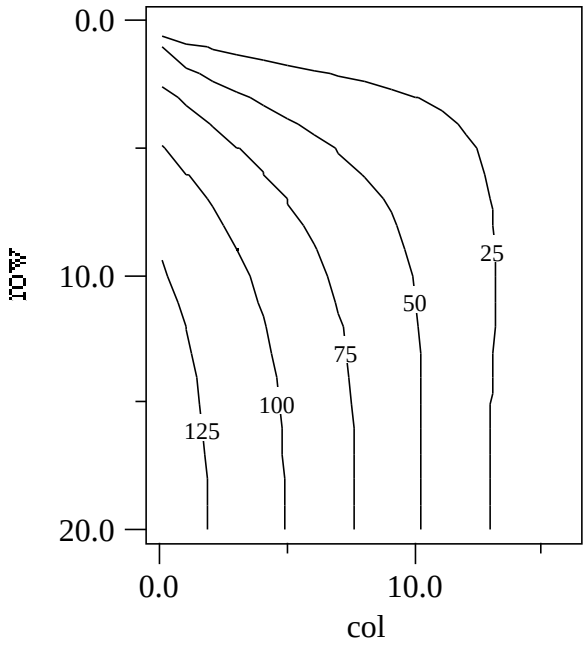

Figure 6.2: Steady-state ice velocity magnitude (dimensional units) for a for a model run which lacked ice-stream input. Contour interval is 25 m/a.

6.3 Test Comparison With an Exact, Analytic Solution

The next model-diagnostic test I performed was to repeat the one-dimensional ice-tongue experiment with the two-dimensional model. In other words, I specified ice-stream input along the entire "back" boundary of the EISMINT model domain, and required the two lateral boundaries of the model to be stress-free. The purpose of this experiment was to test how well the two-dimensional ice-shelf model could reproduce the exact analytic solution for the steady-state ice tongue developed by Van der Veen (and discussed in the previous chapters).

This experiment was further broken down into two sub experiments. In the first sub-experiment, the ice thickness h was specified as the ice-stream boundary condition. In the second sub-experiment, the mass flux $q = hv$ was specified as the ice-stream boundary condition. These two sub-experiments reveal the differences in model performance between the two alternative ways of specifying the boundary condition at the ice-stream inlet reflected in Eqn. ().

The ice-thickness profile down the centerline of the ice tongue (assumed to be 17 nodes long and 21 nodes wide) is displayed in Fig. (6.3). (Note: the variation of model output fields in the direction perpendicular to the flow of the simple one-dimensional ice tongue should be zero, *i.e.,* all model variables should be uniform with respect to x, the transverse coordinate. This was found, indeed, to be the case.)

For the sub-experiment in which the ice thickness was specified at the grounding line, the ice thickness produced by the model exceeds the analytic solution at all points downstream of the ice-stream input boundary (the boundary condition, of course, is satisfied exactly). As a result of this inaccuracy, the mass balance of the ice tongue is grossly in error:

Input	4.24×10^{10} m^3/a
Output	-6.47×10^{10}
net	2.23×10^{10}
percent	53 %

A 53% error associated with the attempt of a two-dimensional ice-shelf model to mimic the one-dimensional ice-tongue solution is strongly suggestive of the cause of the imbalance seen previously in the EISMINT ice-shelf test. The change in ice thickness between the node at the boundary and the next node downstream within the ice shelf is just too great for the numerical method to represent accurately. As shown in Fig. (6.3), approximately half of the total change in ice thickness of the ice tongue (for both the analytic and the numerical solutions) is accomplished between the first two nodes. The lack of resolution of this large thickness change introduces the numerical discretization error that yields the large mass imbalance.

The second sub-experiment was like the first except that the mass flux $q = 400 \times 1000$ per meter length of boundary was specified at the upstream boundary instead of the ice thickness. The result is shown in Fig. (6.4). Clearly, downstream of the upstream boundary, the finite-element model reproduces the exact analytic solution rather well. The first node, where in the previous run the ice thickness was specified, is the only place where model inaccuracy is apparent. For this run, the mass balance analysis is

Input	4.24×10^{10} m^3/a
Output	-4.24×10^{10}
net	5.55×10^{2}
percent	0 %

Now we are getting somewhere! Apparently, by insisting in the previous run that $h = 1000$ m at the upstream boundary, I introduced a flux of "constraint" that accounted for the 53 % mass imbalance. In the present run, the flux at the upstream boundary is specified to be its intended value, but the ice thickness at the upstream node is inaccurate. Given the agreement between the model and the exact analytic solution downstream of the first, upstream node, it is better, apparently, to specify flux and not ice thickness.

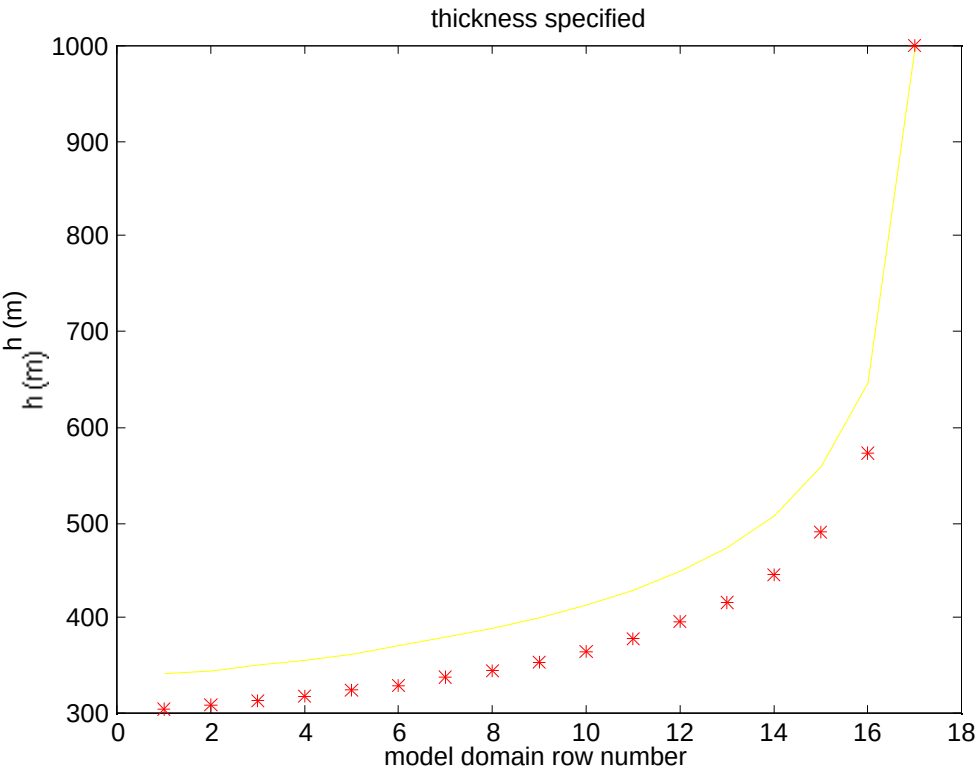

Figure 6.3: Steady-state ice thickness (dimensional units) for a for a model run in which the entire back boundary has ice-stream input (thickness specified). Asterisks denote the exact solution obtained by Van der Veen.

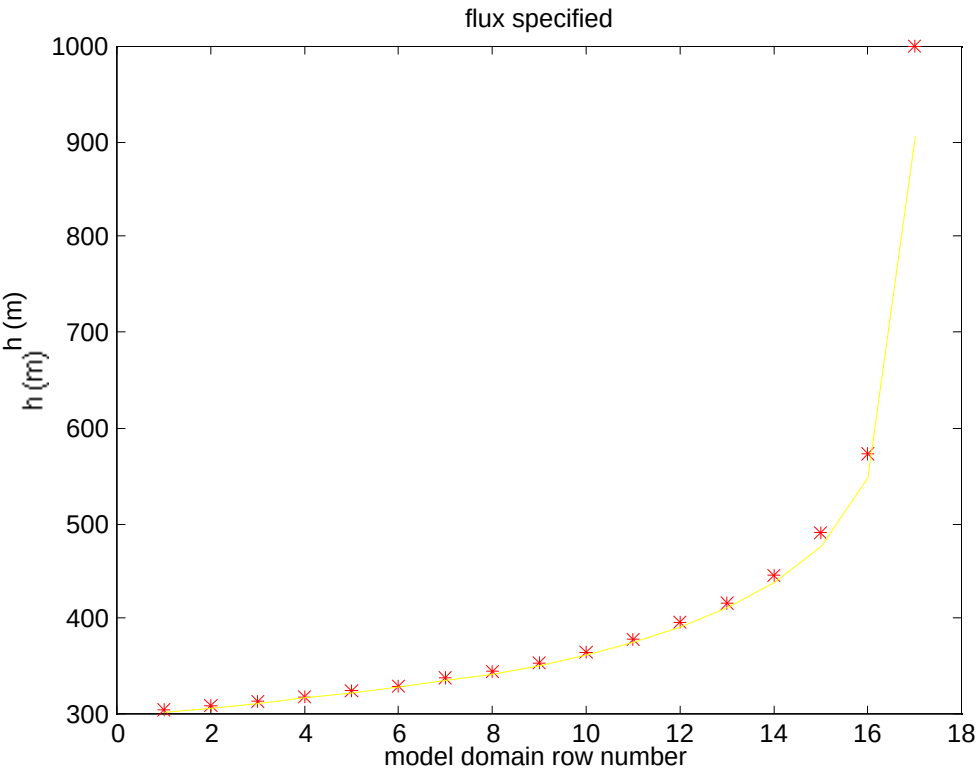

Figure 6.4: Steady-state ice thickness (dimensional units) for a for a model run in which the entire back boundary has ice-stream input (flux specified). Asterisks denote the exact solution obtained by Van der Veen.

6.4 Revision of EISMINT Ice-Shelf Model Test

To confirm the insight into the cause of the mass imbalance suggested above, the EISMINT test described in the previous chapter was re-run, but this time with ice flux specified (instead of ice thickness) at the upstream boundary. To be as careful as possible to avoid other problems which may crop up as a result of implicit time stepping and wiggle control, the time step size was taken to be approximately 14 days, artificial diffusion was eliminated, and the positive-definite constraint on ice thickness was relaxed. (Negative ice thickness was allowed in all routines except the diagnostic routine which solves the momentum balance, in this routine, negative ice thickness was interpreted as zero ice thickness.) The results are shown in Figs. (6.5) and (6.6). (In this particular run, the ice shelf evolved for 740 years. A longer run-time is necessary to produce an exact mass balance. Due to the very short time steps, 14 days, and the slow Macintosh computer, I did not have the patience to wait for the model run to complete once the mass balance was satisfied to better than 1%.)

The insight gained in the previous section is confirmed: the specification of flux at the ice-stream inlet produces a numerical solution that is in mass balance:

Input	8.4000×10^9 m^3/a
Output	-8.4823×10^9
net	0.0823×10^9
percent	1 %

Note: The net figures for the mass influx have changed in this particular run because the width of the ice-stream inlet was reduced by one element spacing.

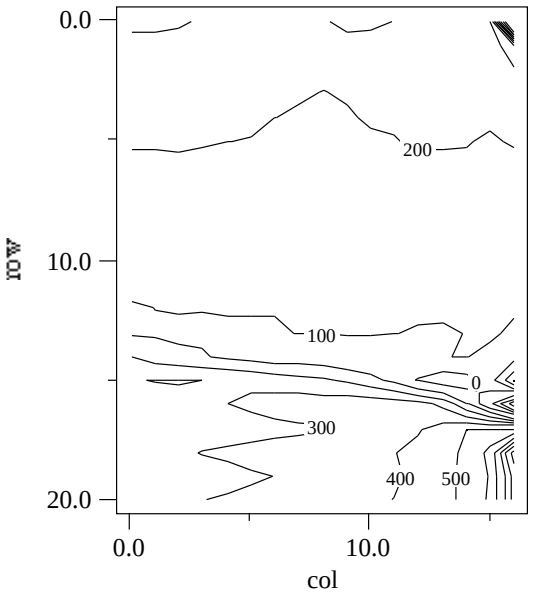

Figure 6.5: Steady-state ice thickness (dimensional units) for a for a model run in which the ice flux was specified as the ice-stream inlet boundary condition in the prognostic equation. Negative ice thickness occurs at the "corner" at the edge of the ice-stream inlet.

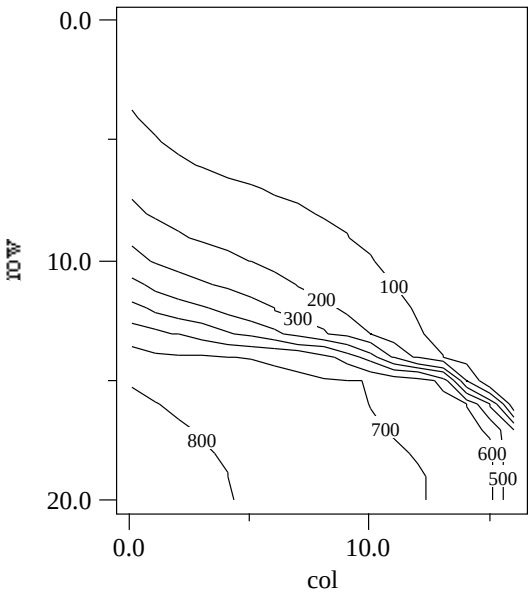

Figure 6.6: Steady-state ice velocity magnitude (dimensional units) for a for a model run in which the ice flux was specified as the ice-stream inlet boundary condition in the prognostic equation.

6.5 Conclusion

The above analysis *quiets* the anxiety over the lack of mass conservation, but does not *comfort* the intellect. The lack of mass balance appears to be associated with an aspect of the problem being solved (the inaccuracy stemming from poor spatial resolution), not the numerical method. (By no means, however, do I imply that there are not better, more mass conserving, numerical methods to be explored). The lack of intellectual comfort stems from the fact that the problem of ice-stream influx, and specifically the "corners" which generate much of the numerical error in mass conservation, is unsolved.

Chapter 7

Ice-Stream Flow Over Sticky Spots: An Inverse Problem

Ian Whillans and Kees Van der Veen (1993) analyzed surface-velocity data from ice stream B, Antarctica to deduce its basal shear stress. A controversial result of their analysis was that basal drag is *negative* in sign over some small portions of the bed. (By negative, we mean that the basal drag tends to push the ice stream forward rather than impede it's flow.) This result has led to a great deal of speculation.

In an effort to understand the nature of ice-stream flow and to understand the analytical process by which we convert measurements of surface velocity to basal friction, we examine an idealized problem in which a flat, slab like ice stream flows across a sticky spot in the bed.

7.1 Stress Balance in a Simple Geometry

Consider the problem of ice flow over a sticky spot depicted in Fig. (7.1). We shall assume that the flow is purely two dimensional and can be described by a horizontal velocity $u(x, z)$ and a vertical velocity $w(x, z)$. Variation in the

transverse horizontal coordinate y is disregarded. Flow at the upstream and downstream ends of the domain is assumed to be independent of z (uniform over depth) and parallel to the bed. Flow at the downstream end is assumed to have a longitudinal strain rate of U/L, a typical ice-stream strain rate, where U is a characteristic horizontal velocity (on the order of 500 m/year) and L is the length of the domain, which we take to be much greater than the ice thickness Z, which is also assumed uniform with x. The upper surface of the ice stream is assumed flat and stress free. The lower surface is assumed to be stress free except for a small localized sticky spot with a Gaussian distribution in x. In other words, we assume that the basal shear stress τ is equal to $\beta(x)^2 u(z = -Z)$, where $\beta(x)^2$ is a positive-definite (hence the square) basal friction coefficient that is a function of x.

Nondimensional Variables

To simplify the arithmetic, we adopt nondimensional variables by employing the following scale transformations:

$$u \rightarrow Uu \tag{7.1}$$
$$w \rightarrow U\delta w \tag{7.2}$$
$$e_{xx} \rightarrow \frac{U}{L} e_{xx} \tag{7.3}$$
$$e_{zz} \rightarrow \frac{U}{L} e_{zz} \tag{7.4}$$
$$e_{xz} \rightarrow \frac{U}{L} \delta^{-1} \frac{1}{2} \left(u_z + \delta^2 w_x \right) \tag{7.5}$$
$$\tau \rightarrow \tau_o \tau \tag{7.6}$$
$$\sigma \rightarrow \rho g Z \sigma \tag{7.7}$$
$$p \rightarrow \rho g Z p \tag{7.8}$$

where $e_{xx} = u_x$, $e_{zz} = w_z$, and e_{xz} are the strain rate components, σ is the normal stress at the bed, and p is the pressure. The scaling of w and e_{zz} is motivated by the incompressibility condition ($e_{zz} = -e_{xx}$). The subscripts on u and w denote partial differentiation.

Stress Balance Equations

Using the above-defined nondimensional variables, the stress-equilibrium equations and their boundary conditions at $z = 0$ and $z = -1$ (the bed) are written as follows. At $z = 0$, the stress-free boundary conditions are

$$u_z + \delta^2 w_x = 0 \tag{7.9}$$

$$w_z - \gamma p = 0 \tag{7.10}$$

In the interior of the ice,

$$u_{xx} - \gamma p_x + \delta^{-2}\frac{1}{2}\left(u_{zz} + \delta^2 w_{xz}\right) = 0 \tag{7.11}$$

$$w_{zz} - \gamma p_z + \frac{1}{2}\left(u_{zx} + \delta^2 w_{xx}\right) = \gamma \tag{7.12}$$

At the lower boundary at $z = -1$,

$$u_z = 2\epsilon\tau \tag{7.13}$$

$$-w_z + \gamma p = \gamma\sigma \tag{7.14}$$

In the above equations we have made use of the fact that $w = 0 = w_x$ at the basal boundary.

In the above equations there are three dimensionless parameters,

$$\delta = \frac{Z}{L} \tag{7.15}$$

$$\epsilon = \frac{\tau_o}{2\nu_o \frac{U}{Z}} \tag{7.16}$$

$$\gamma = \frac{\rho g Z}{2\nu_o \frac{U}{L}} \tag{7.17}$$

where ν_o is the ice viscosity which is assumed constant. The parameter δ represents the aspect ratio of the flow. This parameter is small when the horizontal scale of flow variation is greater than the ice thickness. The parameter ϵ describes the strength of the basal friction. In circumstances where the ice stream flow is large, and predominantly by basal sliding (or

by basal sediment deformation), the parameter ϵ will be much smaller than 1. For ice stream B, in West Antarctica, ϵ is about 3×10^{-2} even for basal friction scales (τ_o) as high as one bar (10^{15} Pa).

To proceed further, we must assume a relation between the three dimensionless parameters. Here we assume the following simple relation

$$\delta^2 = \epsilon = \gamma^{-1} \tag{7.18}$$

In this circumstance, the above governing equations simplify to the following form (here we also make use of the incompressibility condition, $w_z = -u_x$). At $z = 0$,

$$u_z + \epsilon w_x = 0 \tag{7.19}$$

$$-\epsilon u_x - p = 0 \tag{7.20}$$

In the interior of the ice ($-1 < z < 0$),

$$\frac{1}{2}u_{zz} - p_x + \epsilon\left(u_{xx} + \frac{1}{2}w_{xz}\right) = 0 \tag{7.21}$$

$$-\left(p_z + 1\right) - \epsilon\frac{1}{2}u_{xz} + \epsilon^2\frac{1}{2}w_{xx} = 0 \tag{7.22}$$

At the lower boundary ($z = -1$),

$$u_z = 2\epsilon\tau \tag{7.23}$$

$$p - \sigma + \epsilon u_x = 0 \tag{7.24}$$

Series Expansion on Small Parameters

We find the above equations too difficult to solve as they stand because of the mixture of large and small terms (small terms are defined to be the ones which contain a factor of ϵ). To proceed further, we use the systematic simplification provided by the series expansion on the small parameter ϵ. We write the variables as follows,

$$u \;=\; u^{[0]} + \epsilon u^{[1]} + \epsilon^2 u^{[2]} + \cdots \tag{7.25}$$

$$
\begin{aligned}
w &= w^{[0]} + \epsilon w^{[1]} + \epsilon^2 w^{[2]} + \cdots && (7.26)\\
p &= p^{[0]} + \epsilon p^{[1]} + \epsilon^2 p^{[2]} + \cdots && (7.27)\\
\sigma &= \sigma^{[0]} + \epsilon \sigma^{[1]} + \epsilon^2 \sigma^{[2]} + \cdots && (7.28)\\
& && (7.29)
\end{aligned}
$$

When these expressions are substituted into the above equations, and when terms of equal magnitude (equal power of ϵ) are collected, the result is a sequence of problems which may be solved systematically to produce as many of the terms in the above expansion as are desired.

To illustrate the benefit of the series expansion in a simple example, consider the single variable equation (which has nothing to do with glaciology)

$$u_x - \epsilon u = 0 \tag{7.30}$$

As it stands, this equation is a major pain because of the fact that a large, order 1 term is compared with a small, order ϵ term. Substitution of $u^{[0]} + \epsilon u^{[1]} + \epsilon^2 u^{[2]} + \cdots$ for u in the above equation simplifies the situation. After collecting terms, the above equation takes the following form

$$u_x^{[0]} + \epsilon \left\{ u_x^{[1]} - u^{[0]} \right\} + \epsilon^2 \left\{ u_x^{[2]} - u^{[1]} \right\} + \cdots = 0 \tag{7.31}$$

The above equation contains many more terms, but it possess the advantage that each term appearing in brackets contains subterms that are the same size (order 1). To solve the above equation, we simply set each of the terms in brackets to zero to get the following sequence of simpler differential equations:

$$
\begin{aligned}
u_x^{[0]} &= 0 && (7.32)\\
u_x^{[1]} - u^{[0]} &= 0 && (7.33)\\
u_x^{[2]} - u^{[1]} &= 0 && (7.34)\\
\text{etc.} & && (7.35)
\end{aligned}
$$

The above equations are referred to as the zeroth order, first order, second order, etc. problems. A typical effort to solve them will entail truncation at some level. The size of the errors generated by this truncation has the magnitude of ϵ^n, where n is the truncation level. This estimate of the size of approximation errors constitutes one of the principle motivations for using the series expansion approach.

7.2 Zeroth-Order Problem

We proceed to determine the solution for the ice-stream flow over the sticky spot by using a series expansion on ϵ. The zeroth-order equations are written as follows. At $z = 0$,

$$u_z^{[0]} = 0 \tag{7.36}$$

$$p^{[0]} = 0 \tag{7.37}$$

In the ice interior,

$$\frac{1}{2}u_{zz}^{[0]} - p_x^{[0]} = 0 \tag{7.38}$$

$$p_z^{[0]} + 1 = 0 \tag{7.39}$$

At $z = -1$,

$$u_z^{[0]} = 0 \tag{7.40}$$

$$p^{[0]} - \sigma^{[0]} = 0 \tag{7.41}$$

Zeroth-Order Solution

We proceed by first computing the pressure using Eqn. (7.39). Integration over the vertical dimension of the ice stream (from -1 to 0), and use of the boundary conditions at the surface and base gives

$$p^{[0]} = -z \tag{7.42}$$

$$\sigma^{[0]} = 1 \tag{7.43}$$

This is exactly what we expected; the pressure is hydrostatic and the normal stress on the bed is simply the weight of the ice above.

Integration of Eqn (7.38) over z and use of either of the two boundary conditions gives another expected result

$$u_z^{[0]} = 0 \tag{7.44}$$

Namely, the horizontal velocity (at zeroth order) is independent of z.

The incompressibility condition, and the above result, may be used to show that

$$w^{[0]} = -u_x^{[0]}(1+z) \tag{7.45}$$

We have now exhausted the zeroth-order equations, but have yet to completely describe the zeroth-order solution. In particular, we have not yet found a way to describe the x-variation of the z-independent horizontal velocity $u^{[0]}$. Even more important, we have not yet encountered the basal stress, the aspect of the problem which motivates our analysis. To get at these inadequacies of the zeroth-order problem, we proceed to examine the first-order problem.

7.3 First-Order Problem

The first-order equations are written as follows. At $z = 0$,

$$u_z^{[1]} + w_x^{[0]} = 0 \tag{7.46}$$

$$-u_x^{[0]} - p^{[1]} = 0 \tag{7.47}$$

In the ice interior,

$$\frac{1}{2}u_{zz}^{[1]} - p_x^{[1]} = -u_{xx}^{[0]} - \frac{1}{2}w_{xz}^{[0]} = -\frac{1}{2}u_{xx}^{[0]} \tag{7.48}$$

$$-p_z^{[1]} = \frac{1}{2}u_{xz}^{[0]} = 0 \tag{7.49}$$

At $z = -1$,

$$u_z^{[1]} = 2\tau \tag{7.50}$$

$$u_x^{[0]} + p^{[1]} - \sigma^{[1]} = 0 \tag{7.51}$$

In the interior equations above, we made use of the facts that $u_z^{[0]} = 0$ and $w_{xz}^{[0]} = \frac{\partial^2}{\partial x \partial z} - u_x^{[0]}(1+z)$.

First-Order Solution

The solution to Eqn. (7.49) is obtained by integrating over z and making use of the upper boundary condition (integration is easy because $u_x^{[0]}$ is independent of z). To satisfy the lower boundary condition, we determine $\sigma^{[1]}$. The results are,

$$p^{[1]} = -u_x^{[0]} \tag{7.52}$$

$$\sigma^{[1]} = 0 \tag{7.53}$$

Making use of the first-order solution for pressure, $p^{[1]}$, in Eqn. (7.48), we obtain

$$u_z^{[1]} = u_{xx}^{[0]} \tag{7.54}$$

at $z = 0$,

$$u_{zz}^{[1]} = -3u_{xx}^{[0]} \tag{7.55}$$

in the interior, and

$$u_z^{[1]} = 2\tau \tag{7.56}$$

at $z = -1$. Integration of Eqn. (7.55) over $-1 < z < 0$, and use of the boundary conditions gives the equation we need to close the zeroth-order solution (*i.e.,* determine the x-dependence of $u^{[0]}$):

$$2u_{xx}^{[0]} = \tau \tag{7.57}$$

Now that we have determined $u_{xx}^{[0]}(x)$ (Eqn. 7.57), we can move on to determine $u^{[1]}$ using Eqn. (7.55) directly (*i.e.,* without integrating it over z first). Substitution for $u_{xx}^{[0]}$ renders Eqn. (7.55) and its boundary conditions into the following forms. At $z = 0$,

$$u_z^{[1]} = \frac{\tau}{2} \tag{7.58}$$

In the interior,

$$u_{zz}^{[1]} = -\frac{3}{2}\tau \tag{7.59}$$

At $z = -1$,

$$u_z^{[1]} = 2\tau \tag{7.60}$$

Integrating Eqn. (7.59) once gives,

$$u_z^{[1]} = \frac{\tau}{2}(1 - 3z) \tag{7.61}$$

Integrating a second time gives,

$$u^{[1]} = \frac{\tau}{2}\left(z - \frac{3}{2}z^2\right) + c(x) \tag{7.62}$$

where $c(x)$ is an as yet undetermined constant of integration (that can be a function of x).

We note for future reference the following relations (the second of which is derived from the first using the incompressibility condition):

$$u_x^{[1]} = \frac{1}{2}\left(z - \frac{3}{2}z^2\right)\tau_x + c_x \tag{7.63}$$

$$w_z^{[1]} = \frac{-1}{2}\left(z - \frac{3}{2}z^2\right)\tau_x - c_x \tag{7.64}$$

At this stage, we have not only recovered the zeroth-order, depth-independent horizontal flow $u^{[0]}$, but we have almost additionally recovered the most sizable depth-dependent correction to the horizontal flow that is generated by the basal friction, namely $u^{[1]}$. We still have work to do because we have only recovered $u^{[1]}$ up to the constant of integration $c(x)$. To get $c(x)$, we must consider the second-order equations (much in the same way as the determination of the x-dependence of $u^{[0]}$ required us to consider the first-order equations).

7.4 Second-Order Problem

The second-order equations are written as follows. At $z = 0$,

$$u_z^{[2]} + w_x^{[1]} = 0 \tag{7.65}$$

$$-u_x^{[1]} - p^{[2]} = 0 \tag{7.66}$$

In the ice interior,

$$\frac{1}{2}u_{zz}^{[2]} - p_x^{[2]} = -u_{xx}^{[1]} - \frac{1}{2}w_{xz}^{[1]} \tag{7.67}$$

$$-p_z^{[2]} = \frac{1}{2}u_{xz}^{[1]} - \frac{1}{2}w_{xx}^{[0]} \tag{7.68}$$

At $z = -1$,

$$u_z^{[2]} = 0 \tag{7.69}$$

$$u_x^{[1]} + p^{[2]} - \sigma^{[2]} = 0 \tag{7.70}$$

Second-Order Solution

We begin by making use of what we have derived before, namely

$$w_{xx}^{[0]} = -u_{xxx}^{[0]}(1+z) \tag{7.71}$$

$$u_{xz}^{[1]} = \frac{1}{2}(1-3z)\tau_x \tag{7.72}$$

$$u_x^{[1]}(z=0) = c(x) \tag{7.73}$$

These relations may be substituted into Eqn. (7.68) and its attendant boundary conditions to get the following result. At $z = 0$

$$-u_x^{[1]} - p^{[2]} = 0 \tag{7.74}$$

In the ice interior,

$$-p_z^{[2]} = \frac{1}{4}(1-3z)\tau_x + \frac{1}{2}u_{xxx}^{[0]}(1+z) \tag{7.75}$$

At $z = -1$,

$$u_x^{[1]} + p^{[2]} - \sigma^{[2]} = 0 \tag{7.76}$$

Integration of Eqn. (7.75) gives an expression for $p^{[2]}$:

$$p^{[2]} = \frac{1}{4}\left(z - \frac{3}{2}z^2\right)\tau_x + \frac{1}{2}u_{xxx}^{[0]}\left(z + \frac{z^2}{2}\right) + d(x) \tag{7.77}$$

where $d(x)$ is a constant of integration that can be a function of x. Use of the boundary condition at $z = 0$ gives $d(x) = c(x)$. Use of the boundary condition at $z = -1$ determines that $\sigma^{[2]} = \frac{-1}{8}\tau_x$.

Having derived $p^{[2]}$ we are now in a position to integrate Eqn. (7.67) over $-1 < z < 0$ to generate the desired equation for $c(x)$. We first note that

$$\int_{-1}^{0} p_x^{[2]} dz = \frac{-\tau_{xx}}{4} - \frac{u_{xxxx}^{[0]}}{6} + c_{xx} \tag{7.78}$$

Integrating Eqn. (7.67) over z, and use of the boundary conditions ($u_z^{[2]} + w_x^{[1]} = 0$ at $z = 0, -1$), gives

$$0 = \int_{-1}^{0} p_x^{[2]} dz - \frac{\tau_{xx}}{4} + c_{xx} \tag{7.79}$$

Making use of the expression for the integral of $p_x^{[2]}$ derived above, we obtain

$$0 = \frac{-\tau_{xx}}{2} - \frac{u_{xxxx}^{[0]}}{6} + 2c_{xx} \tag{7.80}$$

Making use of the relation derived previously that $2u_{xxxx}^{[0]} = \tau_{xx}$, we obtain the desired equation for $c(x)$:

$$c_{xx} = \frac{7}{32}\tau_{xx} \tag{7.81}$$

We may integrate the above equation at once to obtain,

$$c(x) = \frac{7}{32}\tau + ax + b \tag{7.82}$$

where a and b are constants of integration. In our problem, we shall take these constants to be zero (*i.e.,* we shall assume that the second-order correction to the flow is zero at the upstream and downstream boundaries at $x = 0, 1$). We may thus write the complete first-order horizontal velocity correction as follows:

$$u^{[1]} = \frac{\tau}{2}\left(z - \frac{3}{2}z^2\right) + \frac{7}{32}\tau \tag{7.83}$$

7.5 Flow of an Ice Stream Over a Sticky Spot

In the somewhat lengthy analysis above, we have reduced a very complicated mathematical description of stress balance to a level that can be easily tackled with simple numerical tools. (An exact, analytic solution may be possible, but has not been tried here.) The complicated problem reduces to solving for $u(x,z) = u^{[0]}(x) + u^{[1]}(x,z)$ the simple equations for $u^{[0]}$ and $u^{[1]}$ developed above. To proceed further, we express the basal friction $\tau(x)$ using the following representation:

$$\tau = \beta^2(x) \left(u^{[0]} + \epsilon u^{[1]} \|_{z=-1} \right) \tag{7.84}$$

It is important to note that the above expression does not represent a basal friction "law" in the traditional sense; but rather a basal friction "representor" in the sense used by Bennett (1992). With the representation shown above, $\tau(x)$ can have an arbitrary functional form with only one restriction, namely, that it be positive (hence the representation of the basal friction coefficient, β, as a square). For a Gaussian sticky spot, we assume

$$\beta(x) = 2.5 \exp\left(- \left(\frac{x - .5}{0.15} \right)^2 \right) \tag{7.85}$$

(The somewhat arbitrary choice of parameters in the above definition of $\beta(x)$ is designed to give a pleasing result, *e.g.,* to make the inversion of the second-order accurate surface velocity teach us a valuable lesson.)

To determine $u(x,z)$ to second-order accuracy (*i.e.,* neglecting terms of ϵ^2 and smaller), we must solve the following coupled equations:

$$2u^{[0]}_{xx} - \beta^2 u^{[0]} = \beta^2 \epsilon u^{[1]} \|_{z=-1} \tag{7.86}$$

$$u^{[0]} = 1 \qquad \text{for} \quad x = 0 \tag{7.87}$$

$$u^{[0]}_x = 1 \qquad \text{for} \quad x = 1 \tag{7.88}$$

and

$$u^{[1]} = \beta^2 \left(\frac{z}{2} - \frac{3z^2}{4} + \frac{7}{32} \right) \left(u^{[0]} + \epsilon u^{[1]} \right) \tag{7.89}$$

The above equations are difficult to solve as they currently stand, because both $u^{[0]}$ and $u[1]\|_{z=-1}$ appear in the same equation. These equations can be decoupled by using Eqn. (7.89) to express $u^{[1]}|_{z=-1}$ in terms of $u^{[0]}$, *i.e.,*

$$u^{[1]}|_{z=-1} = \frac{\frac{-33}{32}\beta^2}{1+\frac{33}{32}\beta^2\epsilon}u^{[0]} \tag{7.90}$$

and by substituting the result into Eqn. (7.86). The result of these manipulations is a single, second-order ordinary differential equation in a single unknown:

$$2u^{[0]}_{xx} - \frac{u^{[0]}}{1+\frac{33}{32}\beta^2\epsilon} = 0 \tag{7.91}$$

This equation requires two boundary conditions, the two listed in Eqns. (7.87) and (7.88) will do just fine. Once Eqn. (7.90) is solved for $u^{[0]}(x)$, Eqn. (7.90) is used to determine $u^{[1]}(x,z)$.

Discretization

We find it convenient to use a finite-difference method to solve Eqn. (7.91). (An exact analytic solution was not attempted, but should be relatively straightforward given the simplicity of the functional form of $\beta(x)$ and the differential equation.) We discretized the domain of solution $x \in [0,1]$ into $N = 100$ grid points, and assigned the value of $u^{[0]}$ at each grid point to correspond with an element of the vector $\mathbf{u}^{[0]} \in \mathcal{R}^N$, *i.e.,*

$$\mathbf{u}^{[0]} = \begin{bmatrix} u^{[0]}_1 \\ u^{[0]}_2 \\ \vdots \\ u^{[0]}_N \end{bmatrix} \tag{7.92}$$

The discrete form of Eqn. (7.91) with its boundary conditions is expressed as a simple matrix equation:

$$\mathbf{A}\mathbf{u}^{[0]} = \mathbf{r} \tag{7.93}$$

where,

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \cdots & & & \\ \frac{2}{\Delta x^2} & \frac{-4}{\Delta x^2} - D_2 & \frac{2}{\Delta x^2} & 0 & \cdots & \\ 0 & \frac{2}{\Delta x^2} & \frac{-4}{\Delta x^2} - D_3 & \frac{2}{\Delta x^2} & 0 & \cdots \\ & & & \ddots & & \\ & \cdots & 0 & \frac{2}{\Delta x^2} & \frac{-4}{\Delta x^2} - D_{N-1} & \frac{2}{\Delta x^2} \\ & & \cdots & 0 & \frac{-1}{\Delta x} & \frac{1}{\Delta x} \end{bmatrix} \tag{7.94}$$

$$\mathbf{r} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{7.95}$$

and $D_i = \frac{1}{1+\frac{33}{32}\beta^2(x_i)\epsilon}$, where x_i is the x-coordinate of the i'th grid point.

The solution to Eqn. (7.93) is readily obtained using MATLAB as suggested by the MATLAB script listed below. Once $\mathbf{u}^{[0]} = \mathbf{A}^{-1}\mathbf{r}$ is obtained, $u^{[1]}(x, z)$ in either continuous or discrete form easily follows. The solution so obtained is displayed in Figs. (7.2) - (7.4). What is distinctive about the solution is the fact that, despite $\epsilon << 1$, the velocity field has appreciable vertical structure over the location where $\beta(x)$ is maximum. The vertically averaged horizontal velocity and basal velocity both decrease in the neighborhood of the sticky spot from what their values would be if $\beta = 0$ (Fig. 7.3). A surprising result, however, is that the velocity at the surface, $u^{[0]} + \epsilon u^{[1]}(x, z = 0)$, shows a "bump" or localized increase over the sticky spot. As we shall soon see, this bump in surface velocity causes a great deal of trouble when we attempt to deduce the basal drag from the surface velocity using only first-order accurate ice-stream flow dynamics.

Script for Second-Order Accurate Ice-Stream Flow

```
epsilon=0.1;
N=100;
x=linspace(0,1,N);
```

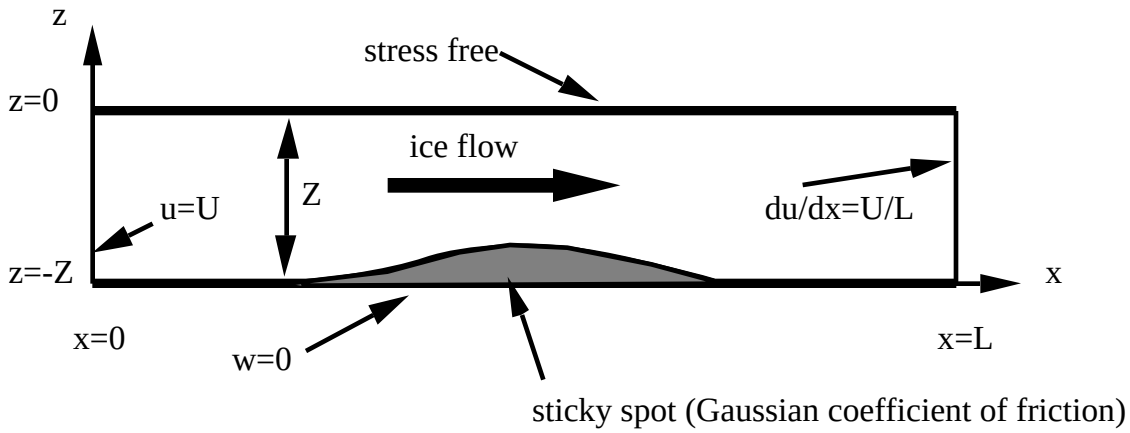

Figure 7.1: Simple one-dimensional flow geometry used to depict ice-stream flow over a basal adhesion commonly referred to as a sticky spot.

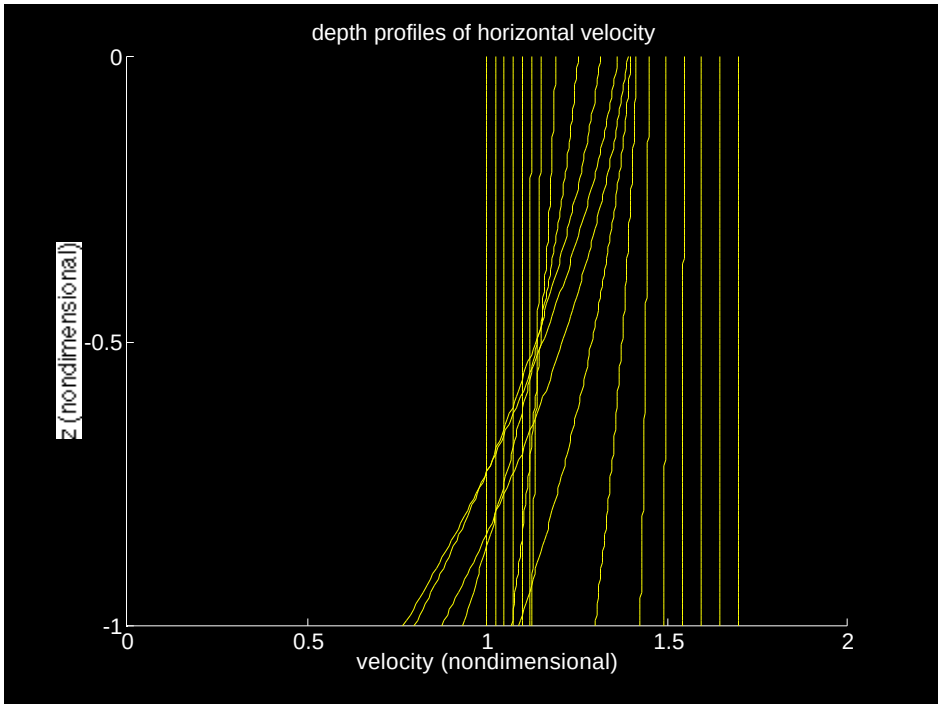

Figure 7.2: Velocity profiles at regular distances along the flow profile. Note the z-structure generated above the sticky spot.

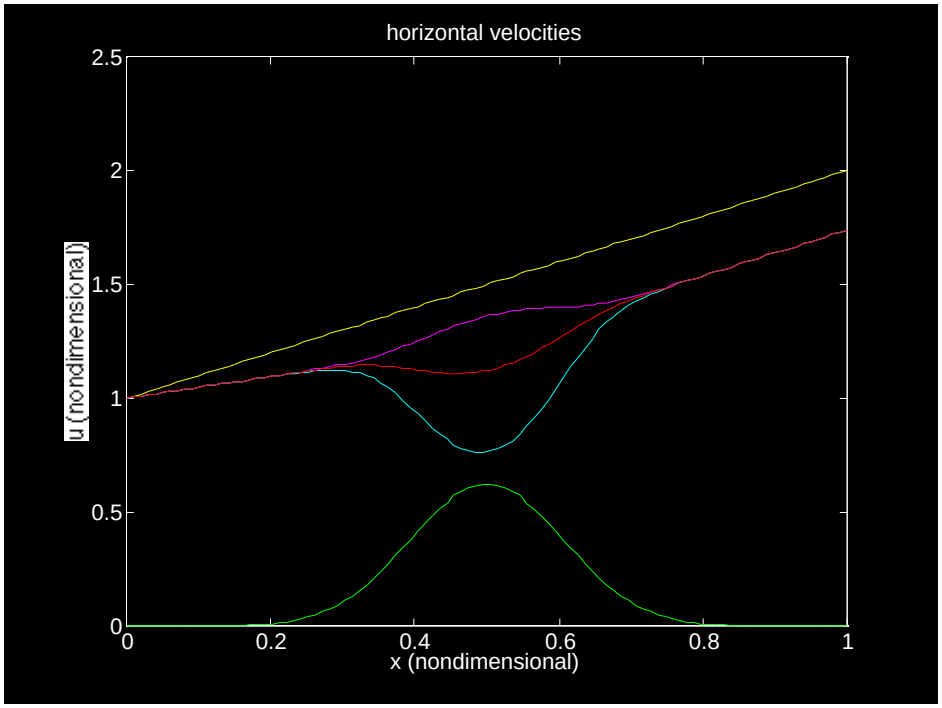

Figure 7.3: Horizontal velocity plotted as a function of x. Upper most curve represents the stress-free solution in which $\beta(x) = 0$. Second upper most curve represents the surface velocity accurate to first order. Next curve represents the depth-averaged velocity. The second-to-last lowest curve represents the bottom velocity. The bottom curve represents $\beta(x)$, and is given (not to scale) for reference. As anticipated, the depth-averaged velocity and the bottom velocity are both reduced in the region of the sticky spot. The surface velocity appears to be larger than expected over the region of the sticky spot.

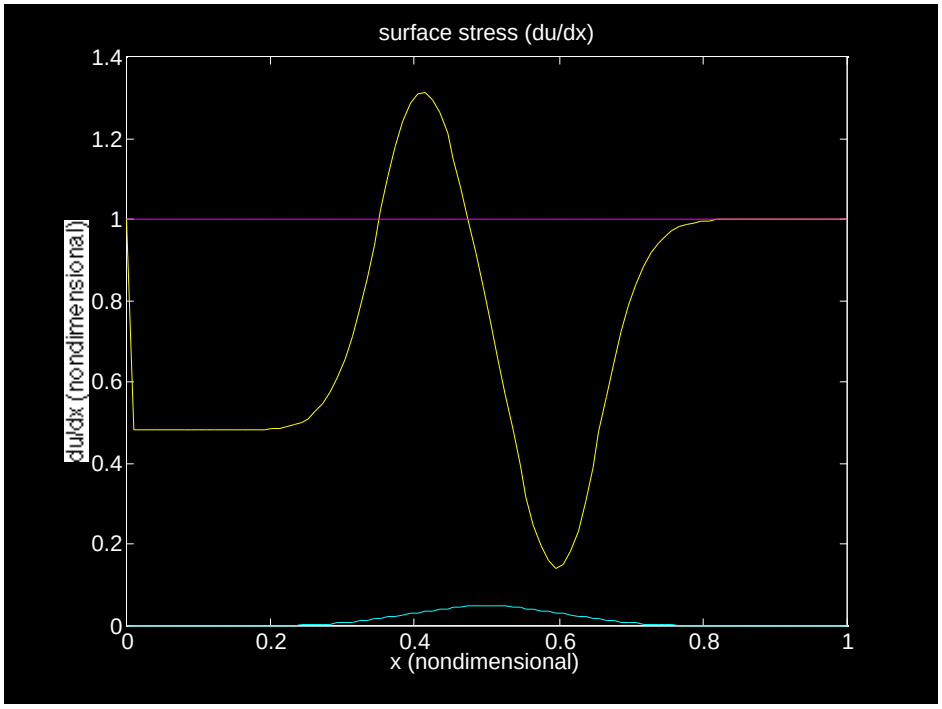

Figure 7.4: Longitudinal deviatoric stress, u_x, as a function of x (wavy curve). For reference, the longitudinal deviatoric stress is 1 when the sticky spot is absent. It is surprising that u_x exceeds its basal stress-free counterpart in some regions.

```
dx=1/(N-1);
dampscale=0.15;
beta=2.5*exp( -((x-.5)/dampscale).^ 2);


A=zeros(N,N);
RHS=zeros(N,1);


RHS(1)=1;
RHS(N)=1;
D=zeros(N,1);
A(1,1)=1;
A(N,N)=1/dx;
A(N,N-1)=-1/dx;


for n=2:N-1
D(n)= 1/2/(1+33/32 beta ^ 2 * epsilon);
A(n,n)=-2/dx^ 2-D(n);
A(n,n-1)=1/dx^ 2;
A(n,n+1)=1/dx^ 2;


end
As=sparse(A);
u=As\RHS;


tau=2*D.*u;
c=7/32*tau;


ubasal=u+epsilon*c-5/4*epsilon*tau;
usurf=u+epsilon*c;


uprofile=zeros(N,N);
```

```
z=[linspace(0,-1,N)]';
for n=1:N
uprofile(:,n)=u(n)+epsilon*(tau(n)/2*(z-3/2*z.^ 2) + c(n));
end


ubar=u+epsilon*(c-tau/2);


hold off
clg
axis([0 2.0 -1 0])
hold on
for n=1:5:N
plot(uprofile(:,n),z)
end


uslope=ones(N,1);
ufreeslope=ones(N,1);
deltaslope=zeros(N,1);
for n=2:N-1
uslope(n)=(usurf(n+1)-usurf(n-1))/(2*dx);
ufreeslope(n)=(ufree(n+1)-ufree(n-1))/(2*dx);
end
deltaslope=uslope-ufreeslope;
```

7.6 Inversion of Surface-Velocity Data

At this stage, having developed the above solution, we are ready to consider the question which originally motivated our interest in ice-stream flow over a sticky spot: If given surface velocity observations $u^d(x)$, what is the corresponding basal drag τ? We shall answer this question using two separate approaches to attempt a diagnosis of what is at the root of Whillans' and

Van der Veen's (1993) negative basal drag result. The first method will be to mimic the approach used by Whillans and Van der Veen in their analysis of ice stream B, Antarctica, surface-velocity data. The other approach will be that recommended by MacAyeal (1993). We shall learn that neither method is able to accurately resolve the basal drag because of inadequacy in the assumptions used by each method concerning the need for second-order accuracy in the treatment of the ice-stream flow physics. The strengths of each method compliment each other, suggesting that both methods have merits that may recommend one over the other in particular applications.

The two methods are similar in that neither has attempted to break the second-order accuracy "barrier". In other words, both methods assume that the surface velocity is adequately modeled by $u^{[0]}$ without need for $u^{[1]}$. In this circumstance, both methods assume that the surface velocity, which we shall describe using a separate variable $u^m(x)$ ("m" for model), satisfies the following flawed description of ice-stream flow dynamics:

$$2u^m_{xx} - \tau = 0 \tag{7.96}$$

subject to the boundary conditions that $u^m(0) = u^d(0)$ and $u^m(1) = u^d(1)$. Equation (7.96) should be contrasted with Eqn. (7.93) to see the difference between first-order accurate and second-order accurate flow dynamics.

The two methods differ, however in one key respect. In Whillans' and Van der Veen's (1993) method (which is described in detail in Van der Veen and Whillans (1989)), the basal drag $\tau(x)$ is allowed to be an arbitrary, and possibly negative, function of x. In MacAyeal's (1993) method, the basal drag is represented by a function that excludes the possibility of negative drag, *i.e.*, $\tau(x) = \beta^2_m(x)u^m(x)$.

Whillans' and Van der Veen's Method

The basal drag τ_{WVdV} deduced using Whillans' and Van der Veen's (1993) method is found by simply substituting $u^d(x)$ into Eqn. (7.96). The result is shown in Fig. (7.5). The distinctive pattern of negative-positive-negative basal drag over the sticky spot is reminiscent of the pattern of basal drag deduced using data from ice stream B.

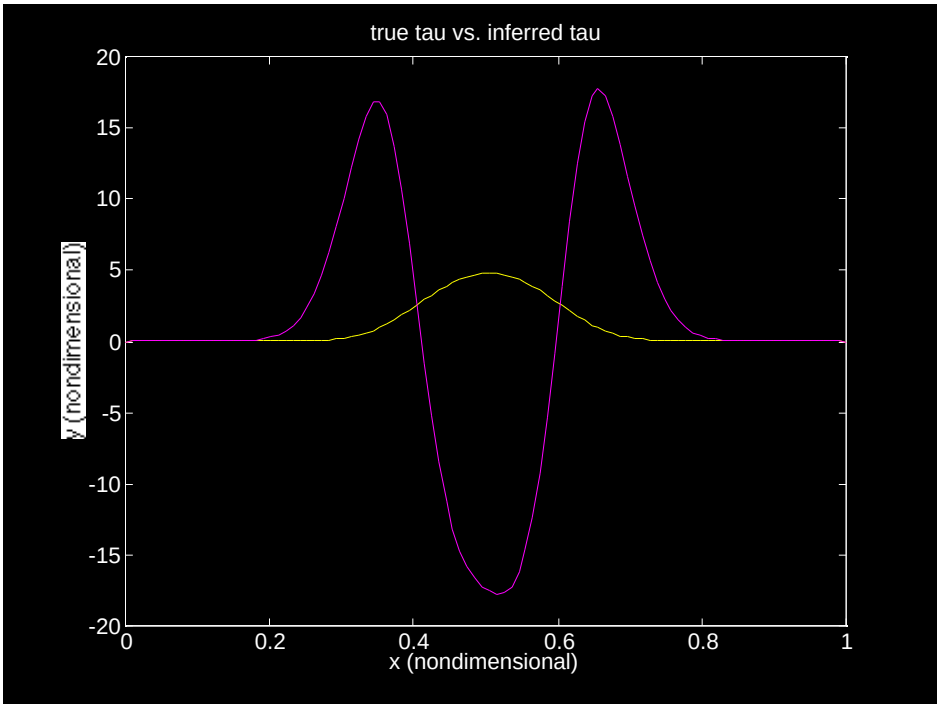

Figure 7.5: The basal drag τ_{wvdv} deduced using the Whillans and Van der Veen (1993) method with pseudo data $u^d(x) = u^{[0]}(x) + \epsilon u^{[1]}(x, z = 0)$. Note that τ_{wvdv} is negative on either side of the sticky spot. This negative basal drag is caused by the fact that first-order accurate physics cannot produce the "bump" in surface velocity over the sticky spot associated with second-order accurate physics without appeal to the power of an unphysical negative basal drag.

7.6.1 MacAyeal Method

The MacAyeal (1993) method works by attempting to minimize the following least square measure of model/data misfit:

$$J = \int_0^1 \frac{1}{2}\left(u^m(x) - u^d(x)\right)^2 dx + \int_0^1 \lambda(x)\left(2u^m_{xx} - \beta^2_m u^m\right) dx \qquad (7.97)$$

where $\lambda(x)$ is a Lagrange undetermined multiplier function who's purpose is to enforce the first-order physics as a constraint. The result of minimizing J is shown in Fig. (7.6). The MacAyeal method does not suffer from the negative basal drag problems of the Whillans and Van der Veen method; however, the MacAyeal method has problems of its own. First, the modelled velocity u^m does not exactly match the data u^d. Second, the inferred basal drag is low by as much as 50%. In some applications, this defect might be as bad as the negative basal drag deduced using the other method.

The MATLAB script used to generate the basal drag by the MacAyeal method is listed below. For a detailed description of the method, consult the paper by MacAyeal (1993).

```
% This program does the inverse problem using
% the flawed assumption that zero-order
% physics only is adequate to invert a first-order accurate surface
velocity.


   N=100;
x=linspace(0,1,N);
dx=1/(N-1);
lambda=zeros(N,1);
uinf=zeros(N,1);
dampscale=0.15;
beta0=2.5*exp( -((x'-.5)/dampscale).^ 2);
betatrue=2.5*exp( -((x'-.5)/dampscale).^ 2)
beta0(1)=0;
beta0(N)=0;
```

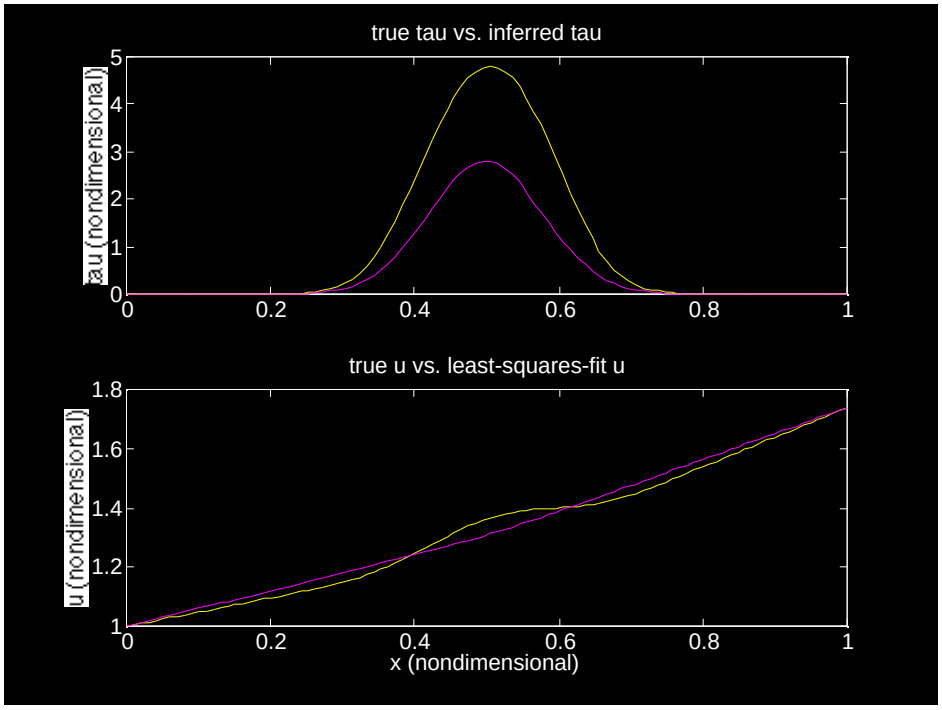

Figure 7.6: The basal drag τ_m (upper panel) deduced using the MacAyeal (1993) method with pseudo data $u^d(x) = u^{[0]}(x) + \epsilon u^{[1]}(x, z = 0)$. Note that τ_m has the proper shape of the sticky spot, but falls short in magnitude. This problem with the magnitude is caused by the fact that first-order accurate physics cannot produce the "bump" in surface velocity over the sticky spot associated with second-order accurate physics. The lower panel shows the mismatch between u^m and u^d, which is impossible to eliminate without appeal to second-order accurate flow dynamics.

```
A=zeros(N,N);
R=zeros(N,1);
R(1)=1;
R(N)=1;
A(1,1)=1;
A(N,N)=1;
R(N)=usurf(N);
for n=2:N-1
A(n,n)=-4/dx^ 2;
A(n,n-1)=2/dx^ 2;
A(n,n+1)=2/dx^ 2; end
A=sparse(A);
global A R lambda uinf usurf betatrial


   options=foptions;
initialJ=J(beta0);
options(9)=0;
options(1)=1;
options(3)=1.e-2;
options(2)=1.e-2;
beta=fminu('J',beta0,options,'grad');
plot(x,beta.^ 2)


   function [j] = J(trial)
global A R lambda uinf usurf betatrial
[rows cols]=size(trial);
B=spdiags(trial.^ 2,0,rows,rows);
B(1,1)=0;
B(rows,rows)=0;
uinf=(A-B)\R;
j = (uinf-usurf)'*(uinf-usurf);


   function [g] = grad(trial)
global A R lambda uinf usurf betatrial
```

```
[rows cols]=size(trial);
B=spdiags(trial.^ 2,0,rows,rows);
B(1,1)=0;
B(rows,rows)=0;
uinf=(A-B)\R;
rhs=-2*(uinf-usurf);
lambda=(A'-B')\rhs;
g = -2*lambda.*trial.*uinf;
g(1)=0;
g(rows)=0;
```

7.7 Conclusion

We have learned two important lessons as a result of our consideration of the idealized flow over a sticky spot. The first lesson is that the second-order physics which pertain to vertical shear of the horizontal velocity can generate structure in the surface velocity field that is incompatible with first-order physics. The incompatible structure found in the example considered was a small "bump" of increased velocity over the sticky spot. The second lesson is that inverse methods which make use of first-order physics only suffer serious defects in the basal drag that they infer from surface velocity data. These defects can lead to the kinds of problems noticed by Whillans and Van der Veen (1993) in their analysis of ice stream B data.

Chapter 8

EISMINT Summer-School Lesson: Temperature Profiles Associated with a Heinrich-event

One of the most difficult problems in ice-sheet modelling is that of coupling of ice-sheet flow dynamics with ice-sheet and basal thermodynamics. In this lesson we shall investigate the simplest, most basic aspects of ice-sheet thermodynamics with an eye toward understanding the numerical methods that have been applied to advective-diffusion type equations. The problem we shall use to illustrate these basic aspects comes from questions which arise in the study of a phenomenon called a Heinrich Event.

Heinrich Events are the great iceberg discharges into the North Atlantic which occurred every 7000-12000 years during the last glacial. These iceberg discharges are revealed to us through the discovery of sharply defined IRD layers in the North Atlantic sediments. The rock debris in these layers (called Heinrich Layers after their discoverer, Hartmut Heinrich of Germany) bears the fingerprint of Hudson Bay (detrital carbonate of Paleozoic age, lead-isotope ages associated with Churchill and Superior terrains). The prevailing

thought on the origin of Heinrich Events–or at lest the thought that has prevailed in this author's mind–is that they represent the product of great surges of the Hudson Strait ice stream which drained a large portion of the interior of the Laurentide Ice Sheet as shown in Fig. (8.1).

For an ice-stream surge to create an armada of debris-bearing icebergs, dirty basal ice must be frozen on to the bottom of ice columns traveling down the ice stream before they come afloat as icebergs. An ice-stream surge provides a plausible mechanism to accomplish this task. Initially, as the surge starts up, ice flow is very fast, basal friction is high, and basal melting rates are expected to be astronomical (modelling suggests that order 10's of meters of basal melting per year is possible). A few decades to a century of such strong basal melting preconditions the ice columns which have yet to be discharged through the mouth of the ice stream for strong basal freezing. In essence, the warmest ice is melted off the bottom of the ice columns, leaving behind very cold basal ice that possesses the temperature normally found high-up in the ice column. During the late stages of the surge, these preconditioned ice columns accrete debris-bearing ice that was stored as water in a basal till layer. Eventually, the debris-bearing ice columns are calved as icebergs into the North Atlantic where they capsize and drift with the prevailing wind and current toward the Mid North Atlantic IRD belt.

The problem we shall consider in this Lesson is based on an idealized notion of the history of an ice column moving down the Hudson Strait ice stream. Initially, the column will be subject to strong thinning rates (as the ice stream spreads horizontally in the same depth-independent manner as an ice shelf) and strong basal melting. After about 100 years (the estimated e-folding time-scale of an ice stream surge), the strong basal melting, which was previously prescribed, will be converted over to basal freezing for another 100 years. The basal freezing rate will be a function of the vertical heat flux, and hence the temperature gradient, at the bottom of the ice column.

While strongly simplified, this problem provides a sense of what is involved in Heinrich-event dynamics. In our effort to understand this problem, we shall learn about techniques for simulating (using finite-difference methods) the advective-diffusion equation which governs vertical heat transport in an ice sheet. We will learn about numerical instability, upwind differencing,

and the advantages of matched asymptotic expansions.

8.1 A Hudson Strait Ice-Column Thermodynamics Problem

Problem: Consider an ice-column that follows the central flowline down the Hudson Strait (Fig. 8.1). Determine the temperature-depth profile $T(z,t)$ of the ice column for the time period $t \in [0, 200]$ a. Assume that $T(z,t)$ is governed by the following equation and boundary conditions:

$$\frac{\partial T}{\partial t} + w(z,t)\frac{\partial T}{\partial z} = \kappa \frac{\partial^2 T}{\partial z^2} \qquad 0 < z < h(t) \tag{8.1}$$

$$T(z = h(t), t) = \Theta \tag{8.2}$$

$$T(z = 0, t) = 0 \tag{8.3}$$

$$T(z, t = 0) = \frac{z\Theta}{h(t = 0)} \tag{8.4}$$

with ice diffusivity $\kappa = 1.4 \times 10^{-6}$ m^{2}s^{-1}, constant surface temperature $\Theta = -30$ C, linearly varying vertical velocity $w(z) = \dot{b} - \dot{e}z$, basal freezing rate $\dot{b}$ (positive for freezing), ice thickness $h(t)$, constant longitudinal strain rate $\dot{e}$, time t, and vertical coordinate z ($z = 0$ at the ice/bed interface, and $z = h(t)$ at the upper free surface). A number of simplifications are expressed above. First, the surface temperature is assumed constant with time. This simplification ignores the surface-temperature change associated with changing surface elevation of the ice column. Second, thermal diffusivity (and other thermal parameters) are independent of temperature and z. Third, the basal temperature is constrained at 0 C, not the pressure-melting point (which varies by up to a degree). Finally, the initial temperature profile is assumed linear. In nature, it is highly improbable that such a linear temperature profile could be constructed during the slow growth of the ice cover above Hudson Strait. Horizontal variation in T is not considered in

Figure 8.1: Map of the North Atlantic region effected by Heinrich Events. The line indicated in Hudson Strait suggests a possible path an ice column might slide along during a surge of the Hudson Strait ice stream. Our task in this Lesson is to predict the temperature depth profile of this ice column as it proceeds down the path toward the iceberg calving front.

this problem because of the smallness of horizontal conduction and the fact that the problem "follows" the ice column in the Lagrangian sense as it flows down the ice stream.

The ice-stream surge influences the ice-column temperature profile through two effects: basal freezing and longitudinal spreading (and hence thinning of the ice column). Basal freezing is divided arbitrarily into two time regimes. First a massive melting regime lasts for approximately 100 a. This melting period begins with a bang (-10 m a^{-1}), but diminishes with time as the surge progresses. After the melting period, the basal freezing rate is determined by the vertical heat flux through the ice column. During the initial melting stage, the effects of vertical heat flux through the ice and geothermal flux are ignored. The effects of geothermal flux and frictional dissipation are ignored during the second stage. With these assumptions, $\dot{b}(t)$ becomes

$$\dot{b} = \begin{cases} \dot{b_o} e^{\frac{-t}{\tau}} & \text{if } 0 \leq t < t_o \\ \\ \frac{-k}{\rho \mathcal{L}} \frac{\partial T}{\partial z}|_{z=0} & \text{if } t \geq t_o \end{cases} \tag{8.5}$$

where the thermal conductivity $k = 2$ W m^{-1} C^{-1}, ice density $\rho = 917$ kg m^{-3}, ice-stream surge time scale $\tau = 100$ a, $\dot{b}_o = -10$ m a^{-1}, the latent heat of fusion $\mathcal{L} = 3.35 \times 10^5$ J kg^{-1}, and the switch-over time from melting to freezing is $t_o = 100$a. The longitudinal spreading rate $\dot{e}$ is taken to be an exponential function of time, as is suggested by finite-element models of a Hudson Strait surge:

$$\dot{e} = \dot{e}_o e^{\frac{-t}{\tau}} \tag{8.6}$$

where $\dot{e}_o = \frac{1}{100}$ a^{-1}. With these definitions, the equation of evolution for the ice thickness h is

$$\frac{\partial h}{\partial t} = \dot{b} - \dot{e}h \tag{8.7}$$

8.1.1 Nondimensional form of the governing equations

To simplify the problem, we adopt dimensionless variables according to the following prescriptions:

$$\begin{aligned} t &\rightarrow \frac{[L]}{[U]}t \\ T &\rightarrow [\Theta]T \\ w &\rightarrow \frac{[Z][U]}{[L]}w \\ z &\rightarrow [Z]z \\ h &\rightarrow [Z]h \\ \dot{b} &\rightarrow \frac{[Z][U]}{[L]}\dot{b} \\ \dot{e} &\rightarrow \frac{[U]}{[L]}\dot{e} \end{aligned} \tag{8.8}$$

where the scales are defined by

$$\begin{aligned} [L] &= 10^5 \text{ m} \\ [U] &= 50 \times 10^3 \text{ m a}^{-1} \\ [Z] &= 2500 \text{ m} \\ [\Theta] &= 30 \text{ C} \end{aligned} \tag{8.9}$$

Of course, the units of time are converted from years to seconds throughout. The above choice of scales is designed to represent the idea that changes to the ice column are primarily determined by the flow of the ice stream rather than by other thermal effects.

8.1.2 Stretched vertical coordinate

We further simplify the problem by adopting a vertical coordinate that allows the ice column to have fixed end points through time: $\zeta = \frac{z}{h}$. With this coordinate, and the dimensionless variables defined above, the governing equations to the problem become:

$$\begin{aligned} T_t + \frac{\dot{b}(1-\zeta)}{h} T_\zeta &= \frac{1}{\epsilon h^2} T_{\zeta\zeta} \\ T(\zeta = 1, t) &= -1 \\ T(\zeta = 0, t) &= 0 \\ T(\zeta, t = 0) &= -\zeta \end{aligned} \tag{8.10}$$

with

$$\begin{aligned} h_t &= \dot{b} - \dot{e}h \\ \dot{e} &= \dot{e}_o e^{\frac{-t}{\tau}} \end{aligned} \tag{8.11}$$

$$\dot{b} = \begin{cases} \dot{b_o} e^{\frac{-t}{\tau}} & \text{if } 0 \leq t < t_o \frac{[U]}{[L]} \\ \left(\frac{k[\theta][L]}{[Z]^2 \rho \mathcal{L}[U]} \right) \frac{-1}{h} T_\zeta|_{\zeta=0} & \text{if } t \geq t_o \frac{[U]}{[L]} \end{cases} \tag{8.12}$$

and where the Peclet number (measuring the importance of advection relative to diffusion) ϵ is given by

$$\epsilon = \frac{[Z]^2[U]}{[L]\kappa} \tag{8.13}$$

In the above equations, I have chosen to denote partial differentiation by using a subscript t or ζ.

A crucial point to make at this stage is that $\epsilon = 7.07 \times 10^4 >> 1$. We will learn later how to take advantage of this fact to simplify the computation of the temperature-depth profile by eliminating the diffusion term in all but the most crucial parts of the ice column (*i.e.,* by making a boundary-layer approximation).

8.2 Explicit Solution

We shall begin our solution of the problem by first examining a simpler problem: one in which the advection term of Eqn. (8.42) is absent and in which $h = 1$ is kept constant through time. In other words, we take $\dot{b} = 0$. With this simplification, we will learn about explicit and implicit methods for solving diffusive problems. We shall learn about numerical stability and about the effects of diffusion on temperature perturbations in the ice column's initial temperature profile. We will postpone for now the effort to actually predict $T(\zeta, t)$ for the Hudson Strait ice stream.

To solve the heat diffusion equation (Eqn. 8.42 with $\dot{b} = 0$), we create a grid (linear sequence) of finite-difference grid points with a regular spacing on $\zeta \in [0, 1]$. An irregular spacing is possible, and perhaps advantageous in many ways, but will not be investigated here. The location of the grid points is denoted by

$$\zeta_k = (k-1)\Delta\zeta \tag{8.14}$$

with

$$\Delta\zeta = \frac{1}{k_{max} - 1} \tag{8.15}$$

where k_{max} is the number of grid points, and ζ_k is the location of the kth grid point. The temperature at each grid point is denoted by T_k, thus derivatives of T with respect to ζ can be formulated in the following way:

$$T_\zeta|_{\zeta_k + \frac{\Delta\zeta}{2}} \rightarrow \frac{T_{k+1} - T_k}{\Delta\zeta} \tag{8.16}$$

$$T_\zeta|_{\zeta_k - \frac{\Delta\zeta}{2}} \rightarrow \frac{T_k - T_{k-1}}{\Delta\zeta} \tag{8.17}$$

$$T_\zeta|_{\zeta_k} \rightarrow \frac{T_{k+1} - T_{k-1}}{2\Delta\zeta} \tag{8.18}$$

and,

$$T_{\zeta\zeta}|_{\zeta_k} \rightarrow \frac{T_{k+1} + T_{k-1} - 2T_k}{\Delta\zeta^2} \tag{8.19}$$

The diffusion equation is also time dependent. We thus break time up into discrete timesteps as we broke space into discrete grid points:

$$t_n = n\Delta t \tag{8.20}$$

with

$$T_k^n = T_k(t_n) \tag{8.21}$$

The time derivative of T is thus

$$T_t|_{t=t_n+\frac{\Delta t}{2}} \rightarrow \frac{T_k^{n+1} - T_k^n}{\Delta t} \tag{8.22}$$

8.2.1 Part A. The CFL stability criterion

The finite difference version of the pure diffusion equation ($\dot{b} = 0$) can be written in several ways depending on where the $T_{\zeta\zeta}$ term is evaluated in time. If it is evaluated at the nth time step, the finite-difference scheme is said to be explicit:

explicit

$$\begin{aligned}
\frac{T_k^{n+1}}{\Delta t} - \frac{T_k^n}{\Delta t} &= \frac{\epsilon^{-1}}{\Delta\zeta^2}\left(T_{k+1}^n + T_{k-1}^n - 2T_k^n\right) \qquad k = 2, \ldots, (k_{max} - 1) \\
T_1^{n+1} &= -1 \\
T_o^{n+1} &= 0
\end{aligned} \tag{8.23}$$

If $T_{\zeta\zeta}$ is evaluated at the $n+1$ time step, the finite difference scheme is said to be implicit:

implicit

$$\begin{aligned}
\frac{T_k^{n+1}}{\Delta t} - \frac{T_k^n}{\Delta t} &= \frac{\epsilon^{-1}}{\Delta\zeta^2}\left(T_{k+1}^{n+1} + T_{k-1}^{n+1} - 2T_k^{n+1}\right) \qquad k = 2, \ldots, (k_{max} - 1) \\
T_1^{n+1} &= -1 \\
T_o^{n+1} &= 0
\end{aligned} \tag{8.24}$$

The $T_{\zeta\zeta}$ term can be evaluated at an arbitrary time between the nth and the $n+1$ time step:

variable

$$\begin{aligned}
\frac{T_k^{n+1}}{\Delta t} - \frac{T_k^n}{\Delta t} &= \frac{\epsilon^{-1}}{\Delta\zeta^2}\left[\alpha\left(T_{k+1}^n + T_{k-1}^n - 2T_k^n\right) + (1-\alpha)\left(T_{k+1}^{n+1} + T_{k-1}^{n+1} - 2T_k^{n+1}\right)\right] \qquad k = \\
T_1^{n+1} &= -1 \\
T_o^{n+1} &= 0
\end{aligned} \tag{8.25}$$

where $\alpha \in [0, 1]$ is the mixing parameter.

When $\alpha = 1/2$, the scheme is called Crank-Nicholson, and is the most accurate of the three schemes outlined above. The implicit scheme is generally the most stable scheme (stable in the sense that it avoids unbounded growth of grid point oscillations), and the explicit scheme is generally the fastest to execute on the computer because it does not involve matrix factorization.

Let's begin with the explicit scheme, which is written with all knowns on the right-hand side in the following way:

$$T_k^{n+1} = T_k^n + \left[\frac{\Delta t}{\epsilon\Delta\zeta^2}\right]\left(T_{k+1}^n + T_{k-1}^n - 2T_k^n\right) \tag{8.26}$$

This equation is sometimes referred to as an iterated map. Let's define the quantity $\left[\frac{\Delta t}{\epsilon \Delta \zeta^2}\right]$ as the Courant number C after the famous applied mathematician who wrote the famous text on mathematical physics with coauthor Hilbert.

Exercise: explicit solution with Scripts 1-3

Create several MATLAB scripts to determine the evolution of T for a variety of C-values (select values ranging from 0.1 to 10) in the semi-infinite time interval $t \to \infty$, and with an initial condition

$$T_k^1 = \begin{cases} -\zeta_k & \text{if } k \neq \frac{k_{max}}{2} \\ \\ -2 & \text{if } k = \frac{k_{max}}{2} \end{cases} \tag{8.27}$$

To help get this exercise started, I have created MATLAB Scripts 1 - 3 which are included on diskette. Script 1 is a statement of the scales used in nondimensionalization, and Script 2 is a "dumb person's" version of the explicit scheme. Script 3 presents an improved version of Script 2 which makes use of the elegant vectorization capability of MATLAB . Notice in this exercise how the solution shifts from being stable to being unstable when a critical value of C is exceeded. What is this critical value? Can you show analytically why this critical value arises?

Exercise: implicit solution with Script 9

Perform the same analysis of the evolution of T as a function of C over a range of C-values using an implicit scheme. Script 9 is provided as a guide. Are there stability problems with the implicit scheme that arise at certain threshold values of C?

A crucial convenience in the set-up of an implicit time-stepping solution is it's matrix formulation:

$$\mathbf{A}\mathbf{T}^{n+1} = \mathbf{T}^n \tag{8.28}$$

where

$$\mathbf{T}^m = \begin{pmatrix} T_1^m \\ T_2^m \\ \vdots \\ T_{k_{max}-1}^m \\ T_{k_{max}}^m \end{pmatrix} \tag{8.29}$$

$m = n+1, n$, and

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & & & \\ \ldots & -C & 1+2C & -C & \ldots \\ & & & 0 & 1 \end{pmatrix} \tag{8.30}$$

The solution of Eqn. (8.28) involves either Cholesky factorization (for symmetric, positive definite $\mathbf{A}$, or LU factorization (if $\mathbf{A}$ is not symmetric or positive definite), followed by back-substitution. These steps can involve a large number of floating point operations and memory storage. The tri-diagonality of $\mathbf{A}$ provides a means to use sparse-matrix algebra (such as is implemented in MATLAB) to reduce the CPU and memory costs of the implicit scheme. (An illustration of this: For $k_{max} = 100$, it took 5,313,000 flops to LU decompose $\mathbf{A}$ using full-matrix algebra, but only 198,404 flops to do it with MATLAB 's sparse-matrix algebra.) In the case of this particular exercise, the matrix $\mathbf{A}$ does not change with time (this is not true when $\dot{b}$ is a function of t). In this circumstance $\mathbf{A}$ need only be factored once, the factors saved, and back-substitution only is required to perform the time stepping. The implicit scheme thus becomes CPU cost efficient relative to the explicit scheme.

8.2.2 Part B. Upwind differencing.

In the previous section, we investigated the merits of explicit and implicit time-stepping schemes for dealing with the diffusion-only part of the heat equation. Next we shall investigate the merits of centered, downwind, and

upwind differencing in dealing with the advection-only part of the heat equation.

For this investigation, we shall take $\dot{b} < 0$ constant in time and set $\epsilon \to \infty$ to eliminate the diffusion term from Eqn. (8.42). As before, we shall take $h = 1$ as constant in time. The finite-difference forms of the centered, downwind and upwind versions of the (explicit time-stepping) advection-only heat equation, respectively, are:

$$\begin{aligned} T_k^{n+1} &= T_k^n - B(1-\zeta_k)\left(T_{k+1}^n - T_{k-1}^n\right) \\ T_k^{n+1} &= T_k^n - 2B(1-\zeta_k)\left(T_k^n - T_{k-1}^n\right) \\ T_k^{n+1} &= T_k^n - 2B(1-\zeta_k)\left(T_{k+1}^n - T_k^n\right) \end{aligned} \tag{8.31}$$

where $B = \dfrac{\Delta t \dot{b}}{2h\Delta\zeta}$ is the Burger's number. Note that our choice of $\dot{b} < 0$ (basal melting) determines the "windedness" of the advection operator. Upwind is the direction toward greater ζ (toward the top of the ice column) because basal melting causes material ice reference points to move downward. Also observe, for future reference, that

$$B = C\left[\frac{\dot{b}\epsilon\Delta\zeta}{2h}\right] \tag{8.32}$$

The order of the differential equation has changed from 2 to 1 (second-order to first-order) in taking $\epsilon \to \infty$, this necessitates an examination of the boundary conditions. The upper (surface) boundary at $\zeta = 1$ for $\dot{b} < 0$ is an "inflow" boundary, and the lower (basal) boundary is an "outflow" boundary. The upper boundary actually has zero vertical velocity because snow accumulation is not specified; but this boundary is treated in the same manner as an inflow boundary. Only boundary conditions at inflow boundaries are possible to specify, thus we require $T_{k_{max}} = -1$ at $\zeta = 1$. At $\zeta = 0$ we do not specify a boundary condition (we are forbidden to do so by the first-order nature of the idealized governing equation). The upwind finite-differencing scheme will work *as is* at the basal boundary, but the centered and downwind schemes won't because the temperature at gridpoint 0 ($\zeta_0 = -\Delta\zeta$)

does not exist. For simplicity, we shall use the upwind scheme to predict the temperature at the base in both centered and downwind schemes:

$$T_1^{n+1} = -2B\frac{T_2^n - T_1^n}{\Delta\zeta} \tag{8.33}$$

Exercise. Explicit solutions

Using vectorized MATLAB algebra, determine the evolution of T for a variety of B-values and each of the above three schemes in the semi-infinite time interval $t \to \infty$, and with an initial condition

$$T_k^1 = \begin{cases} -\zeta_k & \text{if } k \neq \frac{k_{max}}{2} \\ -1 & \text{if } k = \frac{k_{max}}{2} \end{cases} \tag{8.34}$$

To aid you in your task, I have created Scripts 4-6. Notice the improvement attained by use of the upwinding scheme. Beware, however, when using upwind differencing. Upwind differencing changes the finite-difference operator to one which formally includes diffusion with a diffusivity of $\frac{\Delta\zeta}{2}$:

$$\frac{T_{k+1} - T_k}{\Delta\zeta} = \frac{T_{k+1} - T_{k-1}}{2\Delta\zeta} + \frac{\Delta\zeta}{2}\frac{T_{k+1} + T_{k-1} - 2T_k}{\Delta\zeta^2} \tag{8.35}$$

In other words, the upwind finite-differencing scheme is *exactly* equivalent to the sum of a centered finite-differencing scheme to which has been added a diffusion term with diffusivity that depends on the grid spacing $\Delta\zeta$.

Exercise. Implicit solutions

Perform the same exercise as that described above, but with implicit time-stepping. In other words, solve

$$\mathbf{A}\mathbf{T}^{n+1} = \mathbf{T}^n \tag{8.36}$$

where,

$$\mathbf{A} = \begin{pmatrix} 1-2B & 2B & & & \\ \dots & -B(1-\zeta_k) & 1 & B(1-\zeta_k) & \dots \\ & & & 0 & 1 \end{pmatrix} \tag{8.37}$$

for the centered scheme,

$$\mathbf{A} = \begin{pmatrix} 1-2B & 2B & & & \\ \dots & -2B(1-\zeta_k) & 1+2B(1-\zeta_k) & 0 & \dots \\ & & & 0 & 1 \end{pmatrix} \tag{8.38}$$

for the downwind scheme, and

$$\mathbf{A} = \begin{pmatrix} 1-2B & 2B & & & \\ \dots & 0 & 1-2B(1-\zeta_k) & 2B(1-\zeta_k) & \dots \\ & & & 0 & 1 \end{pmatrix} \tag{8.39}$$

for the upwind scheme.

Pay particular attention to whether adoption of the implicit scheme permits any advantages that were unavailable in any or all of the explicit schemes explored in the previous exercise. Also note that the matrix $\mathbf{A}$ is no longer symmetric and positive definite. The efficient Cholesky factorization routine must, in this circumstance, be replaced by the LU factorization scheme.

Scripts 7 and 8 are provided to help you solve this exercise.

8.3 Solution of the Hudson Strait Ice Column Thermodynamics Problem

Having explored the methods for dealing with advective-diffusion (heat type) equations above, we now turn to the solution of the problem described in section (8.1). We will solve this problem with the *best* of the above explored schemes (*i.e.,* upwind differencing of the advection term, implicit timestepping, with sparse matrix algebra). So, the MATLAB script we need to accomplish this task must solve the equation:

$$\mathbf{A}\mathbf{T}^{n+1} = \mathbf{T}^n \tag{8.40}$$

for time steps $n = 1, \ldots, N$ where N is the number of time steps that must be taken to cover 200-years of thermal evolution. During this evolution, as stated in section (8.1), the basal freezing rate shifts from melting to freezing at $t = 100$ a. In addition to computing $\mathbf{T}$, we shall keep track of the thickness d of debris-bearing regelation ice (ice that has frozen onto the bottom of the ice column during the freeze phase of its thermal evolution) using the equation:

$$d_t = \begin{cases} \dot{b} - \dot{e}d & \text{if } \dot{b} > 0 \text{ or } d > 0 \\ 0 & \text{if } d = 0 \text{ and } \dot{b} \leq 0 \end{cases} \tag{8.41}$$

Script 15, included on diskette, is the MATLAB script I created to simulate the thermal behavior of the Hudson Strait ice column. The evolution of h, $\dot{b}$, d and T as a function of t is displayed in Figs. (8.3) and (8.2). Note that the evolution of h is accounted for in the construction of $\mathbf{A}$.

8.4 Asymptotic Analysis

Having completed the main task of this chapter, computing the thermal evolution of a Hudson Strait ice column, let's now investigate whether any portions of the computing task could have been done more easily (less computation) by first thinking through the problem more carefully. The problem in dimensionless form has a small parameter, ϵ^{-1}. Let's see if we can exploit this fact to simplify the problem.

Zero-order problem

To lowest order in ϵ^{-1}, the ice-column heat transfer problem has no diffusion term:

$$\begin{aligned} T_t^{[0]} + \frac{\dot{b}(1-\zeta)}{h} T_\zeta^{[0]} &= 0 \\ T^{[0]}(\zeta = 1, t) &= -1 \\ &\text{or} \end{aligned}$$

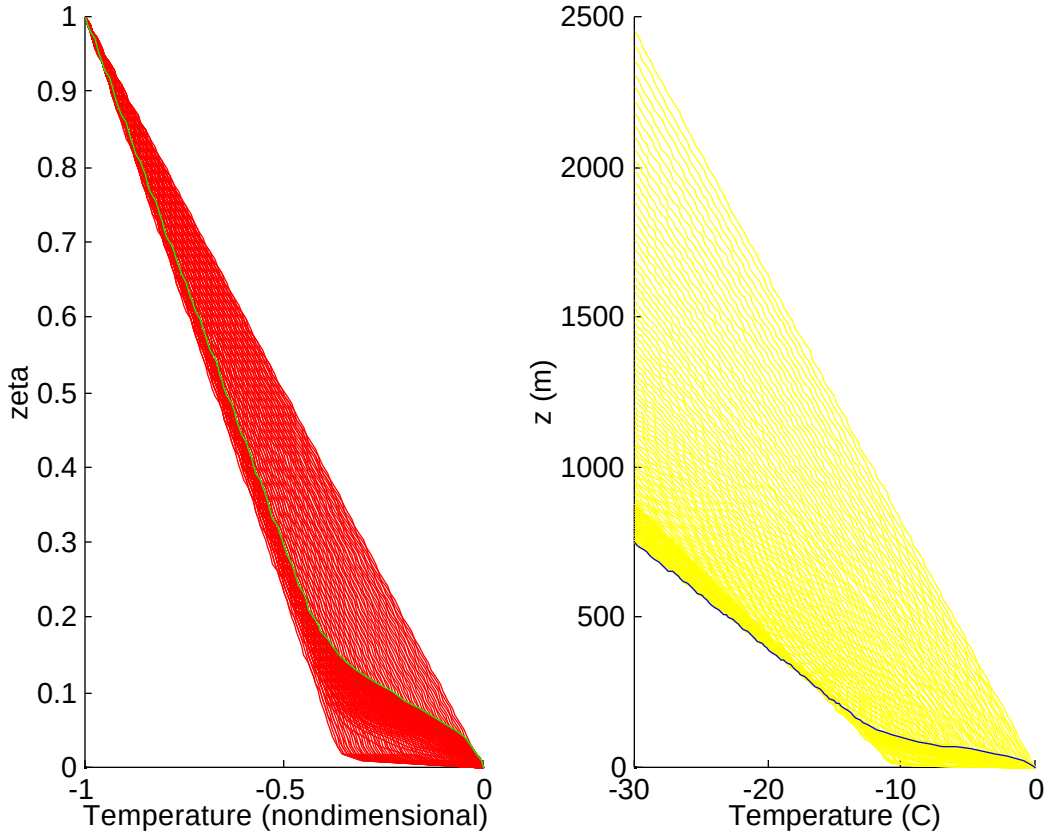

Figure 8.2: Thermal evolution of the Hudson-Strait ice column as function of ζ and z.

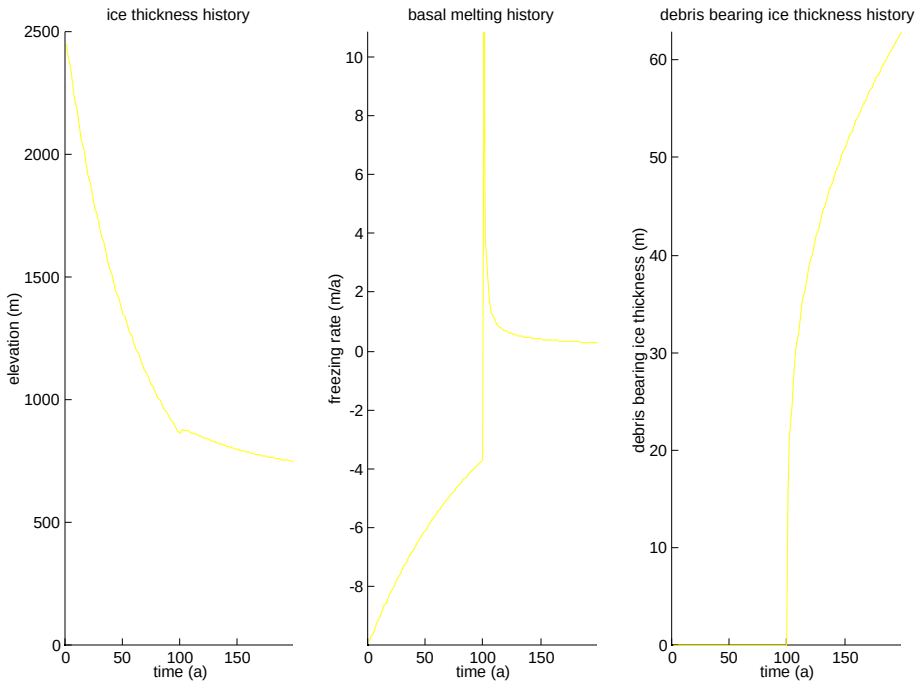

Figure 8.3: Evolution of h, $\dot{b}$ and d as a function of $t \in [0, 200a]$.

$$
\begin{aligned}
T^{[0]}(\zeta = 0, t) &= 0 \\
&\text{and} \\
T^{[0]}(\zeta, t = 0) &= \mathcal{T}(\zeta) = -\zeta
\end{aligned}
\tag{8.42}
$$

This heat transfer problem is first-order in both ζ and t, thus we can anticipate being able to satisfy only one of the boundary conditions with $T^{[0]}$ and relying on a boundary layer solution $\theta(\zeta, t)$ to satisfy the other. The boundary condition we will satisfy during the period of evolution prior to the basal melting to basal freezing transition, *i.e.,* $t < t_o$, is the one at the surface. The bottom boundary during this period of initial melting is an outflow boundary, so the constraint $T^{[0]}(\zeta = 0) = 0$ is impossible to satisfy without diffusion.

Zero-order solution

The solution to the above problem is readily obtained via the method of characteristics. Essentially, the temperature of each material ice parcel is unchanged through the evolution of the ice column, because the time scale of the evolution is short compared to the time scale for thermal diffusion. The temperature-depth profile of the ice column is thus simply the *initial* temperature profile of the ice column over the portion of the initial ice column that hasn't yet melted off:

$$
T^{[0]}(\zeta, t) = \mathcal{T}(\zeta_b + (1 - \zeta_b)\zeta) \tag{8.43}
$$

where $\zeta_b(t)$ is the value of $\zeta \in [0, 1]$ that the ice parcel at the ice/bed interface at $t > 0$ had at the initial time $t = 0$. This variable, $\zeta_b(t)$, is determined by the following evolution equation which tracks the location of the ice/bed interface as it melts its way up through the ice column:

$$
\dot{\zeta}_b = \frac{-\dot{b}h(t = 0)}{h_s(t)} = \frac{-\dot{b}}{h_s(t)} \tag{8.44}
$$

since $h(t = 0) = 1$, and where h_s is the (dimensionless) ice thickness that would exist if $\dot{b} = 0$:

$$
\dot{h}_s = -\dot{e}h_s \tag{8.45}
$$

The purpose of the $\frac{h(t=0)}{h_s(t)}$ in the above equations is to normalize $\dot{b}$ according to how melting would effect the initial ice column *after* it has thinned by strain for a period of time.

Boundary layer

The above zero-order solution meets the initial condition and the boundary condition at $\zeta = 1$. To meet the boundary condition at $\zeta = 0$, a boundary layer solution must be added:

$$T(\zeta, t) \approx T^{[0]}(\zeta, t) + \theta(\eta, t) \tag{8.46}$$

where the boundary-layer coordinate η is defined by

$$\eta = \epsilon^{\alpha} \zeta \tag{8.47}$$

and α is an, as yet, unchosen parameter. Substitution of θ into the heat equation, and use of the relations

$$\begin{aligned} \frac{\partial}{\partial \zeta} &\rightarrow \epsilon^{\alpha} \frac{\partial}{\partial \eta} \\ \frac{\partial^2}{\partial \zeta^2} &\rightarrow \epsilon^{2\alpha} \frac{\partial^2}{\partial \eta^2} \end{aligned} \tag{8.48}$$

gives

$$\theta_t + \frac{\dot{b}(1 - \epsilon^{-\alpha})}{h} \epsilon^{\alpha} \theta_{\eta} = \epsilon^{2\alpha - 1} \theta_{\eta\eta} \tag{8.49}$$

If we choose $\alpha = 1$, and we retain only the lowest order terms, the above equation simplifies to

$$\frac{\dot{b}}{h} \theta_{\eta} = \theta_{\eta\eta} \tag{8.50}$$

The domain over which Eqn. (8.49) is solved is $0 \leq \eta \rightarrow \infty$ because $\eta \rightarrow \infty$ as $\zeta \rightarrow 1$. The boundary conditions on Eqn. (8.49) are

$$\begin{aligned} \theta(\eta = 0, t) &= -\mathcal{T}(\zeta_b(t)) \\ \theta(\eta \rightarrow \infty, t) &= 0 \end{aligned} \tag{8.51}$$

An initial condition is not needed because Eqn. (8.49) is not time dependent.

Boundary layer solution

The solution to the boundary layer problem is readily found by integrating Eqn. (8.49) twice with respect to η and applying the boundary conditions to resolve undetermined integration constants:

$$\theta(\eta, t) = -\mathcal{T}(\zeta_b(t))e^{\frac{-\dot{b}\eta}{h}} \tag{8.52}$$

Full solution

The full solution is

$$T(\zeta, t) \approx \mathcal{T}(\zeta_b + (1 - \zeta_b)\zeta) - \mathcal{T}(\zeta_b(t))e^{\frac{-\dot{b}\epsilon\zeta}{h}} \tag{8.53}$$

What is interesting and possibly useful about this solution, and about the asymptotic approach itself, is the fact that the above solution can be arrived at by solving only two ordinary differential equations for the unknown scalar ζ_b:

$$\dot{\zeta}_b = \frac{-\dot{b}h(t=0)}{h_s(t)} = \frac{-\dot{b}}{h_s(t)} \tag{8.54}$$

and

$$\dot{h}_s = -\dot{e}h_s \tag{8.55}$$

Solving the above two ordinary differential equations for the scalars ζ_b and h_s requires a great deal less computational effort than required to solve the heat equation with a finite-difference method for the unknown function $T(\zeta, t)$.

8.4.1 Exercise: Comparison between finite-difference and asymptotic methods

Compute and compare $T(\zeta, t = t_o)$ for the Hudson Strait ice column using two methods (one finite difference and one asymptotic). Which method is most accurate? Which method is most cost efficient?

Scripts 13 and 14 were used to perform this exercise in Chicago. The finite-difference method (using sparse matrix algebra, upwind differencing and an implicit time step with $C = 1$) took 19,758 floating-point operations to achieve. In comparison, the asymptotic method required 1,417 floating-point operations to achieve the same result. A comparison of the two solutions is provided in Figure (8.4).

8.4.2 Exercise: Asymptotic solution for $t > t_o$

See if you can develop an asymptotic solution for the period of evolution when the ice column experiences basal freezing.

8.5 Wrap-up

In this chapter, we have investigated a very simple ice-sheet thermodynamic problem which has a relatively speculative application (Heinrich Events). This investigation has offered us the opportunity to learn about several numerical methods, algebraic short cuts and asymptotic analysis.

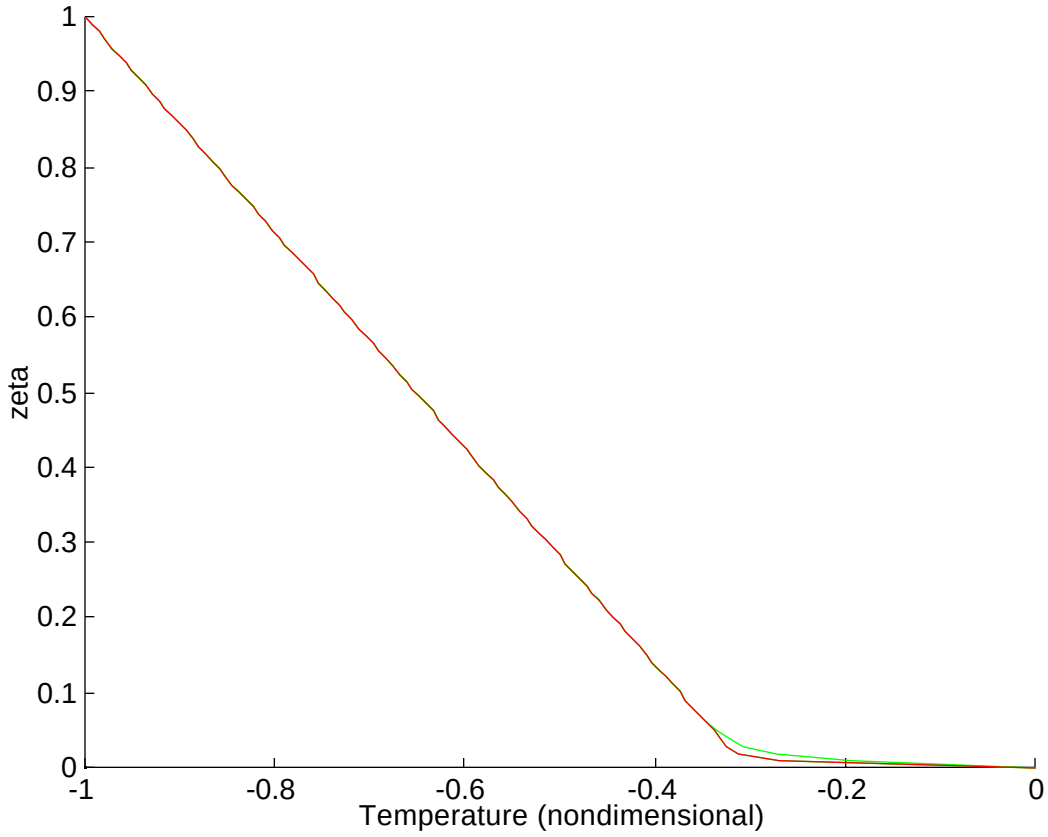

Figure 8.4: Comparison of solutions obtained by finite-difference method and by the asymptotic method.

Chapter 9

Ice Sheet Thermodynamics: 3-D Modelling Techniques

The temperature field is perhaps the single most influential variable on ice-sheet behavior. When ice is warm, it is less viscous; thus, strain heating near an ice-sheet bed can have an important impact on shear deformation. For ice shelves, where the depth-averaged temperature determines the strain rates, basal melting and freezing can have a significant impact on apparent ice-shelf viscosity. Perhaps the single most important influence of the temperature on ice-sheet motion is basal melting. When grounded ice has a melted bed, it is subject to several types of fast-flow processes. The most significant of these involves the combination of sliding and subglacial sediment deformation.

The modelling of ice-sheet temperature has a rich history (perhaps the richest of all modelling histories in glaciology). Early interests were in determining the temperature-depth profile of various drill sites around Antarctica and Greenland (e.g., Robin, 1954). In the late 1970's and through the 1980's, several modelling milestones were passed with the construction of fully coupled dynamic (ice-motion) thermodynamic (ice-heat flow and water content) models to address "whole ice sheet" modelling problems. Examples of these milestone models are Huybrechts (1992) and Greve (1995).

In this chapter, we will construct a dynamic/thermodynamic ice-sheet

model to complete the EISMINT Level 1 intercomparison test (only the fixed margin test will be presented here, others will be left as exercises). This model will be based on the finite-element method. The dynamic (ice flow) component of the model will involve the same physics as that constructed in Chapter 2. Details of the finite-element model, in particular the order of the interpolation of the ice-thickness field within each triangular element, will be different to accomodate computation of the vertical ice velocity. The thermodynamic portion of the model will be new. The most important of the many caveats to be noted in this chapter is that we shall not allow ice to be polythermal (i.e., have both temperate and frozen regions). Analysis of polythermal ice-sheet thermodynamics is significantly more complex than the scope of the present chapter allows. For a tutorial on polythermal ice-sheet modelling, consult Greve (1995).

9.1 Ice-Sheet Flow Equations

The equations which govern the horizontal and vertical velocity fields in a simple ice sheet (following Greve, 1995) are:

$$u(z) = -\rho g(s-b)C(T)\frac{\partial s}{\partial x} - 2\rho g\frac{\partial s}{\partial x}I(z) \tag{9.1}$$

$$v(z) = -\rho g(s-b)C(T)\frac{\partial s}{\partial y} - 2\rho g\frac{\partial s}{\partial y}I(z) \tag{9.2}$$

$$w(z) = -\int_b^z \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) dz' + \frac{\partial b}{\partial t} + u(b)\frac{\partial b}{\partial x} + v(b)\frac{\partial b}{\partial y} - \dot{B} \tag{9.3}$$

where x and y are horizontal coordinates, z is the vertical coordinate (positive up), u and v are the x and y velocity components, respectively, w is the vertical velocity, s and b are the surface and basal elevation, respectively, ρ is the ice density (assumed to be constant, 910 kg m^{-3}), g is the acceleration of gravity (9.81 m s^2), and $\dot{B}$ is the basal melting rate (positive for melting, negative for freezing, in meters of ice equivalent per second). The basal

sliding constant $C(T)$ is assumed to be nonzero when the basal temperature $T(b)$ is at the pressure-melting point (Payne, 1995):

$$C(T) = \begin{cases} 5 \times 10^{-3} \text{ m a}^{-1} \text{ Pa}^{-1} & \text{if } T(b) = T_m \\ \\ 0 & \text{if } T(b) < T_m \end{cases} \tag{9.4}$$

The variable $T(x, y, z, t)$ is the temperature, and T_m is the z-dependent melting point,

$$T_m(z) = T_o - \rho g(s - z)\Phi \tag{9.5}$$

where $\Phi = 8.71 \times 10^{-4}$ K Pa^{-1} (Payne, 1995), and $T_o = 273.15$ K. The factor $I(z)$ determines the deformational velocity of the ice flow, and for Glen's flow law (with exponent 3) is defined by

$$I(z) = \int_b^z EA(T^*)(\rho g)^2 \left(\nabla s \cdot \nabla s\right) \left(s - z'\right)^3 dz' \tag{9.6}$$

The rate-enhancement factor E is essentially a fudge factor designed to account for empirically determined inadequacies of Glen's law. The value of E is commonly taken to be 1 for "Holocene ice" and 3 for "glacial-period ice", for example, in modelling studies of the Greenland ice sheet. The creep-rate factor $A(T^*)$ is assumed to have a standard thermodynamic form involving a rate constant factor a and an activation energy Q:

$$A(T^*) = a \exp\left(\frac{-Q}{RT^*}\right) \tag{9.7}$$

where $R = 8.31$ J mol^{-1} K^{-1} is the gas constant, and the flow-law parameters (determined from laboratory studies of polycrystaline ice assumed to be isotropic) are commonly taken to be:

$$a = \begin{cases} 7.23 \times 10^{-12} \text{ s}^{-1} \text{ Pa}^{-3} & \text{if } T^* < 263 \text{ K} \\ \\ 3.47 \times 10^{4} \text{ s}^{-1} \text{ Pa}^{-3} & \text{if } T^* \geq 263 \text{ K} \end{cases} \tag{9.8}$$

and

$$Q = \begin{cases} 6.0 \times 10^{4} \text{ J}^{-1} \text{ mol}^{-1} & \text{if } T^* < 263 \text{ K} \\ \\ 13.9 \times 10^{4} \text{ J}^{-1} \text{ mol}^{-1} & \text{if } T^* \geq 263 \text{ K} \end{cases} \tag{9.9}$$

The temperature in the creep-rate factor formula $T^* = T - T_m + T_o$ is the Homologous temperature (temperature relative to the pressure melting point) in degrees Kelvin (note the addition of T_o).

Justification of Equations (9.1) and (9.2) using the standard shallow-ice approximation is presented in Hutter (198?). Longitudinal stresses are eliminated from the diagnostic equations for u and v when the shallow-ice approximation applies. Note that this approximation produces a set of governing equations for horizontal velocity that is quite different from those which govern the same variables on an ice shelf (see Chapter 4).

Mass balance of the ice sheet is expressed by the following equation for ice thickness h,

$$\frac{\partial h}{\partial t} = -\nabla \cdot (\mathbf{q}) + \dot{A} - \dot{B} \tag{9.10}$$

where $\mathbf{q} = D\nabla s$ is the vector-valued horizontal mass flux integrated over the ice column, $\dot{B}$ is the basal melting rate (positive for melting, negative for freezing), and $\dot{A}$ is the surface snow accumulation rate expressed in meters of ice equivalent per year. Note that ice densification effects are disregarded. A common practice is to substitute Equations (9.1) and (9.2) into Equation (9.10) giving,

$$\frac{\partial h}{\partial t} = \nabla \cdot (D\nabla s) + \dot{A} - \dot{B} \tag{9.11}$$

where the effective diffusivity D is defined by,

$$D = \int_b^s 2\rho g I(z')dz' + \rho g(s-b)^2 C \tag{9.12}$$

Kinematic boundary conditions on the free surface $z = s$ and basal surface $z = b$ are recorded for use elsewhere:

$$\frac{\partial s}{\partial t} + u(s)\frac{\partial s}{\partial x} + v(s)\frac{\partial s}{\partial y} = w(s) + \dot{A} \tag{9.13}$$

and,

$$u(b)\frac{\partial b}{\partial x} + v(b)\frac{\partial b}{\partial y} = w(b) + \dot{B} \tag{9.14}$$

where we have made the assumption that $\frac{\partial b}{\partial t} = 0$ (no isostatic or bed erosion effects are included).

9.2 Ice-Sheet and Bedrock Heat-Flow Equations

Heat flow continuity in both ice and underlying bedrock is expressed using the standard advective/diffusive equation with variable heat-flow parameters:

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} + w\frac{\partial T}{\partial z} = \frac{1}{\rho c}\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) + \frac{W}{\rho c} \tag{9.15}$$

for x and y within the footprint of the ice sheet, and $b < z < s$; and

$$\frac{\partial T}{\partial t} = \frac{1}{\rho_r c_r}\frac{\partial}{\partial z}\left(k_r\frac{\partial T}{\partial z}\right) \tag{9.16}$$

for x and y below the footprint of the ice sheet (including below portions of floating ice), and $z_r \leq z \leq b$, and where z_r is a level at a fixed elevation below the ice/bedrock interface (or seabed) where the geothermal flux gradient is applied as a boundary condition (see below). The above equations reflect the standard assumption that horizontal heat conduction is neglegible in comparison with horizontal advection, veritcal advection and conduction, and viscous heat dissipation. The horizontal and vertical velocities referred to in Equations (9.15) and (9.16) are described by Equations (9.1) - (9.3).

Observe that the vertical gradients of k and k_r are acknowledged in the form of the thermal diffusion in Equations (9.15) and (9.16). The horizontal gradients of k and k_r, and all the spatial gradients of ρc and $\rho_r c_r$ are disregarded. This constitutes a simplification that is used widely in ice-sheet thermodynamic studies. It's justification is not provided here (and indeed, there may be an inconsistency to be corrected in future work). See Greve [1995] for further discussion of this matter.

Parameters appearing in Equations (9.15) and (9.16) include the thermal heat capacities for ice and rock, c and c_r, respectively:

$$c = 2115.3 + 7.79293(T - T_o) \tag{9.17}$$

$$c_r = 1000 \tag{9.18}$$

in units of J kg^{-1} K^{-1} (Huybrechts, 1993), and the thermal conductivities for ice and rock, k and k_r, respectively:

$$k = 3.101 \times 10^8 \exp\left(-0.0057T\right) \tag{9.19}$$

$$k_r = 3.3 \tag{9.20}$$

in units of W m K^{-1} (Huybrechts, 1993). The density of rock ρ_r is taken to be that of typical sedimentary rock of interest below the central portions of the Laurentide and West Antarctic ice sheets, 2700 kg m^{-3}. The term W in Eqn. (9.15) is the viscous heating term formally defined by

$$W = \sum_i \sum_j \dot{e}_{ij} T'_{ij} \tag{9.21}$$

where $\dot{e}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ is the strain rate, and T'_{ij} is the deviatoric stress. According to the common assumptions associated with grounded ice-sheet stress balance (*i.e.,* neglecting longitudinal stress), W can be written as:

$$W(z) = (\rho g)^4 \, 2 \left(\nabla s \cdot \nabla s\right)^2 (s - z)^4 \, EA(T^*) \tag{9.22}$$

Boundary conditions are necessary at the top $z = s$ and bottom $z = z_r$ of the ice/rock column. At the top, a surface temperature (depending on atmospheric conditions) is specified

$$T(s) = T_s \tag{9.23}$$

At the bottom $z = z_r$, well below the ice/bedrock interface, a geothermal temperature gradient is specified

$$\frac{\partial T}{\partial z} = -\frac{\Gamma}{k_r}. \tag{9.24}$$

The geothermal flux Γ is typically taken to be one geophysical heat flow unit (0.42 W m^{-2}). The value of z_r is typically chosen to be a fixed distance below $z = b$. A useful consideration for choosing this distance $(b - z_r)$ is the e-folding penetration depth λ for temperature oscillations forced at the ice/bedrock interface of frequency ω (Carlsaw and Jaeger, 1954):

$$\lambda = \left(\frac{2\kappa_r}{\omega}\right)^{\frac{1}{2}} \tag{9.25}$$

where the thermal diffusivity of bedrock is defined by $\kappa_r = \frac{k_r}{\rho_r c_r}$. For Heinrich-event frequency oscillations (periodicity of about 10,000 years), the penetration depth λ is 375 m.

Unless there are inflow boundaries around the horizontal edges of the ice sheet, thermal boundary conditions are not required around the edges of the ice sheet. This is because the horizontal mode of heat transfer in an ice sheet is advection, a process that demands boundary conditions only where ice flow into the domain.

A material constraint on the ice-sheet thermodynamics is that ice never warm above the presssure-melting point. When layers of finite thickness reach the pressure-melting point, the ice sheet is said to become "polythermal" (Hutter, 1982; Hutter, Blatter and Funk, 1988). A review of polythermal ice-sheet modelling is provided by Greve (1995). Polythermal conditions typically occur as a result of viscous dissipation, which heats the ice column internally and permits the locus of a temperature maximum (the pressure-melting point, by definition) to move away from the boundary (a constraint that is otherwised forced by a consideration similar to the maximum principle of solutions of Laplace Equation). When ice becomes temperate (at the pressure melting point), its vertical temperature gradient becomes that dictated by the "Clapyron slope" of water (the change of the melting temperature with pressure). In this circumstance, the vertical heat flux is fixed (near zero), so further heat transfer (via internal heating and flux from non-temperate portions of the ice sheet) acts only to modify the liquid water content of the ice. In the ice-sheet model by Greve (1995), for example, the heat-flow continuity equation in the temperate ice is replaced with a diffusive-type equation for water content, and an internal free surface (called the CTS, cold/temperate ice transition surface) must be monitored through the use of energy flux and mass flux matching conditions.

In the examples described in this chapter, polythermal ice conditions are not treated; thus, we shall make a standard (but not necessarily well justified) assumption that the melting point is achieved only at the ice base. In this simplification, we augment the heat equation developed above with logical conditions determining when the basal ice boundary condition is fixed at the melting point, or when the basal ice temperature (frozen) is allowed to freely

vary according to the combined ice/bedrock heat equation. When the base is at the pressure-melting point, we compute a basal melting rate, $\dot{B}$ (positive for melting, meters of ice equivalent per year), to balance the heat budget at the basal ice interface. We define heat-flux terms H_o and H_i as the outward- and inward-directed heat fluxes to the interface $z = b$, rexpectively, by

$$H_o = -k\frac{\partial T}{\partial z}|_{z=b+} \tag{9.26}$$

and,

$$H_i = -k_r\frac{\partial T}{\partial z}|_{z=b-} + \mathbf{u}(b)\cdot\tau_b + \begin{cases} \frac{\rho\dot{B}}{L_f} & \text{if } \dot{B} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{9.27}$$

where τ_b is the vector-valued basal stress which, in the absence of longitudinal stress, is $-\rho g h \nabla s$, and $\mathbf{u}(b)$ is the basal sliding velocity (which can be nonzero when the bed is melted). The dot product $\mathbf{u}(b)\cdot\tau_b$ is assumed positive. With the above definitions for H_o and H_i, $\dot{B}$ is determined by

$$\dot{B} = \frac{H_i - H_o}{\rho L_f} \tag{9.28}$$

where $L_f = 3.35 \times 10^5$ J kg^{-1} is the latent heat of fusion for pure water. In circumstances, where $\dot{B}$ is nonzero, a basal water layer can develop. When $\dot{B} > 0$ (melting), the presence or absence of basal water does not influence the thermal evolution of the ice base. When $\dot{B} < 0$ (freezing), a basal water layer must be present to supply the ice that is accumulated on the base. In this circumstance, if there is insufficient basal water to supply the required basal freezing, the base of the ice will freeze to the bed and the basal temperature will drop below the pressure melting point.

The above expressions describing the basal melting condition are interlinked; so, it is not simple to determine when a frozen bed should be converted to a melted bed, or a melted bed to a frozen bed. In brief, a frozen bed is melted when the solution of the heat equation dictates that $T(b)$ rise to the pressure-melting point; and a melted bed will freeze when the vertical heat flux into the ice is greater than the heat source into the bed by the combination of (1) conduction from the rock below, (2) heat generation by basal sliding, and (3) latent heat released by freezing of available water stored in the bed.

9.3 Contour-Following Vertical Coordinate

To handle the variable vertical dimension of the ice sheet, a new vertical coordinate ζ is defined such that $\zeta = 1$ at $s(x, y, t)$ and $\zeta = 0$ at $b(x, y, t)$. The relation between ζ and z is

$$\zeta = \frac{z - b}{s - b} \tag{9.29}$$

and

$$z = (s - b)\zeta + b \tag{9.30}$$

In this circumstance, z-derivatives are easily converted to ζ-derivatives:

$$\frac{\partial \cdot}{\partial z} = \frac{1}{(s - b)} \frac{\partial \cdot}{\partial \zeta} \tag{9.31}$$

$$\frac{\partial^2 \cdot}{\partial z^2} = \frac{1}{(s - b)^2} \frac{\partial^2 \cdot}{\partial \zeta^2} \tag{9.32}$$

The conversion of x- y- and t-derivatives evaluated at fixed z to their counterparts evaluated at fixed ζ is somewhat more complicated. Following the considerations described in the caption to Figure (9.1), the derivative of T with respect to x on a surface of constant z is

$$T_x|_{z=\text{constant}} = T_x|_{\zeta=\text{constant}} - \frac{1}{h} T_\zeta \left(\zeta \frac{\partial s}{\partial x} + (1 - \zeta) \frac{\partial b}{\partial x} \right) \tag{9.33}$$

where subscripts x, z and ζ denote partial derivatives of T, and $h = (s - b)$ is the ice thickness. The expressions for T_y and T_t are

$$T_y|_{z=\text{constant}} = T_y|_{\zeta=\text{constant}} - \frac{1}{h} T_\zeta \left(\zeta \frac{\partial s}{\partial y} + (1 - \zeta) \frac{\partial b}{\partial y} \right) \tag{9.34}$$

and,

$$T_t|_{z=\text{constant}} = T_t|_{\zeta=\text{constant}} - \frac{1}{h} T_\zeta \left(\zeta \frac{\partial s}{\partial t} + (1 - \zeta) \frac{\partial b}{\partial t} \right) \tag{9.35}$$

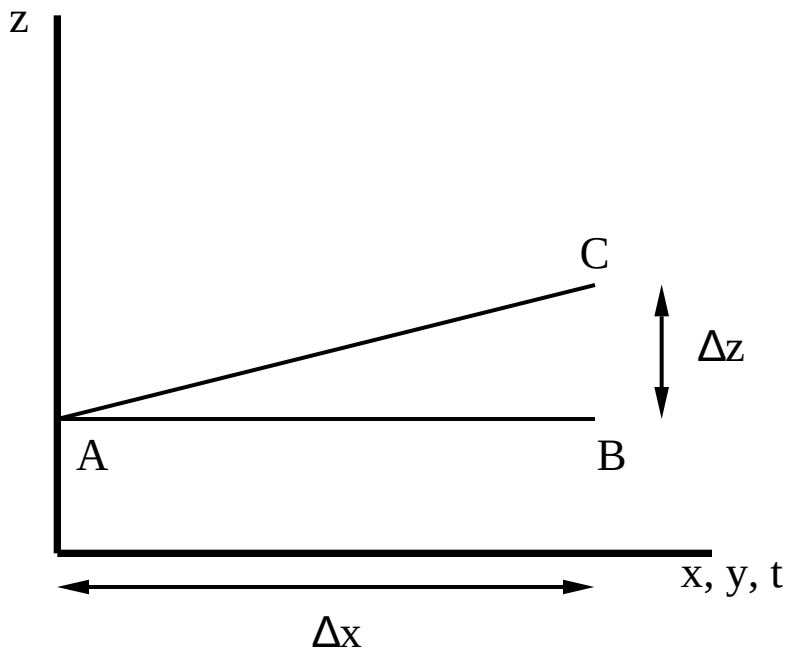

Figure 9.1: Points A and B lie on a constant z surface and are spaced horizontally by Δx. Points A and C lie on a constant ζ surface and are also spaced horizontally by Δx. The vertical distance between points C and B, Δz, is determined by the variation of s and b with x according to the definition of ζ in terms of z. The x-derivative of a function $f(x)$ on a constant z-surface is simply $\lim_{\Delta x \to 0} \frac{f(B)-f(A)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(C)-f(A)}{\Delta x} - \frac{f(C)-f(B)}{\Delta z} \frac{\Delta z}{\Delta x}|_{\zeta=\text{constant}}$.

With the above conversion formulae for T_x, T_y, T_z and T_t, we may rewrite Equation (9.15) as follows:

$$\begin{aligned} &T_t + uT_x + vT_y + \frac{w}{h}T_\zeta \\ -\frac{1}{h}T_\zeta\Big(u\zeta\frac{\partial s}{\partial x} + u(1-\zeta)\frac{\partial b}{\partial x} + v\zeta\frac{\partial s}{\partial y} + v(1-\zeta)\frac{\partial b}{\partial y} &\\ + \zeta\frac{\partial s}{\partial t} + (1-\zeta)\frac{\partial b}{\partial t}\Big) &\\ = \frac{1}{\rho c h^2}\frac{\partial}{\partial\zeta}\left(k\frac{\partial T}{\partial\zeta}\right) + \frac{W}{\rho c} & \end{aligned} \tag{9.36}$$

Observe that

$$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} + v\frac{\partial s}{\partial y} = w(s) + \dot{A} \tag{9.37}$$

and,

$$\frac{\partial b}{\partial t} + u\frac{\partial b}{\partial x} + v\frac{\partial b}{\partial y} = w(b) + \dot{B} \tag{9.38}$$

Substitution of the above equations into Equation (9.36) gives,

$$\begin{aligned} &T_t + uT_x + vT_y \\ +\frac{1}{h}T_\zeta\left(w(z) - \zeta\left(w(s) + \dot{A}\right) - (1-\zeta)\left(w(b) + \dot{B}\right)\right) &\\ = \frac{1}{\rho c h^2}\frac{\partial}{\partial\zeta}\left(k\frac{\partial T}{\partial\zeta}\right) + \frac{W}{\rho c} & \end{aligned} \tag{9.39}$$

This equation may be rewriten as follows:

$$\begin{aligned} &T_t + uT_x + vT_y \\ +\frac{1}{h}T_\zeta\left(w(z) - w(b) - \zeta\left(w(s) - w(b)\right) - \zeta\dot{A} - (1-\zeta)\dot{B}\right) &\\ = \frac{1}{\rho c h^2}\frac{\partial}{\partial\zeta}\left(k\frac{\partial T}{\partial\zeta}\right) + \frac{W}{\rho c} & \end{aligned} \tag{9.40}$$

Observe, however, that

$$w(z) - w(b) = -\int_b^z \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) dz' \tag{9.41}$$

and,

$$w(s) - w(b) = -\int_b^s \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) dz' \tag{9.42}$$

Substitution of Eqns. (9.41) and (9.42) into Eqn. (9.39) gives

$$\begin{aligned} T_t + uT_x + vT_y & \\ +\frac{1}{h}T_\zeta \left(\mathcal{D}(\zeta) - \zeta\dot{A} - (1-\zeta)\dot{B}\right) & \\ = \frac{1}{\rho c h^2}\frac{\partial}{\partial \zeta}\left(k\frac{\partial T}{\partial \zeta}\right) + \frac{W}{\rho c} & \end{aligned} \tag{9.43}$$

where $\mathcal{D}(\zeta)$ is an ice-divergence parameter defined by,

$$\mathcal{D}(\zeta) = h\left(\zeta\int_0^1 \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) d\zeta - \int_0^\zeta \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) d\zeta'\right) \tag{9.44}$$

Equation (9.44) represents the final form of the heat equation for use in grounded ice-sheet modelling. For ice-shelf and ice-stream modelling, where u, v, and therefore $\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$ are independent of z and ζ, $\mathcal{D}(\zeta) = 0$; thus the appropriate form of the heat equation becomes,

$$\begin{aligned} T_t + uT_x + vT_y & \\ -\frac{1}{h}T_\zeta \left(\zeta\dot{A} + (1-\zeta)\dot{B}\right) & \\ = \frac{1}{\rho c h^2}\frac{\partial}{\partial \zeta}\left(k\frac{\partial T}{\partial \zeta}\right) + \frac{W}{\rho c} & \end{aligned} \tag{9.45}$$

The form for use in ice-shelf and ice-stream modelling is considerably simpler than the form for use in ice-sheet modelling (Eqn. 9.44). There are three primary simplifications which apply when u and v are independent of z and ζ:

1. The horizontal advection operators $u\frac{\partial}{\partial x}$ and $v\frac{\partial}{\partial y}$ are the same for every level in the vertical.

2. The vertical velocity $w(\zeta)$ need not be calculated.

3. The ice-divergence parameter $\mathcal{D}(\zeta)$ is 0.

These simplifications make ice-shelf and ice-stream modelling far more simple and computationally efficient than ice-sheet modelling.

9.4 Discretization With High-Order Element Interpolation

According to the diagnostic relations between u, v, w and s for a grounded ice-sheet with "inland ice" type flow (Eqns. 9.1 - 9.3), the computation of the vertical velocity w and the $\mathcal{D}$-term as a function of ζ requires that the second spatial derivatives of s, *i.e.*, $\frac{\partial^2 s}{\partial x^2}$ and $\frac{\partial^2 s}{\partial y^2}$, be evaluated on the computational domain. This requirement necessitates the use of high-order element interpolation. We describe this interpolation by first considering linear interpolation, where first derivatives of s are piecewise constant; but where second derivatives are undefined.

9.4.1 Linear Triangle Elements

For triangular finite elements with linear interpolation, the discretization technique used in previous chapters, s is interpolated by

$$s(x,y) = \sum_{j=1}^{3} s_j L_j(x,y) \tag{9.46}$$

where the interpolation functions L_j, $j = 1, \ldots, 3$, are linear in form, *i.e.*,

$$L_j = \alpha_j x + \beta_j y + \gamma_j \tag{9.47}$$

where, the coefficients of L_j are given by the following expression,

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}^{-1} \tag{9.48}$$

The above expression for the α's, β's and γ's is equivalent to the solution of the nine equations:

$$\alpha_j x_i + \beta_j y_i + \gamma_j = \delta_{ij} \qquad \forall_i \; \forall_j \tag{9.49}$$

where, $\delta_{ij} = 0$ if $i \neq j$ and $\delta_{ij} = 1$ if $i = j$. The first derivatives of s according to Eqn. (9.46) are piecewise constant and vary from element to element,

$$\frac{\partial s}{\partial x} = \sum_{j=1}^{3} s_j \alpha_j \tag{9.50}$$

and,

$$\frac{\partial s}{\partial y} = \sum_{j=1}^{3} s_j \beta_j \tag{9.51}$$

9.4.2 Quadratic Triangular Elements

With linear interpolation, the second derivatives of s on the computational domain are unresolved. Thus, it is not possible to evaluate $w(\zeta)$ or $D(\zeta)$ for use in solving Eqn. (9.44). We overcome this difficulty by adopting a higher order interpolation within a triangular element that has 6 nodes (on its vertices and at the midpoints of its faces, as in Fig. 9.2). In particular, we define

$$s(x, y) = \sum_{j=1}^{6} s_j N_j(x, y) \tag{9.52}$$

where the interpolation functions N_j are quadratic in x and y so that second derivatives are resolved, *i.e.*,

$$N_j(x, y) = \gamma_j + \alpha_j x + \beta_j y + \epsilon_j x^2 + \delta_j y^2 + \phi_j xy \tag{9.53}$$

and where the coefficients are determined by the expression,

$$\begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \phi_4 & \phi_5 & \phi_6 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 \\ \epsilon_1 & \epsilon_2 & \epsilon_3 & \epsilon_4 & \epsilon_5 & \epsilon_6 \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 \end{bmatrix} = \begin{bmatrix} x_1 y_1 & y_1^2 & x_1^2 & x_1 & y_1 & 1 \\ x_2 y_2 & y_2^2 & x_2^2 & x_2 & y_2 & 1 \\ x_3 y_3 & y_3^2 & x_3^2 & x_3 & y_3 & 1 \\ x_4 y_4 & y_4^2 & x_4^2 & x_4 & y_4 & 1 \\ x_5 y_5 & y_5^2 & x_5^2 & x_5 & y_5 & 1 \\ x_6 y_6 & y_6^2 & x_6^2 & x_6 & y_6 & 1 \end{bmatrix}^{-1} \tag{9.54}$$

The above expression for the coefficients in the polynomial form of N_j is equivalent to the solution of the 36 equations:

$$\phi_j x_i y_i + \epsilon_j y_j^2 + \delta_j x_j^2 + \alpha_j x_i + \beta_j y_i + \gamma_j = \delta_{ij} \qquad \forall_i \; \forall_j \tag{9.55}$$

where, $\delta_{ij} = 0$ if $i \neq j$ and $\delta_{ij} = 1$ if $i = j$. An alternative expression for the definition of the quadratic interpolation functions $N_j(x, y)$ given in Eqn. (9.53) is given by Zienkiewicz [1971, p. 119],

$$N_1 = (2L_1 -)L_1, \quad \text{etc.} \tag{9.56}$$

for corner nodes, and

$$N_4 = 4L_1 L_2, \quad \text{etc.} \tag{9.57}$$

for mid-side nodes, where the L_i's are the usual linear interpolation functions given above.

The first derivatives of s according to Eqn. (**??**) are piecewise linear in each element,

$$\frac{\partial s}{\partial x} = \sum_{j=1}^{3} s_j(\phi_j y + 2\delta_j x + \alpha_j) \tag{9.58}$$

and,

$$\frac{\partial s}{\partial y} = \sum_{j=1}^{3} s_j(\phi_j x + 2\epsilon_j y + \beta_j) \tag{9.59}$$

While s is continuous across element boundaries, the first derivatives of s are not; they are disjoint piecewise linear functions. The second derivatives of s are piecewise constant in each element, and are given by

$$\frac{\partial^2 s}{\partial x^2} = \sum_{j=1}^{3} s_j 2\delta_j \tag{9.60}$$

and,

$$\frac{\partial^2 s}{\partial y^2} = \sum_{j=1}^{3} s_j 2\epsilon_j \tag{9.61}$$

The existance of these piecewise constant evaluations of the second derivatives allow the divergence of horizontal velocity and the vertical velocity to be computed from the expressions in Eqns. (9.1) - (9.3). Note that $D(\zeta)$ and $w(\zeta)$ will vary with ζ in each element, but will not vary with x and y because the underlying expression, involving second derivatives of s, is piecewise constant.

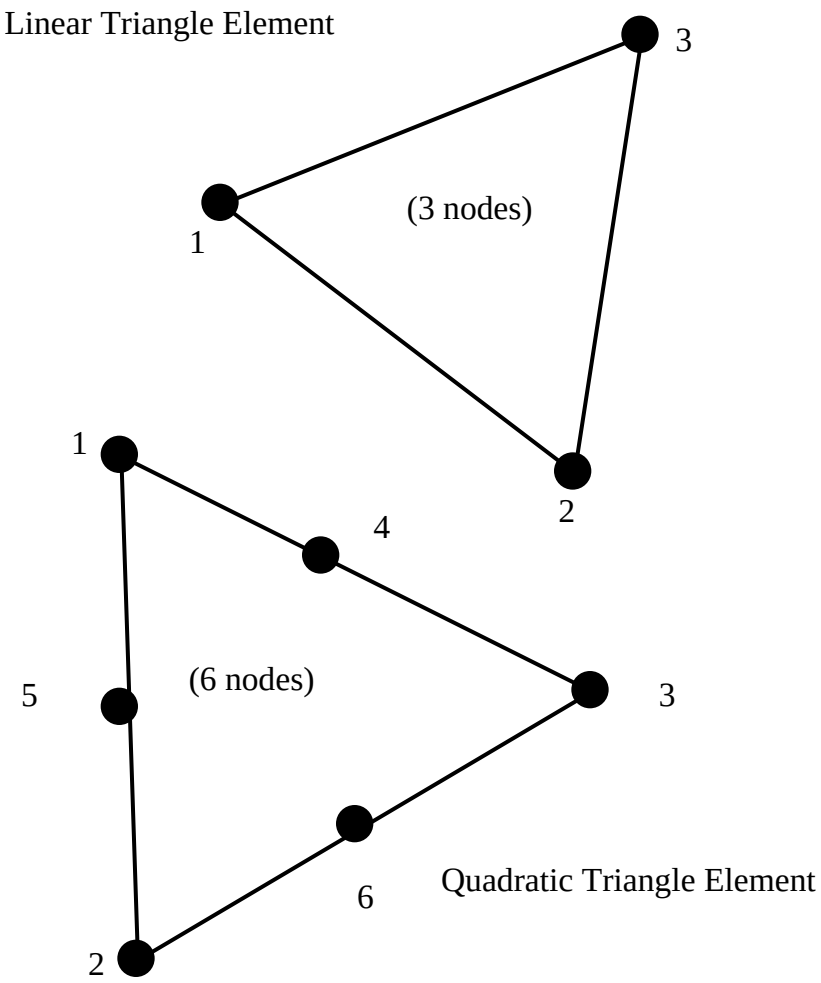

Figure 9.2: Linear interpolation over 3-node linear triangular elements is adequate for the solution of the equations for u and v in ice-shelf and ice-stream motion. Quadratic interpolation over 6-node quadratic triangular elements is needed to resolve second derivatives of s (*i.e.*, the curvature of the surface elevation) necessary for the evaluation of the vertical velocity and $\mathcal{D}$-term in the heat equation (Eqn. 9.44) for grounded (rigid bed) ice sheets.

9.5 Computation of u, v, w and the $\mathcal{D}$-term

We now address the nitty gritty of ice-sheet temperature modelling and begin with a discussion of how the velocity components referenced in the advection terms of Eqn. (9.44) are computed.

9.5.1 Horizontal Velocity

The horizontal velocity compontents, u and v, have both ζ-dependent and ζ-independent parts (depending on whether there is basal sliding or not, according to the common notion of basal sliding being a process that does not invoke the plug-flow dynamics described previously for ice-stream applications). The ζ-dependent parts owe their ζ dependence to the factor $I(\zeta)$ which appears in the second terms on the right-hand sides of Eqns. (9.1) and (9.2). The factor $I(\zeta)$ is computed using the vertical integral formula given in Eqn. (9.6). In MATLAB , this vertical integral is evaluated with the following statements:

```
for n=2:icelev
I(:,n)=E*grads2.*(Abar(:,n-1)+Abar(:,n))/2*((rho*g)^2) ...
           .* (((sbar-bbar).*(1-zetaice(n-1))).^3  ...
           + ((sbar-bbar).*(1-zetaice(n))).^3).*dz(:,n-1)/2 + I(:,n-1);
udefbar(:,n)=   -2*rho*g*I(:,n).*sxbar;
vdefbar(:,n)=    -2*rho*g*I(:,n).*sybar;
diffbar(:)=diffbar(:)+2*rho*g*(I(:,n-1)+I(:,n))/2.*dz(:,n-1);
end
```

where `udefbar` and `vdefbar` are the deformational parts of u and v, and `grads2`, `sxbar`, and `sybar` are the variables representing $(\nabla s \cdot \nabla s)$, $\frac{\partial s}{\partial x}$, and $\frac{\partial s}{\partial y}$, respectively:

```
for gauss=1:7
for m=1:6
```

```
sbar(:)=sbar(:)+GaussWeights(gauss).*(hgamma(:,m)+halpha(:,m).*xg(:,gauss) ...
+ hbeta(:,m).*yg(:,gauss) ...
+ hdelta(:,m).*xg(:,gauss).^2 + hepsilon(:,m).*yg(:,gauss).^2 ...
+ hphi(:,m).*xg(:,gauss).*yg(:,gauss) ) .* hs(hindex(:,m));

bbar(:)=bbar(:)+GaussWeights(gauss).*(hgamma(:,m)+halpha(:,m).*xg(:,gauss)...
+ hbeta(:,m).*yg(:,gauss) ...
+ hdelta(:,m).*xg(:,gauss).^2 + hepsilon(:,m).*yg(:,gauss).^2 ...
+ hphi(:,m).*xg(:,gauss).*yg(:,gauss) ) .* hb(hindex(:,m));

sxbar(:)=sxbar(:)+GaussWeights(gauss).*(halpha(:,m)   ...
    + 2*hdelta(:,m).*xg(:,gauss)  + hphi(:,m).*yg(:,gauss) ) .* hs(hindex(:,m));

sybar(:)=sybar(:)+GaussWeights(gauss).*( hbeta(:,m) ...
     + 2*hepsilon(:,m).*yg(:,gauss) + hphi(:,m).*xg(:,gauss)) .* hs(hindex(:,m));

sgauss(:,gauss)=sgauss(:,gauss)+(hgamma(:,m)+halpha(:,m).*xg(:,gauss)...
+ hbeta(:,m).*yg(:,gauss) ...
+ hdelta(:,m).*xg(:,gauss).^2 + hepsilon(:,m).*yg(:,gauss).^2 ...
+ hphi(:,m).*xg(:,gauss).*yg(:,gauss) ) .* hs(hindex(:,m));

bgauss(:,gauss)=bgauss(:,gauss)+(hgamma(:,m)+halpha(:,m).*xg(:,gauss)...
+ hbeta(:,m).*yg(:,gauss) ...
+ hdelta(:,m).*xg(:,gauss).^2 + hepsilon(:,m).*yg(:,gauss).^2 ...
+ hphi(:,m).*xg(:,gauss).*yg(:,gauss) ) .* hb(hindex(:,m));

end
end

for gauss=1:7
for m=1:6
for k=1:6

grads2(:)=grads2(:)+GaussWeights(gauss).* ...
((halpha(:,m)+2*hdelta(:,m).*xg(:,gauss)+hphi(:,m).*yg(:,gauss)) ...
.*(halpha(:,k)+2*hdelta(:,k).*xg(:,gauss)+hphi(:,k).*yg(:,gauss)) ...
```

```
+(hbeta(:,m)+2*hepsilon(:,m).*yg(:,gauss)+hphi(:,m).*xg(:,gauss)) ...
.*(hbeta(:,k)+2*hepsilon(:,k).*yg(:,gauss)+hphi(:,k).*xg(:,gauss)) ) ...
.*hs(hindex(:,m)).*hs(hindex(:,k));

grads2gauss(:,gauss)=grads2gauss(:,gauss) ...
+((halpha(:,m)+2*hdelta(:,m).*xg(:,gauss)+hphi(:,m).*yg(:,gauss)) ...
.*(halpha(:,k)+2*hdelta(:,k).*xg(:,gauss)+hphi(:,k).*yg(:,gauss)) ...
+ (hbeta(:,m)+2*hepsilon(:,m).*yg(:,gauss)+hphi(:,m).*xg(:,gauss)) ...
.*(hbeta(:,k)+2*hepsilon(:,k).*yg(:,gauss)+hphi(:,k).*xg(:,gauss)) ) ...
.*hs(hindex(:,m)).*hs(hindex(:,k));

end
end
end
```

The variable `GaussWeights` embodies the Gaussian quadrature integration technique used to construct averages and to determine matrix elements in the finite-element solution of the problem. More will be said about Gaussian quadrature later.

In circumstances where $I(x, y, z)$ is not a function of x and y (true for the Level 1 Eismint test, false for the Level 2 test where temperature variations are coupled to the flow law; a mistake I caught only recently, the reader is urged to use caution over the next bit, as I have not had a chance to correct this part of the text yet), the vertical velocity is computed with the following code,

```
for n=1:icelev
div(:,n)=2*( hs(hindex(:,1)).*hdelta(:,1)+ ...
                   hs(hindex(:,2)).*hdelta(:,2)+ ...
   hs(hindex(:,3)).*hdelta(:,3)+ ...
   hs(hindex(:,4)).*hdelta(:,4)+ ...
   hs(hindex(:,5)).*hdelta(:,5)+ ...
   hs(hindex(:,6)).*hdelta(:,6)+ ...
```

```
   hs(hindex(:,1)).*hepsilon(:,1)+ ...
                  hs(hindex(:,2)).*hepsilon(:,2)+ ...
   hs(hindex(:,3)).*hepsilon(:,3)+ ...
   hs(hindex(:,4)).*hepsilon(:,4)+ ...
   hs(hindex(:,5)).*hepsilon(:,5)+ ...
   hs(hindex(:,6)).*hepsilon(:,6)) ...
   .* (-2*rho*g*I(:,n)  ) ;
end

% Computation of the "D-term" and Vertical velocity:

Intdiv=zeros(nel,icelev);
Dterm=zeros(nel,icelev);
Vertvel=zeros(nel,icelev);
for n=2:icelev
Intdiv(:,n)=Intdiv(:,n-1)+(div(:,n-1)+div(:,n))/2*(zetaice(n)-zetaice(n-1));
Vertvel(:,n)=Vertvel(:,n-1)-(div(:,n-1)+div(:,n))/2.*dz(:,n-1);
end
```

The above fragments of MATLAB code are, of course, combined with a great deal of other code to simulate the combined mass balance/thermal evolution of an ice sheet.

Note added in proof: The above determination of the divergence of the horizontal velocity treats the integral $I(x, z)$ as a piecewise constant function (*i.e.,* constant within an element); thus, the finite-element scheme employed here gives

$$\frac{\partial u}{\partial x} \approx -2\rho g I(x, z) \frac{\partial^2 s}{\partial x^2} \tag{9.62}$$

A more accurate determination of the divergence would account for the variation of $I(x, z)$ within an element, *i.e.,*

$$\frac{\partial u}{\partial x} = -2\rho g I(x, z) \frac{\partial^2 s}{\partial x^2} + -2\rho g \frac{\partial I(x, z)}{\partial x} \frac{\partial s}{\partial x} \tag{9.63}$$

The results presented here treat I as a constant function within each element. This may account for differences between the test results presented here and those presented by Huybrechts and Payne (in press). While it may seem inappropriate not to treat I as variable within each element, such treatment is consistent with the Type I numerical scheme where "effective ice diffusion" (represented by D) is constant within each element.

9.6 Gaussian Quadrature

When the method of weighted residuals is used to solve the heat and mass balance equations, matrix terms for the s_j and $T_j(z_k)$'s (where j and k are horizontal and vertical node points) are generated via integrals of the form:

$$A_{\mathcal{I}(el,i)\mathcal{I}(el,j)} = A_{\mathcal{I}(el,i)\mathcal{I}(el,j)} + \mathcal{C} \int\int_{\Omega_{\text{el}}} N_i N_j dx dy \tag{9.64}$$

where $\mathcal{C}$ is an unspecified coefficient, $\mathcal{I}(el,i)$ and $\mathcal{I}(el,j)$ are the global node labels (numbers) for the i'th and j'th nodes in element el, and Ω_{el} is the two-dimensional plan area of the triangular element. Recalling that the interpolation functions $N_j(x,y)$ are second-order polynomials in x and y, we might find the task of evaluating the integral in Eqn. (9.65) daunting due to the tedious nature of double integral evaluation. In the previous chapters, we used exact integration formulae for integrands involving products of the L_j's. Such exact integration is possible here, but we shall use the present circumstance to introduce Gaussian quadrature.

Following Zienkiewicz [1971, p. 149], we evaluate the integral using an approximation involving a sum and the evaluation of the integrand at specific points within the triangular domain Ω_{el} called Gaussian quadrature points:

$$A_{\mathcal{I}(el,i)\mathcal{I}(el,j)} = A_{\mathcal{I}(el,i)\mathcal{I}(el,j)} + \mathcal{C} \sum_{g=1}^{N_g} \mathcal{W}_g N_i(x_g, y_g) N_j(x_g, y_g) \tag{9.65}$$

where N_g is the number of Gaussian quadrature points, (x_g, y_g) are the coordinates of the Gaussian quadrature points, and $\mathcal{W}_g$ is a Gaussian quadrature weighting coefficient.

According to Zeinkiewicz [1971, p. 151], round-off error in Gaussian quadrature is a function of N_g and the location of the quadrature points (x_g, y_g). A method optimized for reducing errors in the integrals of quintic polynomials (products such as $N_j(x,y)N_i(x,y)$ generate polynomials of order 4, so algorithms optimized for quintic polynomials are fully adequate for our purposes) has $N_g = 7$ with quadrature points and weights given by (see Fig. 9.3):

point	triangular coordinates			$\mathcal{W}_g$
a	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.225
b	a_1	b_1	b_1	0.13239415
c	b_1	a_1	b_1	0.13239415
d	b_1	b_1	a_1	0.13239415
e	a_2	b_2	b_2	0.12593918
f	b_2	a_2	b_2	0.12593918
g	b_2	b_2	a_2	0.12593918

where,

$$\begin{aligned} a_1 &= 0.05971587 \\ b_1 &= 0.47014206 \\ a_2 &= 0.79742699 \\ b_2 &= 0.10128651 \end{aligned}$$

and "triangular coordinates" refer to the definition given by Zeinkiewicz [1971, p. 117] restated as follows. The i'th triangular coordinate of a point P contained within a triangular element is defined as the ratio of the area of the triangle with P, vertex j, and vertex k as it's three vertices to the area of the entire triangle. Three triangular coordinates are necessary to uniquely specify point P within the (x, y) domain of the triangular element. In other

words,

$$x = [\, l_1 \quad l_2 \quad l_3 \,] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{9.66}$$

and,

$$y = [\, l_1 \quad l_2 \quad l_3 \,] \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{9.67}$$

where the l_i's are the triangular coordinate values, and the (x_j, y_j)'s are the vertex points of the triangle.

The MATLAB code which sets up the Gaussian quadrature points and weights for the models we will use in this chapter is provided as follows,

```
alpha1=0.05971587;
beta1=0.47014206;
alpha2=0.79742699;
beta2=0.10128651;

Lcoord=[ 1/3 1/3 1/3
alpha1 beta1 beta1
beta1 alpha1 beta1
beta1 beta1 alpha1
alpha2 beta2 beta2
beta2 alpha2 beta2
beta2 beta2 alpha2];

xg=zeros(nel,7);
yg=zeros(nel,7);
for n=1:nel
xg(n,:)=(Lcoord*[x(index(n,1)) x(index(n,2)) x(index(n,3))]')';
yg(n,:)=(Lcoord*[y(index(n,1)) y(index(n,2)) y(index(n,3))]')';
end

GaussWeights=[0.225 0.13239415 0.13239415 0.13239415 0.12593918 ...
 0.12593918 0.12593918]';
```

9.7 Numerical Integration of the Heat Equation

With the above preliminaries taken care of, we concentrate on describing a finite-element solution proceedure for Eqn. (9.44).

9.7.1 Split Timestep

One of the principal obstacles to "easy living" with ice-sheet thermodynamics is the fact that Eqn. (9.44) involves a three-dimensional, time-dependent variable $T(x, y, z, t)$. A true finite-element solution of this problem would involve three-dimensional elements that fill the volume of the ice sheet. This approach is problematical because the vertical dimension of the ice sheet is very much smaller than the horizontal dimension; so a space-filling finite element mesh would either have an unacceptably high number of elements or an unacceptably high aspect ratio (ratio of horizontal span to vertical span) if a low number of elements is used. The mathematical structure of Eqn. (9.44) suggests a means to overcome this problem. The diffusion term involves only vertical diffusion. Horizontal diffusion is absent (because of the low aspect ratio of the ice sheet, justification may be found in Hutter [?]). A natural simplification is thus to separate, or split, the horizontal part of the equation off from the vertical part. Using a finite-difference representation of the time derivatives, *i.e.*,

$$T_t = \frac{T^{n+1} - T^n}{\Delta t} \tag{9.68}$$

where n is the discrete time level, and Δt is the timestep size, we write a split timestep form of Eqn. (9.44) as follows:

$$\frac{\tilde{T}^{n+1}}{\Delta t} + u^n(\zeta_l)\tilde{T}_x^{n+1} + v^n(\zeta_l)\tilde{T}_y^{n+1} + \mathcal{S}(\tilde{T}^{n+1}; \zeta_l) = \frac{T^n}{\Delta t} \quad \forall k \tag{9.69}$$

Gaussian Quadrature points (Quintic polynomial accuracy)

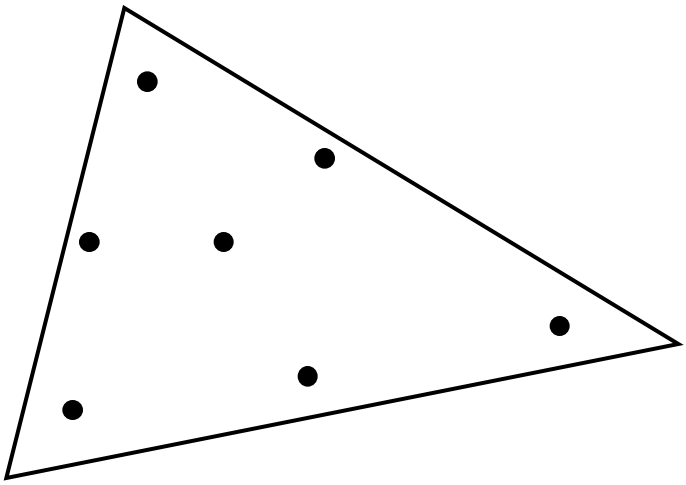

Figure 9.3: Numerical evaluation of double integrals is accuratly achieved by use of Gaussian quadrature, such as that which involves the 7 quadrature points shown above.

$$\frac{T^{n+1}}{\Delta t} - \frac{1}{\rho c h^2}\left(kT_\zeta^{n+1}\right)_\zeta + \frac{\omega(\zeta)}{h}T_\zeta^{n+1} = \frac{\tilde{T}^{n+1}}{\Delta t} + \frac{W(\zeta)}{\rho c} \quad \forall k \tag{9.70}$$

where in the first of the above equations, l is the vertical level number, ζ_l is the l'th vertical level in the ice, and $\omega(\zeta)$ is the vertical velocity in the "ζ-system" given by:

$$\omega = D(\zeta) - \zeta\dot{A} - (1-\zeta)\dot{B} \tag{9.71}$$

The term $\mathcal{S}(\tilde{T}^{n+1};\zeta_l)$ refers to an artificial horizontal diffusion term that can be either "upwinding" related or the "streamline upwind Petrov-Galerkin" (SUPG) term described below. This artificial diffusion term is added to supress numerical noise (wiggles) generated in high Peclet number flows (see the previous chapter). The $\mathcal{S}$-term will be explained later. (Note, in later versions of this model, we have eliminated artificial diffusion alltogether. This is possible when we do not attempt to impose ill-posed boundary conditions at the edges of the horizontal domain.)

We discretize the horizontal plan of the domain into a finite-element mesh consisting of two-dimensional elements, and discretize the vertical dimension of the domain using a stack of vertical levels spanning the range $\zeta \in [0,1]$ above each node point. The horizontal discretization uses 3-node triangles with linear interpolation (involving the linear interpolation functions $L_j(x,y)$ described earlier). Six-node triangles with quadratic interpolation (involving the quadratic polynomial interpolation functions $N_j(x,y)$) are needed only for the discretization of s, where second derivatives of s are important to determine vertical velocities needed in the heat equation. We stack additional vertical levels below the ice sheet to cover bedrock temperature conditions. The finite-element discretization used to solve the EISMINT Level 1 intercomparison benchmark problem (discussed below) is shown in Figs. (9.4) and (9.5). The vertical spacing of layers is defined by the following MATLAB statement in the programs presented here:

```
dummy=logspace(-2,-1,lowlayers);
zetaice=[logspace(-2,-1,lowlayers)-dummy(1)*ones(1,lowlayers) ...
   linspace(.13,1,icelev-lowlayers) ]'
```

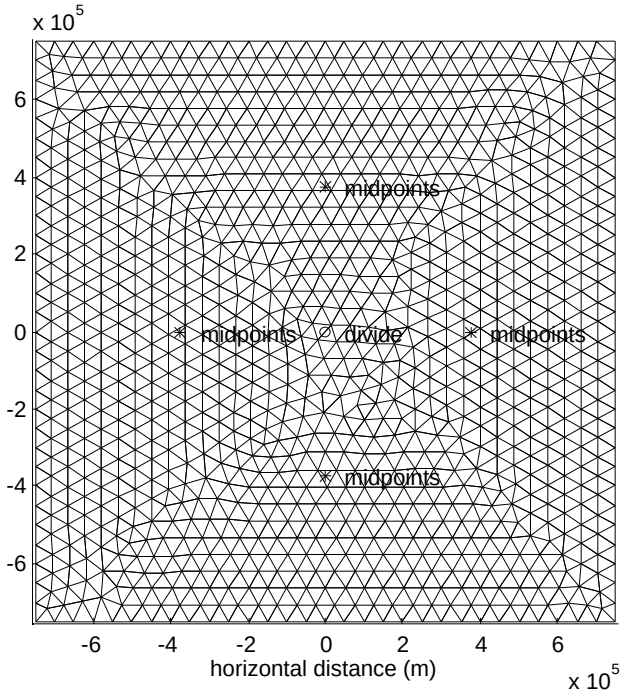

Figure 9.4: Finite-element mesh for EISMINT Level 1 (fixed margin) intercomparison benchmark test. The full 1500 × 1500 km domain is discretized because it is not practical to have internal boundaries where temperature boundary conditions might have to be specified (this is a subtle point having to do with the fact that the velocity in the temperature model code is not specified at nodes). For diagnostics, and to ensure an accurate intercomparison with other models, four ice-divide/edge midpoint sections were used to construct an average.

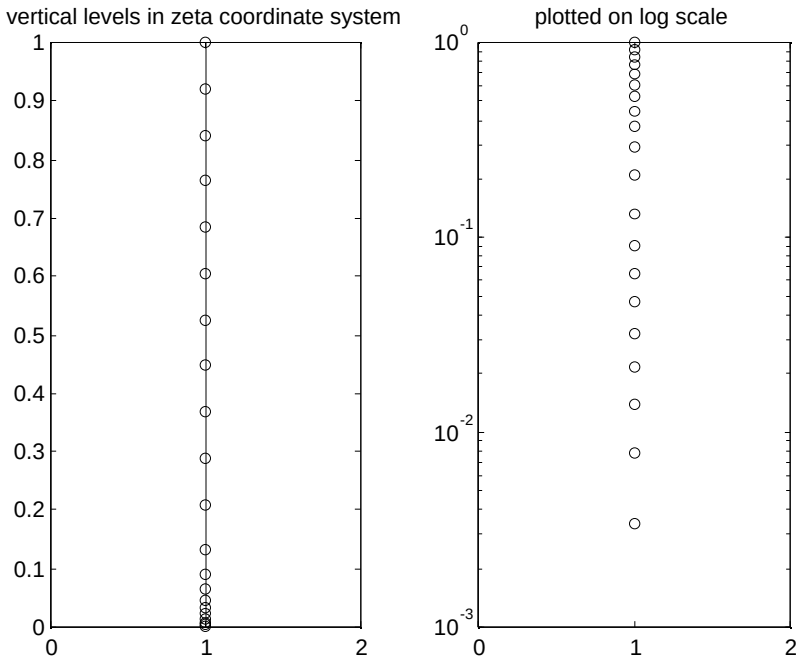

Figure 9.5: Twenty-one vertical levels were distributed logarithmically through the bottom 10% of the ice column, and linearly through the upper 90% of the ice column to properly represent the basal heat-flux boundary condition (bedrock was not included in the level 1 intercomparison benchmark test) and viscous dissipation (which is concentrated in the lower 10% of the ice column).

9.7.2 Horizontal Advection Equation: SUPG vs. "upwinding"

Let's face it. Wiggles are not a modeller's best friend. Wiggles, what "modelling grunts" call mesh-point-to-mesh-point numerical noise, can devastate modelling experiments because they represent an unphysical part of the solution of the model equations that can overwhelm the physical part of the solution. One way to avoid wiggles is to take small time steps. This remediation is expensive, because large amounts of computer time must be devoted to supressing a solution who's unwanted structure possesses the shortest possible timescales (order the time it takes for a point to move across a mesh element). Two common remediation techniques are called SUPG and "upwinding", and are described below.

"Upwinding"

In finite-element methods, the benefits of "upwinding" are achieved by adding an artificial diffusion term to the horizontal part of the heat equation (Eqn. 9.69). This artificial diffusion term is written:

$$\nabla \cdot (\mathbf{k}_{\text{upwind}} \nabla T) \tag{9.72}$$

where $\nabla = \frac{\partial}{\partial x}\mathbf{n}_x + \frac{\partial}{\partial y}\mathbf{n}_y$ is the horizontal "del" operator, and $\mathbf{k}_{\text{upwind}}$ is the artificial diffusivity tensor for "upwinding" defined by,

$$\mathbf{k}_{\text{upwind}} = \frac{1}{2}\begin{bmatrix} |u|\Delta x & 0 \\ 0 & |v|\Delta y \end{bmatrix} \tag{9.73}$$

where u and v are the x- and y-velocities, respectively, and Δx and Δy are the "widths" of each element in the x- and y-directions.

SUPG

Another approach to the repression of wiggles is the use of SUPG diffusion. Like "upwinding", this term is artificial and can be eliminated alltogether

by taking a sufficiently small Δt. The SUPG technique was developed by T. J. R. Hughes (not to be confused with "our" T. J. Hughes!) [1979, 1988]. (Refer to Hughes's articles to learn about other techniques, such as quadrature "upwinding".) Essentially, SUPG represents the addition of a nonisotropic diffusion that is active down flowlines and inactive across flowlines. Numerical noise is swept downstream without disturbing cross-stream structure in the temperature field. The SUPG term is defined by

$$\mathcal{S}(T;\zeta_k) = \nabla\left(\kappa_{\text{SUPG}}\nabla T\right) \tag{9.74}$$

where κ_{SUPG} is an artificial diffusivity tensor defined by

$$\left(\kappa_{\text{SUPG}}\right)_{ij} = \frac{1}{2}\left(|u_1|\Delta x_1 + |u_2|\Delta x_2\right)\hat{u}_i\hat{u}_j \tag{9.75}$$

where we have temporarily used subscripts i and j, and 1 and 2, to denote x- and y-components, where Δx_1 and Δx_2 are the horizontal "span" dimensions of each element (like $\mathbf{k}_{\text{upwind}}$, κ_{SUPG} varies from element to element), and

$$\hat{u}_k = \frac{|u_k|}{\left(u_l u_l\right)^{\frac{1}{2}}} \tag{9.76}$$

where $|\cdot|$ denotes absolute value, and repeated subscripts are to be summed. Note that the variable ζ_k is referenced in the definition of $\mathcal{S}(T;\zeta_k)$ because the u_i velocity components vary from one vertical level to another (one would not want to use the artificial diffusivity computed for levels near the ice-sheet surface to control wiggles near the ice-sheet bed!).

It is worth taking pause to describe how the method of weighted residuals will handle the SUPG term. (This is done because the artificial diffusivity tensor in the SUPG scheme can cause headaches for the unsuspecting SUPG neophyte.) We begin by manipulating the integral form of the SUPG term to distribute the derivatives onto the weighting function ψ (for notational convenience, we drop the subscript "SUPG" from κ_{SUPG}):

$$\int\int \psi\nabla\cdot\left(\kappa\nabla T\right)dxdy = \oint \not{\psi}\ {}_{jm}\frac{\partial T}{\partial x_m}\mathbf{n}_j ds - \int\int \nabla\psi\cdot\kappa\cdot\nabla T dxdy \tag{9.77}$$

where $\mathbf{n}_j$ is the j'th component of the outward pointing normal vector to the boundary. Observe that,

$$\nabla\psi\cdot\kappa\cdot\nabla T \;\; = k_{\text{SUPG}} \times \left[\begin{array}{cc} \frac{\partial\psi}{\partial x} & \frac{\partial\psi}{\partial y}\end{array}\right]\left[\begin{array}{cc}\hat{u}^2 & \hat{u}\hat{v}\\ \hat{u}\hat{v} & \hat{v}^2\end{array}\right]\left[\begin{array}{c}\frac{\partial T}{\partial x}\\ \frac{\partial T}{\partial y}\end{array}\right] \tag{9.78}$$

where $k_{\text{SUPG}} = \frac{1}{2}\left(|u_1|\Delta x_1 + |u_2|\Delta x_2\right) = \frac{1}{2}\left(|u|\Delta x + |v|\Delta y\right)$ is the scalar constant in the definition of κ_{SUPG}. The above vector-tensor-vector product is a scalar which can be written as follows,

$$\left[\begin{array}{cc} \frac{\partial\psi}{\partial x} & \frac{\partial\psi}{\partial y} \end{array}\right] \left[\begin{array}{cc} \hat{u}^2 & \hat{u}\hat{v} \\ \hat{u}\hat{v} & \hat{v}^2 \end{array}\right] \left[\begin{array}{c} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{array}\right] = \frac{\partial\psi}{\partial x}\left(\hat{u}^2\frac{\partial T}{\partial x} + \hat{u}\hat{v}\frac{\partial T}{\partial y}\right) + \frac{\partial\psi}{\partial y}\left(\hat{u}\hat{v}\frac{\partial T}{\partial x} + \hat{v}^2\frac{\partial T}{\partial y}\right) \tag{9.79}$$

As stated previously, the use of SUPG is optional. We find that it can be avoided when Δt is chosen to be very small. Definition of the SUPG diffusivity is done by empirical analysis (there is nothing special about the formula given above). Hughes suggests that SUPG according to the above prescription is comparable to, and possibly superior to, finite-difference "upwinding" because it allows good numerical stability without "crossflow" diffusion.

9.7.3 Vertical Advective/Diffusion Equation

Having glossed over the solution of the first part of the two-step timestepping procedure (the part involving horizontal advection and optional SUPG), we turn to the second part involving vertical advection, diffusion and heat generation by viscous dissipation. The method of weighted residuals is used to convert Eqn. (9.70) to the following integral form:

$$\begin{aligned} \int \psi\left(\frac{T^{n+1}}{\Delta t} - \frac{1}{\rho c h^2}\left(kT_\zeta^{n+1}\right)_\zeta + \frac{\omega(\zeta)}{h}T_\zeta^{n+1}\right)d\zeta \\ = \int \psi\left(\frac{\tilde{T}^{n+1}}{\Delta t} + \frac{W(\zeta)}{\rho c}\right)d\zeta \end{aligned} \tag{9.80}$$

The divergence theorem is used on the diffusion term on the left-hand side to distribute one ζ-derivative to the unknown weighting function ψ giving,

$$\begin{aligned} \int \left(\frac{\psi T^{n+1}}{\Delta t} + \left(\frac{\psi}{\rho c h^2}\right)_\zeta kT_\zeta^{n+1} + \frac{\psi\ (\zeta)}{h}T_\zeta^{n+1}\right)d\zeta \\ = \int \psi\left(\frac{\tilde{T}^{n+1}}{\Delta t} + \frac{W(\zeta)}{\rho c}\right)d\zeta + \psi\frac{k}{\rho c h^2}T_\zeta^{n+1}\Big|_{\zeta=0}^{\zeta=1} \end{aligned} \tag{9.81}$$

The last term on the right-hand side of the above equation expresses the boundary conditions at the top and bottom of the ice column. (Note that in the above development, we have not explicitly accounted for the possible inclusion of a bedrock layer below the ice-sheet bottom, *i.e.*, below $\zeta = 0$. To do so requires only minor modification of the above equation to account for the appropriate variation of k, ρ, c and $\omega(\zeta)$ in the elements below the ice base.) At boundaries where heat flux is specified (the bed when the bed is frozen), the factor $\frac{k}{\rho c h^2} T_\zeta^{n+1}$ at $\zeta = 0$ is replaced with the value of the heat flux (divided by $\rho c h^2$). At boundaries where temperature is specified (the surface), the weighting function ψ is set to zero and the constraint $T(\zeta = 1) = T_s$ is enforced, where T_s is the atmospherically determined surface temperature.

The finite-element solution of Eqn. (9.81) is accomplished by developing a vertical finite-element expression for $T(\zeta)$, k, c, ρ, W, ψ and ω at each individual node of the horizontal finite-element mesh, *e.g.,*

$$T(\zeta) = \sum_{k=1}^{2} M_j(\zeta) T_j \tag{9.82}$$

where j is the node index for 2-node line-segment elements extending through the vertical dimension of the ice column, and T_j is the value of T at the particular node. The interpolation function $M_j(\zeta)$ is linear, *i.e.,*

$$M_j(\zeta) = \mathcal{A}_j \zeta + \mathcal{G}_j \tag{9.83}$$

where the coefficients $\mathcal{A}_j$ and $\mathcal{G}_j$ are generated in the usual fashion from the constraints:

$$\begin{bmatrix} \zeta_1 & 1 \\ \zeta_2 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{A}_j \\ \mathcal{G}_j \end{bmatrix} = \begin{bmatrix} \delta_{1j} \\ \delta_{2j} \end{bmatrix} \tag{9.84}$$

where δ_{lk} is the Kronecker delta index (0 or 1, depending on if $l \neq k$ or not). The result of the above finite-element discretization of Eqn. (9.81) is a tridiagonal matrix algebra equation to be solved at each node point in the horizontal domain for the vertical profile of temperature through the ice column located there.

9.7.4 Summary of Numerical Integration of the Heat Equation

The two-step timestepping procedure can be summarized by expressing the variable T in its full, combined numerical representation. In each element,

$$T(x, y, \zeta, t) = \sum_{k=1}^{3} \left(\sum_{j=1}^{2} L_k(x, y) M_j(\zeta) T_{j,k}^n \right) \tag{9.85}$$

where n refers to the timestep level, $t_n = (n-1)\Delta t$, and $T_{j,k}^n$ is the temperature value at timestep n at the j'th vertical level of the k'th node. The horizontal advection equation is solved first, then the resulting solution is used in the right-hand side of the vertical advection/diffusion equation to complete the time step.

9.8 EISMINT Level 1 Fixed Margin Intercomparison Benchmark

To test the above conception of a finite-element ice-sheet dynamic/thermodynamic ice-sheet model, and to compare it with other finite-element and finite-difference models of a similar nature, we rerun the fixed margin intercomparison benchmark described in Chapter 2. This time, however, we run the benchmark using the 6-node element for the discretization of s and compute the temperature profile using the above prescription. Recall that the fixed-margin intercomparison test [now described fully by Huybrechts and others, in press, Annals of Glaciology 23] imposes fixed $\dot{A}$, k, c and a surface temperature (in Kelvin) given by

$$T_s = 239 + 8 \times 10^{-8} d^3 \tag{9.86}$$

where $d = \max(|x|, |y|)$, in a square domain centered at (0,0) with 1500-km sides, with zero ice thickness at the margins. The value of parameters used in this intercomparison experiment are summarized as follows:

parameter	value
$\dot{A}$	0.3 m a^{-1}
A	10^{-16} Pa^{-3} a^{-1}
g	9.81 m s^{-2}
ρ	910 kg m^{-3}
k	2.1 W m^{-1} K^{-1}
c	2009 J kg^{-1} K^{-1}
T_o	273.15 K
Φ	8.7×10^{-4} K m^{-1}
G	42 mW m^{-2}
31556926	s a^{-1}

Horizontal and vertical finite-element mesh discretizations are displayed in Figs. (9.4) and (9.5). Horizontal resolution is 50 km for temperature and 25 km for ice thickness (s) due to the fact that 6-node triangular elements are used to represent s. Differences between the present test and those of finite-difference models reported by Huybrechts and others (in press) may be due to the slightly higher resolution accomplished in the present study. Vertical resolution is variable, and is designed to represent the temperature profile in the bottom 10% of the ice column with a logarithmic spacing of nodes (finest spacing at the bottom). Spacing is linear in the upper 90% of the ice column. This arrangement is *ad hoc* but appears to work well in representing the vertical variation of $\omega(\zeta)$ and $W(\zeta)$ near the bed. (In the course of running the intercomparison experiment, it was learned that $\omega(\zeta)$ can have a small region of positive values located in the lowest 2-5 % of the ice column, especially at points remote from the ice divide. It was found to be critical to resolve this fine-scale variation to produce accurate temperature results.)

9.8.1 Results

According to the model intercomparison instructions, the model is run until steady state conditions prevail. Temperature, velocity, mass flux and ice thickness data are then sampled at specific points and along specific sections. In particular, temperature and vertical velocity profiles at the ice divide and

at "midpoint" nodes (here, taken to be the average of the four midpoints labeled in Fig. 9.4) are recorded for the intercomparison. Ice thickness, horizontal ice flux, and homologous basal temperature are presented along a transect extending from the ice divide to the midpoint of one of the ice-sheet's sides (here, we take the average of four such transects) and are recorded for intercomparison.

Cross sections of ice thickness and horizontal ice flux are displayed in Figs. (9.6) and (9.7). Homologous basal temperature along the same cross sections is displayed in Fig. (9.8). Vertical profiles of homologous temperature, vertical velocity and $\omega(\zeta)$ at the ice divide and at midpoints (which are averaged together) are displayed in Figs. (9.9) and (9.10). Ice flux at the midpoints is 742.5×10^2 m^2 a^{-1}, and basal temperature at the ice divide is -9.3564 C (these numbers can be compared with $795 \pm 5.67 \times 10^2$ m^2 a^{-1} and -8.97 ± 0.71 C, respectively, for the intercomparison results reported by Huybrechts and others, in press). Horizontal velocity data, and other miscellaneous diagnostics, are shown in Fig. (9.11).

Numerical values of various derived quantities are provided in the following table (other numerical data is available on request):

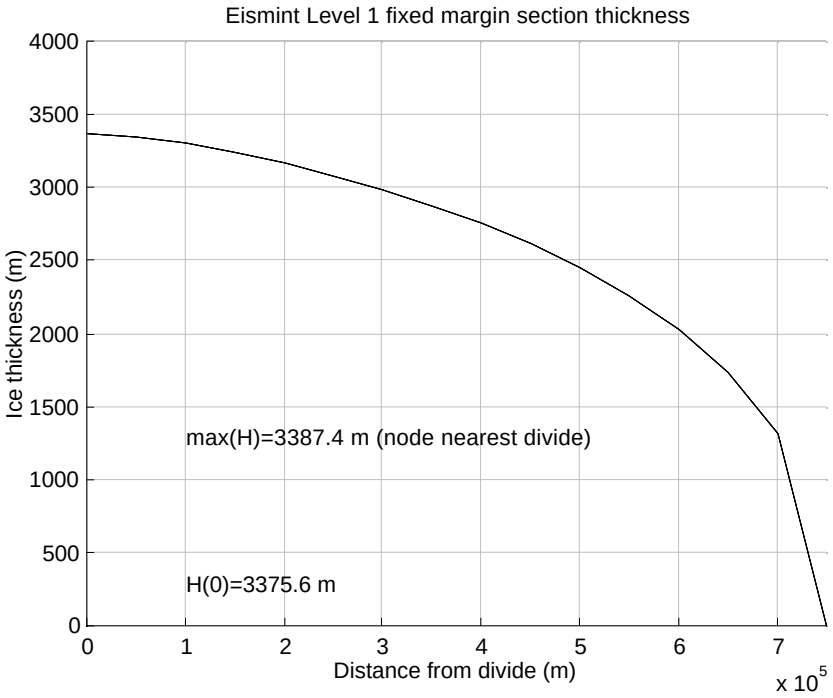

Figure 9.6: Ice thickness averaged over 4 ice-divide to side midpoint transects. Thickness at the point (0,0) where the ice divide is supposed to be is indicated along with the maximum thickness achieved at the true "numerical" ice divide located at the nearest node.

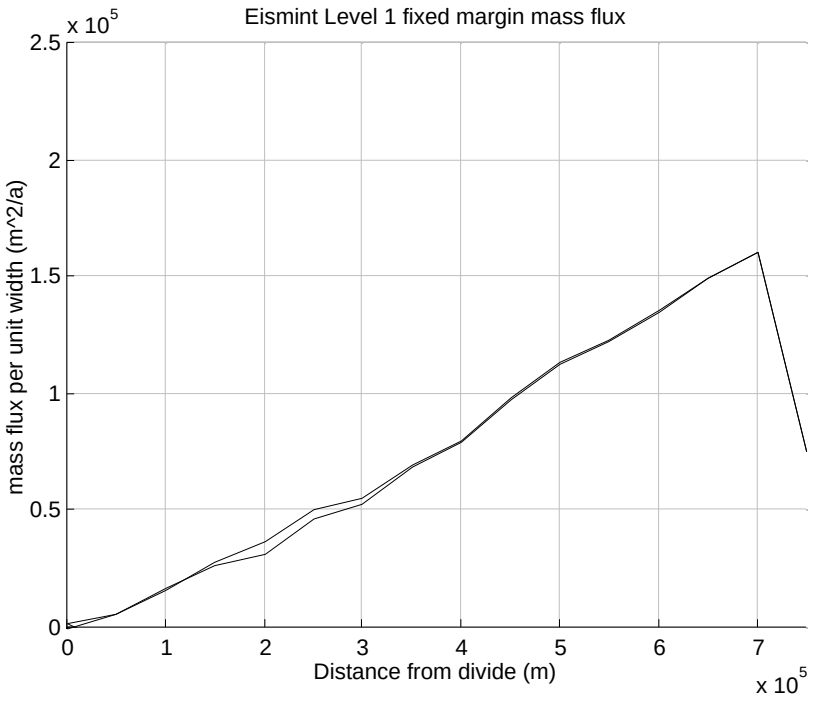

Figure 9.7: Horizontal depth-integrated ice flux averaged over 4 ice-divide to side midpoint transects.

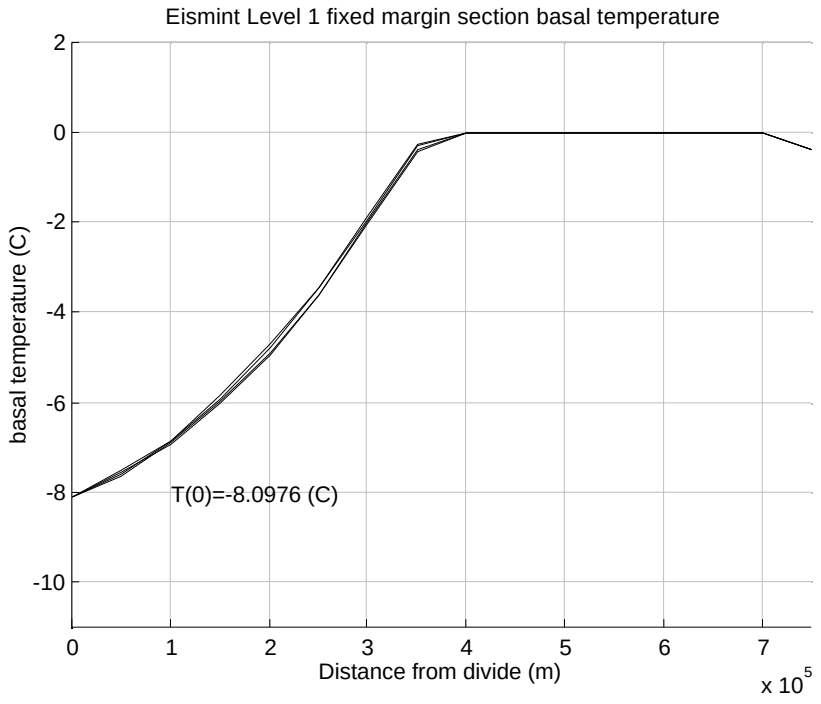

Figure 9.8: Homologous basal temperature along four ice divide to side midpoint transects.

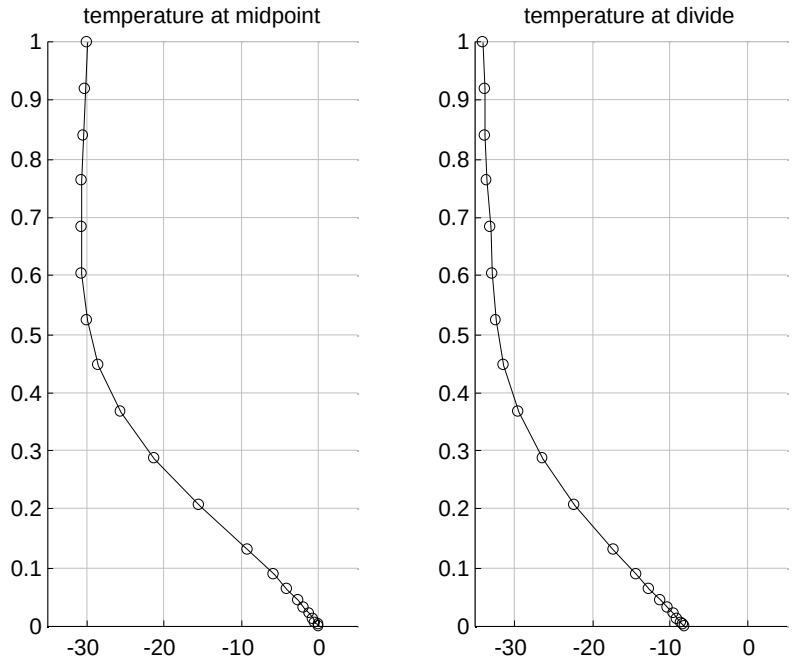

Figure 9.9: Homologous temperature at ice divide and midpoints (average of four).

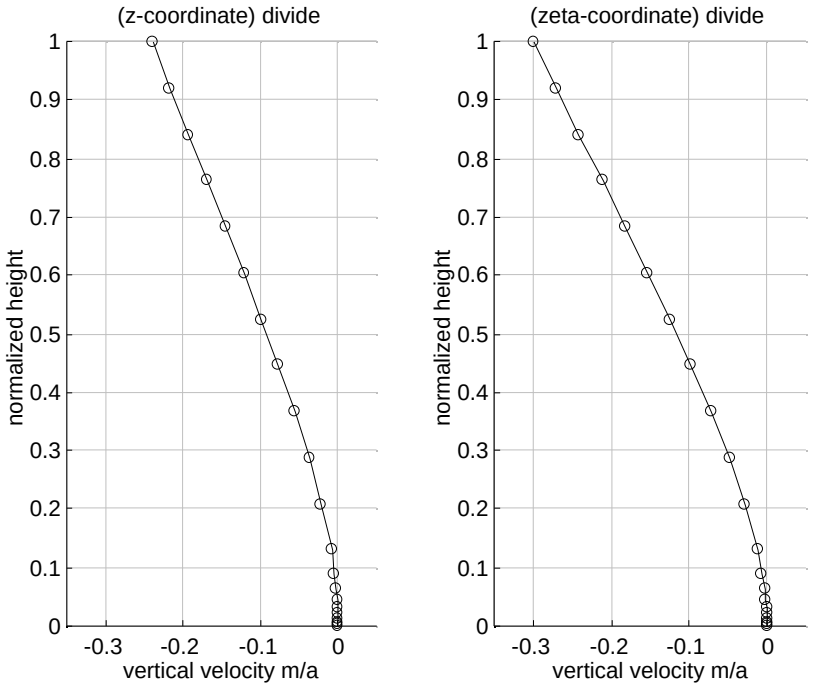

Figure 9.10: Vertical velocity and $\omega(\zeta)$ at an element which contains the ice divide (vertical velocity is not defined by the finite-element discretization at the ice divide).

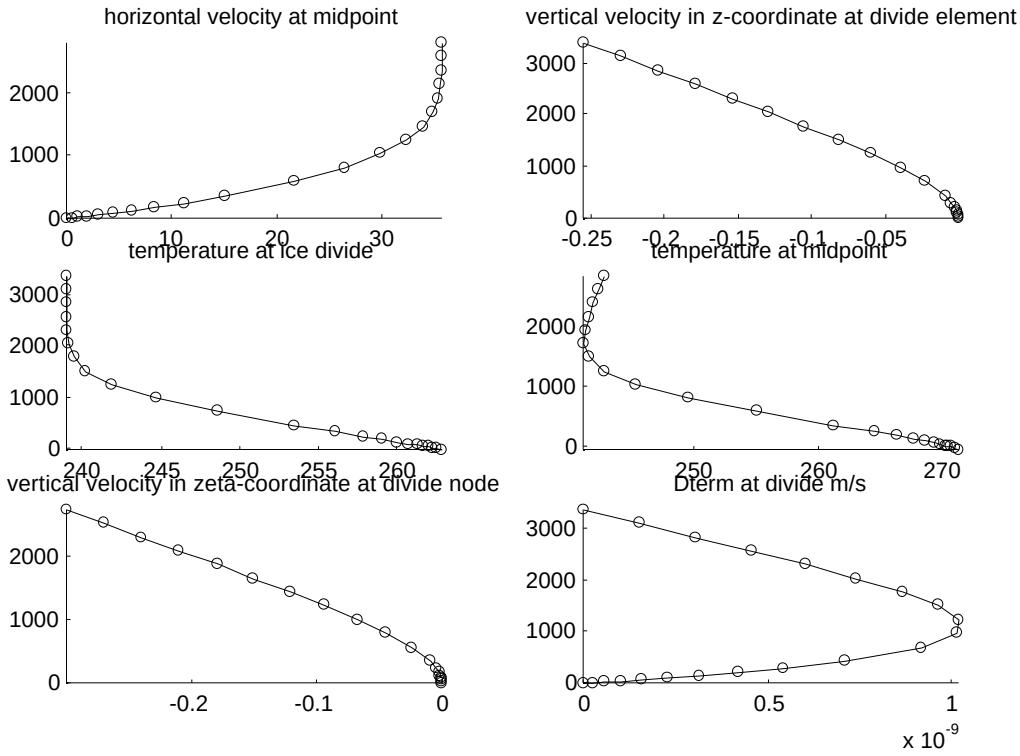

Figure 9.11: Horizontal velocity near the midpoint along one of the ice-divide to side midpoint transects and various other diagnostics, including the $\mathcal{D}$-term (lowest right panel).

ζ	T^*_{divide}	T^*_{midpoint}	ω_{divide}
0	-8.0976	0	0
0.0033	-8.3324	-0.1793	-0.0001
0.0078	-8.6458	-0.4244	-0.0002
0.0137	-9.0639	-0.7617	-0.0004
0.0216	-9.6218	-1.2293	-0.0007
0.0322	-10.3660	-1.8830	-0.0012
0.0462	-11.3584	-2.8038	-0.0022
0.0650	-12.6797	-4.1093	-0.0038
0.0900	-14.4316	-5.9637	-0.0067
0.1300	-17.1827	-9.1149	-0.0126
0.2091	-22.2352	-15.5039	-0.0284
0.2882	-26.4054	-21.2662	-0.0486
0.3673	-29.4187	-25.6934	-0.0721
0.4464	-31.3019	-28.5550	-0.0980
0.5255	-32.3366	-30.0599	-0.1253
0.6045	-32.8789	-30.6412	-0.1537
0.6836	-33.2027	-30.7181	-0.1826
0.7627	-33.4539	-30.5713	-0.2118
0.8418	-33.6877	-30.3417	-0.2412
0.9209	-33.9186	-30.0842	-0.2706
1.0000	-34.1493	-29.8930	-0.3000

where T^* refers to homologous temperature (temperature relative to the local pressure melting temperature). The finite-element MATLAB script used to generate the above solution is provided at the end of this chapter.

SUPG Example

The solution listed in the above table was generated using "upwinding" to supress wiggles. We also used an SUPG formulation to solve the same problem. Very little difference between the solutions, shown in the table given below, was noted.

$T^*_{\text{divide SUPG}}$	$T^*_{\text{divide upwind}}$	$T^*_{\text{midpoint upwind}}$	$T^*_{\text{midpoint SUPG}}$
-7.8779	-8.0976	0	0
-8.1128	-8.3324	-0.1793	-0.1843
-8.4262	-8.6458	-0.4244	-0.4362
-8.8443	-9.0639	-0.7617	-0.7825
-9.4024	-9.6218	-1.2293	-1.2620
-10.1471	-10.3660	-1.8830	-1.9314
-11.1405	-11.3584	-2.8038	-2.8728
-12.4637	-12.6797	-4.1093	-4.2046
-14.2196	-14.4316	-5.9637	-6.0920
-16.9808	-17.1827	-9.1149	-9.2898
-22.0658	-22.2352	-15.5039	-15.7442
-26.2805	-26.4054	-21.2662	-21.5350
-29.3396	-29.4187	-25.6934	-25.9626
-31.2595	-31.3019	-28.5550	-28.8110
-32.3175	-32.3366	-30.0599	-30.2997
-32.8715	-32.8789	-30.6412	-30.8639
-33.2000	-33.2027	-30.7181	-30.9188
-33.4528	-33.4539	-30.5713	-30.7398
-33.6872	-33.6877	-30.3417	-30.4642
-33.9185	-33.9186	-30.0842	-30.1455
-34.1493	-34.1493	-29.8930	-29.8930

The MATLAB code used to impliment the SUPG formulation is presented as follows. These statements should replace those for the "upwinding" scheme in the MATLAB finite-element model listed at the end of this chapter.

```
for lev=1:icelev
velmag(:,lev)=sqrt((udefbar(:,lev)+uslidebar(:)).^2   + (vdefbar(:,lev)+vslidebar(:)).^2);
uhat(:,lev)=abs(udefbar(:,lev)+uslidebar(:))./(velmag(:,lev).*   ...
(velmag(:,lev)~=zeros(nel,1))+ones(nel,1).*(velmag(:,lev)==zeros(nel,1)));
vhat(:,lev)=abs(vdefbar(:,lev)+vslidebar(:))./(velmag(:,lev).*   ...
(velmag(:,lev)~=zeros(nel,1))+ones(nel,1).*(velmag(:,lev)==zeros(nel,1)));    end

for lev=1:(icelev-1)
for m=1:3
for k=1:3
count=count+nel;
row(count:count+nel-1)=(lev-1)*nods*ones(nel,1)+index(:,m);
col(count:count+nel-1)=(lev-1)*nods*ones(nel,1)+index(:,k);
value(count:count+nel-1)=dt*area.*(  ( (m==k)/6 + (m~=k)/12 )/dt  ...
+  (abs(udefbar(:,lev)+uslidebar(:)).*dx(:)   + abs(vdefbar(:,lev)+vslidebar(:)).*dy(:)   )./2 ...
.* ( alpha(:,m).*alpha(:,k).*uhat(:,lev).^2   + alpha(:,m).*beta(:,k).*uhat(:,lev).*vhat(:,lev)   ...
```

```
      + beta(:,m).*alpha(:,k).*uhat(:,lev).*vhat(:,lev)   + beta(:,m).*beta(:,k).*vhat(:,lev).^2  ) ...
+  (udefbar(:,lev)/3  +uslidebar(:)/3).*alpha(:,k)   ...
+  (vdefbar(:,lev)/3  +vslidebar(:)/3).*beta(:,k)  );
end
end
end
```

The MATLAB script used to perform diagnostic analysis of model output is listed as follows:

```
Eismint Precision Diagnostics;

Temperature thickness and velocity at divide and midpoint(s)

xc=(x(index(:,1))+x(index(:,2))+x(index(:,3)))/3;
yc=(y(index(:,1))+y(index(:,2))+y(index(:,3)))/3;

xdivide=0;
ydivide=0;

[junk,seq]=sort(  (xdivide-xc(:)).^2  + (ydivide-yc(:)).^2  );
notfound=1;
n=0;
tritr=0;
while (notfound & n<=9)

n=n+1;

T1=[ 1              1              1
   x(index(seq(n),1))  x(index(seq(n),2))  xdivide
y(index(seq(n),1))  y(index(seq(n),2))  ydivide ];
T2=[ 1              1              1
   x(index(seq(n),2))  x(index(seq(n),3))  xdivide
    y(index(seq(n),2))  y(index(seq(n),3))  ydivide ];
T3=[ 1              1              1
   x(index(seq(n),3))  x(index(seq(n),1))  xdivide
y(index(seq(n),3))  y(index(seq(n),1))  ydivide ];


 if (abs(area(seq(n))-sum(abs(det(T1))+abs(det(T2))+abs(det(T3)))/2)<=1e-6*area(seq(n))    )
 notfound=0;
 tritr=seq(n);
 end


end

dividelement=tritr

compute fields at the divide:
```

```
sdivide=(gamma(dividelement,:))*s(index(dividelement,:))
max(hs)
wdivide=zeros(1,icelev);
for lev=1:icelev
Tdivide(lev)=(gamma(dividelement,:))*T(index(dividelement,:),lev+rocklev);
wdivide(lev)=(gamma(dividelement,:))*w(index(dividelement,:),lev);
end
for lev=1:icelev
THdivide(lev)=(gamma(dividelement,:))*TH(index(dividelement,:),lev);
end


figure(4)
clg
subplot(1,2,1)
axis([-0.35  0.05 0.0  1.0]),grid,hold  on
plot(31556926*Vertvel(dividelement,:),zetaice','w-')
plot(31556926*Vertvel(dividelement,:),zetaice','wo')
title('(z-coordinate)  divide')
xlabel('vertical  velocity m/a')
ylabel('normalized  height')
subplot(1,2,2)
axis([-0.35  0.05 0.0  1.0]),grid,hold  on
plot(31556926*wdivide,zetaice','w-')
plot(31556926*wdivide,zetaice','wo')
title('(zeta-coordinate)  divide')
xlabel('vertical  velocity m/a')
ylabel('normalized  height')



xmid=[-375e3  0 375e3 0];
ymid=[0  -375e3 0 375e3];

for nn=1:4
[junk,seq]=sort(  (xmid(nn)-xc(:)).^2  + (ymid(nn)-yc(:)).^2  );
notfound=1;
n=0;
tritr=0;
while (notfound & n<=9)

n=n+1;

T1=[ 1              1              1
   x(index(seq(n),1))  x(index(seq(n),2))  xmid(nn)
y(index(seq(n),1))  y(index(seq(n),2))  ymid(nn) ];
T2=[ 1              1              1
   x(index(seq(n),2))  x(index(seq(n),3))  xmid(nn)
    y(index(seq(n),2))  y(index(seq(n),3))  ymid(nn) ];
T3=[ 1              1              1
   x(index(seq(n),3))  x(index(seq(n),1))  xmid(nn)
y(index(seq(n),3))  y(index(seq(n),1))  ymid(nn) ];
```

```
 if (abs(area(seq(n))-sum(abs(det(T1))+abs(det(T2))+abs(det(T3)))/2)<=1e-6*area(seq(n))    )
 notfound=0;
 tritr=seq(n);
 end


end

midelement(nn)=tritr
end

compute fields at the midpoint:
smid=zeros(1,4);
Tmid=zeros(4,icelev);
THmid=zeros(4,icelev);
umid=zeros(4,icelev);
vmid=zeros(4,icelev);

for n=1:4
smid(n)=(gamma(midelement(n),:)+alpha(midelement(n),:)*xmid(n)...
+beta(midelement(n),:)*ymid(n)  )*s(index(midelement(n),:));


for lev=1:icelev
Tmid(n,lev)=(gamma(midelement(n),:)+alpha(midelement(n),:)*xmid(n)...
+beta(midelement(n),:)*ymid(n)  )*T(index(midelement(n),:),lev+rocklev);
umid(n,lev)=31556926*udefbar(midelement(n),lev);
vmid(n,lev)=31556926*vdefbar(midelement(n),lev);

end
for lev=1:icelev
THmid(n,lev)=(gamma(midelement(n),:)+alpha(midelement(n),:)*xmid(n)...
+beta(midelement(n),:)*ymid(n)  )*TH(index(midelement(n),:),lev);
end


end


speedmid=sqrt(umid.^2+vmid.^2)

figure(6)
clg
axis([0 40 0 1]),grid,hold on
plot(speedmid(1,:),zetaice','w-')
plot(speedmid(1,:),zetaice','wo')
plot(speedmid(2,:),zetaice','w-')
plot(speedmid(3,:),zetaice','w-')
plot(speedmid(4,:),zetaice','w-')
title('horizontal  velocity at midpoints')
xlabel('velocity  m/a')
ylabel('normalized  height')

midpointthick=mean(smid)
```

```
figure(1)
clg
axis([-755e3 755e3 -755e3 755e3]),axis('image'),shading  flat,hold on,colormap('jet')
for n=1:nel
plot(x(index(n,:)),y(index(n,:)),'w-')
end
title('Eismint Level 1 (fixed Margin) diagnostic points on FE mesh')
plot(xmid,ymid,'w*')
text(xmid+50e3,ymid,'midpoints')
plot(xdivide,ydivide,'wo')
text(xdivide+50e3,ydivide,'divide')
xlabel('horizontal  distance  (m)')


figure(5)
clg
subplot(1,2,1)
axis([-35  5 0 1]),grid,hold  on
plot(mean(THmid)',[   zetaice'],'w-')
plot(mean(THmid)',[  zetaice'],'wo')
title('temperature  at midpoint')
subplot(1,2,2)
axis([-35  5 0  1]),grid,hold  on
plot(THdivide(:),zetaice','w-')
plot(THdivide(:),zetaice','wo')
title('temperature  at divide')

xsec=[linspace(-750e3,0,16)
zeros(1,16)
linspace(0,750e3,16)
zeros(1,16)];
ysec=[zeros(1,16)
linspace(-750e3,0,16)
zeros(1,16)
linspace(0,750e3,16)];
secelement=zeros(4,16);

for nnn=1:16
for nn=1:4
[junk,seq]=sort(  (xsec(nn,nnn)-xc(:)).^2  + (ysec(nn,nnn)-yc(:)).^2  );
notfound=1;
n=0;
tritr=0;
while (notfound & n<=9)

n=n+1;

T1=[ 1              1              1
   x(index(seq(n),1))  x(index(seq(n),2))  xsec(nn,nnn)
y(index(seq(n),1))  y(index(seq(n),2))  ysec(nn,nnn) ];
T2=[ 1              1              1
   x(index(seq(n),2))  x(index(seq(n),3))  xsec(nn,nnn)
    y(index(seq(n),2))  y(index(seq(n),3))  ysec(nn,nnn) ];
T3=[ 1              1              1
```

```
   x(index(seq(n),3))  x(index(seq(n),1))  xsec(nn,nnn)
y(index(seq(n),3))  y(index(seq(n),1))  ysec(nn,nnn) ];


 if (abs(area(seq(n))-sum(abs(det(T1))+abs(det(T2))+abs(det(T3)))/2)<=1e-6*area(seq(n))    )
 notfound=0;
 tritr=seq(n);
 end


end

secelement(nn,nnn)=tritr
end
end

ssec=zeros(4,16);
Tbsec=zeros(4,16);
TbHsec=zeros(4,16);
Qsec=zeros(4,16);



for nn=1:16
Qsec(1,nn)=31556926*diffbar(secelement(1,nn))*sxbar(secelement(1,nn));
Qsec(2,nn)=31556926*diffbar(secelement(2,nn))*sybar(secelement(2,nn));
Qsec(3,nn)=31556926*diffbar(secelement(3,nn))*sxbar(secelement(3,nn));
Qsec(4,nn)=31556926*diffbar(secelement(4,nn))*sybar(secelement(4,nn));
for n=1:4
ssec(n,nn)=(gamma(secelement(n,nn),:)+alpha(secelement(n,nn),:)*xsec(n,nn)...
+beta(secelement(n,nn),:)*ysec(n,nn)  )*s(index(secelement(n,nn),:));



Tbsec(n,nn)=(gamma(secelement(n,nn),:)+alpha(secelement(n,nn),:)*xsec(n,nn)...
+beta(secelement(n,nn),:)*ysec(n,nn)  )*T(index(secelement(n,nn),:),1+rocklev);
TbHsec(n,nn)=(gamma(secelement(n,nn),:)+alpha(secelement(n,nn),:)*xsec(n,nn)...
+beta(secelement(n,nn),:)*ysec(n,nn)  )*TH(index(secelement(n,nn),:),1);


end
end
Tbsec=Tbsec-Tnot;

distance=sqrt(xsec.^2+ysec.^2);
figure(2)
clg
axis([0 750e3 0 4000]),grid,hold on
plot(distance(1,:),ssec(1,:),'w-')
plot(distance(2,:),ssec(2,:),'w-')
plot(distance(3,:),ssec(3,:),'w-')
plot(distance(4,:),ssec(4,:),'w-')
title('Eismint Level 1 fixed margin section thickness')
xlabel('Distance from divide (m)')
ylabel('Ice thickness (m)')
```

```
text(100e3,300,'H(0)=3375.6  m')
text(100e3,1300,'max(H)=3387.4  m (node nearest divide)')

figure(7)
clg
axis([0 750e3 0 250e3]),grid,hold  on
plot(distance(1,:),Qsec(1,:),'w-')
plot(distance(2,:),Qsec(2,:),'w-')
plot(distance(3,:),Qsec(3,:),'w-')
plot(distance(4,:),Qsec(4,:),'w-')
title('Eismint  Level 1 fixed margin mass flux')
xlabel('Distance  from divide  (m)')
ylabel('mass  flux per unit width  (m^2/a)')


figure(3)
clg
axis([0 750e3 -11 2]),grid,hold  on
plot(distance(1,:),TbHsec(1,:),'w-')
plot(distance(2,:),TbHsec(2,:),'w-')
plot(distance(3,:),TbHsec(3,:),'w-')
plot(distance(4,:),TbHsec(4,:),'w-')
title('Eismint  Level 1 fixed margin section basal temperature')
xlabel('Distance  from divide  (m)')
ylabel('  basal temperature  (C)')
text(100e3,-8,'T(0)=-9.3565  (C)')
```

Model Intercomparisons

Huybrechts and others [in press] present an intercomparison of a number of finite-difference model results using the intercomparison benchmark we present above. The model results presented by Huybrechts and others are similar to those developed here. There are several points to be noted in the comparison:

1. Basal temperatures in the finite-element model make the transition to the pressure melting point at a position closer to the ice divide than those of the finite-difference models (lumped together as a group). This may be due to the fact that the finite-element model resolves the $\mathcal{D}$-term with a finer vertical separation than some of the finite-difference models. Any modification of the vertical velocity field, such as a finer resolution, will have a major effect on basal temperature (this was

pointed out by Huybrechts and others in their paper). Note the small area of positive $\omega(\zeta)$ near the bed in Fig. (9.11). The ability of the finite-element model to resolve this small area may make a substantial difference in the basal temperature achieved by the model.

2. Ice divide thickness in the finite-element model is between the Type I and Type II versions of the finite-difference models. This may be due to the fact that our 6-node finite-element used to represent s effectively resolves the thickness field on a 25-km horizontal scale (the finite-difference representations resolve s on a 50-km horizontal scale).

3. The temperature profile at the midpoint displays a slightly less pronounced temperature minimum at depth (due to horizontal advection of cold ice from upstream) than that displayed by the finite-difference models. This may be due, in part, to the fact that the finite-difference models achieve higher resolution in the upper ice column than that achieved by the finite-element model where the concentration of nodes is focussed in the lowest 10% of the ice column. Another possible reason for the difference is the use of "upwind" (and, alternatively, SUPG) artificial diffusion in the finite-element model to control numerical noise generated by the horizontal advection terms.

4. Mass flux along the ice-divide to margin-midpoint transect is slightly less in the finite-element results. Overall, the finite-element model and the Type I finite-difference models conserve mass. The Type II finite-difference models do not conserve mass (according to Huybrechts and others), but are advantageous for being able to take long timesteps. A second formulation of the finite-element model developed below is presented to show how a long-timestep formulation of the mass balance portion of the model can be achieved.

9.9 Timestep Size and the "Tiling Instability"

Huybrechts and others [in press] reported the effect of various schemes for representing D the "effective diffusivity" on the finite-difference grids (see Eqn. 9.12). Two classes of schemes were reported. Those designated Type I had the property of conserving mass, but timesteps were required to be quite small (order 10 years) to keep numerical instability at bay. Those designated Type II did not conserve mass as well, but were advantageous in allowing long timesteps (desireable in long time integrations in typical ice-sheet model experiments).

A similar classification of schemes can be developed for the finite-element method presented here. The scheme similar to Type I of the finite-difference categories has been presented above. The effective diffusivity D is computed as an element average, and used as a piecewise constant function in the method of weighted residuals solution of Eqn. (12.13). The D was computed in the MATLAB script with the following statement:

```
diffbar(:)=grads2.*sbar.^5*3.169e-24*(rho*g)^3*2/5;
```

where `grads2` and `sbar` are elemental averages of $\nabla s \cdot \nabla s$ and s, respectively.

We found that this scheme produced ice-sheet characteristics very similar to those resulting from Type I finite-difference schemes. Indeed, the timestep size necessary to maintain stability of the finite-element model using piecewise constant D had to be short, on the order of 5-10 years. (We make note of the fact that mass is always conserved by the finite-element method as defined by whatever scheme is used for the representation of D throughout the domain.) When timestep size was greater than about 10 years, a "tiling instability" developed in the piecewise constant D-field. A tiling instability gets its name from the fact that the values of D alternate from high to low from element to element, giving the pattern reminicient of an Italian fresco.

An alternative technique, somewhat comparable to the Type II class of finite-difference schemes, is to smooth D by interpolating it to the nodes

(using a 3-node triangle), then to integrate Eqn. (12.13) using a linear interpolation of D from the nodes to the Gaussian quadrature points (*i.e.,* using the *linear* interpolation polynomials $L_j(x, y)$). The MATLAB code for this alternative treatment is:

```
diffbar(:)=grads2.*sbar.^5*3.169e-24*(rho*g)^3*2/5;

diffnodes=zeros(nods,1);
totarea=zeros(nods,1);
for n=1:nel
totarea(index(n,:))=totarea(index(n,:))+area(n);
end
for n=1:nel
diffnodes(index(n,:))=diffnodes(index(n,:))+diffbar(n)*area(n);
end
diffnodes(:)=diffnodes(:)./totarea;

for gauss=1:7
diffgauss(:,gauss)=diffnodes(index(:,1)).*(gamma(:,1)+alpha(:,1).*xg(:,gauss)+beta(:,1).*yg(:,gauss))    ...
 + diffnodes(index(:,2)).*(gamma(:,2)+alpha(:,2).*xg(:,gauss)+beta(:,2).*yg(:,gauss))    ...
 + diffnodes(index(:,3)).*(gamma(:,3)+alpha(:,3).*xg(:,gauss)+beta(:,3).*yg(:,gauss))    ;
end
```

Type II Finite-Element Model Results

We shall call the above described scheme, in which the effective diffusivity D is smoothed, a "Type II" scheme after the nomenclature used by Huybrechts and others [in press]. The results of this scheme are shown in Figs. (9.12) - (9.16). The Type II scheme produces quite satisfactory mass-balance results (the mass flux is actually smoother than that of the Type I scheme near the ice-sheet margin). The temperature results for the Type II finite-element method, however, depart significantly from those of the Type II finite-difference results (*e.g.,* basal temperature at the ice divide is about 2-degrees colder in the Type II finite-element results than in the Type II finite-difference results) because the vertical velocity $\omega(\zeta)$ is not computed with sufficient accuracy in the Type II finite-element model. The main symptom of the poor vertical velocity is that the base of the ice sheet below the ice divide is about 3 C colder than in the Type I finite-element results. This problem demonstrates the extreme sensitivity of ice-sheet temperature modelling to the accuracy of vertical velocity.

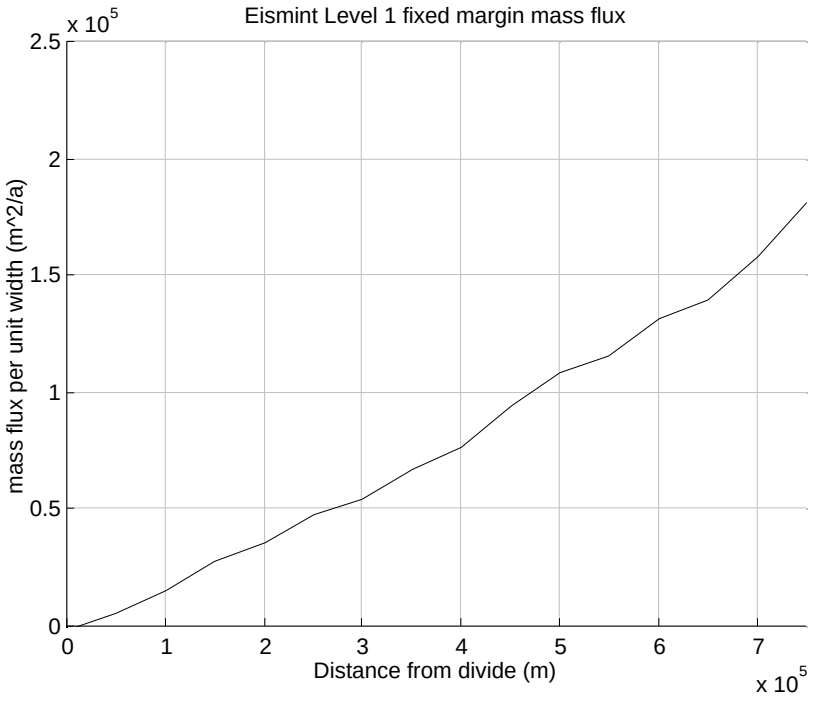

Figure 9.12: Horizontal mass flux along divide to side midpoint transect using Type II finite-element model (in which effective diffusivity is smoothed so to better take long timesteps).

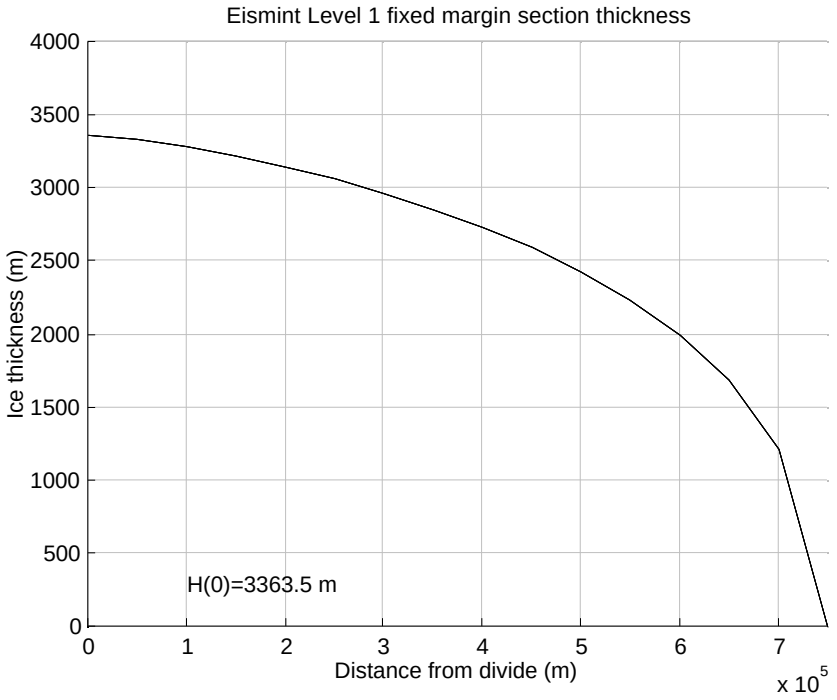

Figure 9.13: Ice thickness along divide to side midpoint transect using Type II finite-element model (in which effective diffusivity is smoothed so to better take long timesteps).

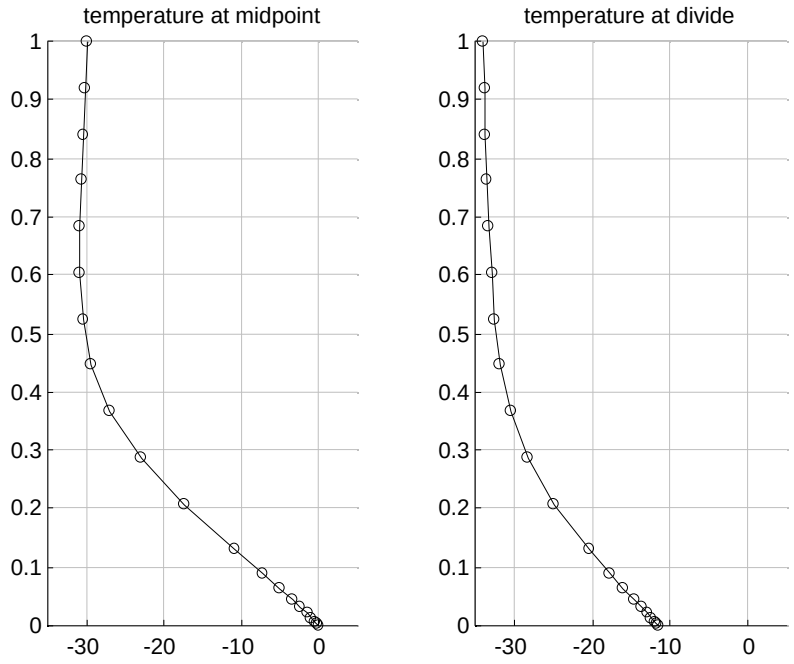

Figure 9.14: Homologous temperature profiles using Type II finite-element model (in which effective diffusivity is smoothed so to better take long timesteps). Note that the basal temperature at the ice divide is considerably cooler (by about 3 C) than that produced by the Type I finite-element method. This difference is primarily due to the inaccuracy of vertical velocity in the Type II finite-element method.

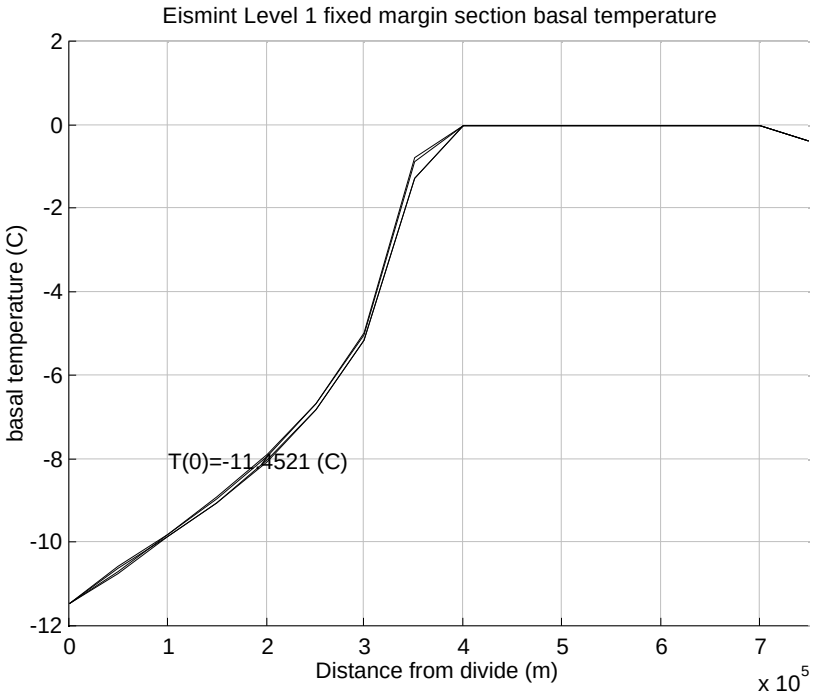

Figure 9.15: Homologous basal temperature along divide to side midpoint transect using Type II finite-element model (in which effective diffusivity is smoothed so to better take long timesteps). Note that the basal temperature at the ice divide is considerably cooler (by about 3 C) than that produced by the Type I finite-element method. This difference is primarily due to the inaccuracy of vertical velocity in the Type II finite-element method.

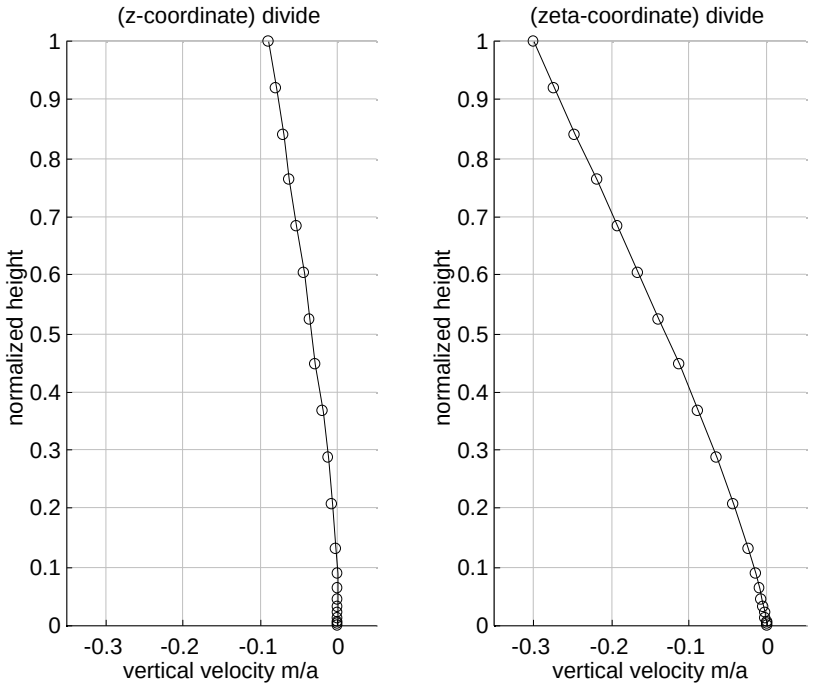

Figure 9.16: Vertical velocity at ice divide using Type II finite-element model (in which effective diffusivity is smoothed so to better take long timesteps). Note that the curvature of $\omega(\zeta)$ is reversed from that produced by the Type I finite-element method. This difference is primarily due to the inaccuracy of vertical velocity in the Type II finite-element method, and leads to colder temperature at the bed in the Type II run.

9.10 MATLAB Code Used to Produce Finite-Element Model Benchmark

To provide a record of the finite-element code employed in the model intercomparison benchmark developed here, and to provide an idea of how such a finite-element model can be constructed, we list the MATLAB code below. Note that in this implimentation, the "upwind" artificial diffusion technique is used to supress wiggles. A demonstration of SUPG is provided above.

```
% This code computes u, v and w for "inland ice" flow (Greve, 1995 and Payne, 1995)

%***************
%
% Designed to perform EISMINT level 1 test
%
%***************

dt=1000*31556926;

Phi=8.71e-4 /(917*9.81);% K Pa^-1 Payne, 1995  note error in Payne's table
Phi=8.7e-4 /(917*9.81);
Tnot=273.15;
Rgas=8.31;
E=1.0; % figuring "glacial" in literature really means a narrow time period
rho=910;
g=9.81;
Ao=1e-16 *1/31556926;
fusion=3.35e5;
mantleconst=0;
rhomantle=3000;

icelev=21;
rocklev=2;
nlev=icelev+rocklev;
lowlayers=9;
lindex=[linspace(1,nlev-1,nlev-1)
linspace(2,nlev,nlev-1)]';
nvel=nlev-1;


Adot=.3/31556926*ones(nods,1);
Bdot=zeros(nods,1);
Geoflux=0.042;  % geothermal flux for Canadian rocks
krock=1.041e8/3.1559e7;  % Heat conductivity of rock (from Huybrechts)
rhorock=2700;  % My assumed density of surface rock (from my brain)
caprock=1000;  % Huybrecht's heat capacity of rock
kapparock=krock/rhorock/caprock;
kice=zeros(nods,icelev);
```

```
capice=zeros(nods,icelev);
capicebar=zeros(nel,icelev);
melted=zeros(nods,1);  % initially frozen
heatin=zeros(nods,1);
heatout=zeros(nods,1);
heatprod=zeros(nods,1);
hwater=zeros(nods,1);



% mass balance values
mb=zeros(nods,1);  % Note, this is positive for freezing
ma=.3/3.1559e7*ones(nods,1);

% initial conditions
hs=1*ones(nods+nshadow,1).*mhs(:)  + (~mhs); % Note, minimum ice thickness is 1 m
s=1*ones(nods,1).*ms(:)  + (~ms);
radius=sqrt(x.^2+y.^2);
rockscale=50;


dzrock=rockscale/rocklev;
hb=zeros(nods+nshadow,1);
bnot=zeros(nods,1);
b=bnot;  %
hb=zeros(nods+nshadow,1);
hbnot=hb;
%b=bnot-rho/rhomantle*(s-b);
%hb=hbnot-rho/rhomantle*(hs-hb);
T=zeros(nods,nlev);
dist=zeros(nods,1);
for n=1:nods
dist(n)=max([abs(x(n))  abs(y(n))]);
end
for n=1:nlev
T(:,n)=(239*ones(nods,1)+8e-8*(dist/1000).^3);
end
T(:,nlev)=(239*ones(nods,1)+8e-8*(dist/1000).^3);
theta=zeros(nlev*nods,1);
count=-nods+1;
for n=1:nlev
count=count+nods;
theta(count:count+nods-1)=T(:,n);
end
mt=ones(nods*nlev,1);
mt(((nlev-1)*nods+1):(nlev*nods))=zeros(nods,1);
load EismintT_initial.mat
load EismintH_initial.mat


dummy=logspace(-2,-1,lowlayers);
zetaice=[logspace(-2,-1,lowlayers)-dummy(1)*ones(1,lowlayers)   linspace(.13,1,icelev-lowlayers)  ]';
%zetaice=linspace(0,1,icelev)';
```

```
for n=1:nel
dy(n)=max([abs(y(index(n,1))-y(index(n,2)))   abs(y(index(n,1))-y(index(n,3)))  ...
abs(y(index(n,2))-y(index(n,3)))]);
end
for n=1:nel
dx(n)=max([abs(x(index(n,1))-x(index(n,2)))   abs(x(index(n,1))-x(index(n,3)))  ...
abs(x(index(n,2))-x(index(n,3)))]);
end
dx=dx';
dy=dy';


% element to node interpolation:
value=zeros(nel*9,1);
row=zeros(nel*9,1);
col=zeros(nel*9,1);
count=-nel+1;
for m=1:3
for k=1:3
count=count+nel;
row(count:count+nel-1)=index(:,m);
col(count:count+nel-1)=index(:,k);
value(count:count+nel-1)=area.*(  (m==k)/6 + (m~=k)/12 ).*ones(nel,1);
end
end
El2Nodes=sparse(row,col,value);


nactive=sum( ( mhs==1 ) );
nspec=nods+nshadow-nactive;
rowDI=zeros(nactive,1);
colDI=zeros(nactive,1);
valueDI=zeros(nactive,1);
rowcnt=0;
for n=1:nods+nshadow
if ( mhs(n)==1 )
rowcnt=rowcnt+1;
rowDI(rowcnt)=rowcnt;
colDI(rowcnt)=n;
valueDI(rowcnt)=1;
end
end
PDI=sparse(rowDI,colDI,valueDI,nactive,nods+nshadow);
clear rowDI colDI valueDI

% Let's vectorize the filling of R:


row=zeros(nel*6*6*7,1);
col=ones(nel*6*6*7,1);
Rvalueperm=zeros(nel*6*6*7,1);
count=1-nel;
for m=1:6
for k=1:6
for gauss=1:7
```

```
count=count+nel;
Rvalueperm(count:count+nel-1)=  area(:).* ( GaussWeights(gauss).*  ( (hgamma(:,m)+halpha(:,m).*xg(:,gauss)  ...
 + hbeta(:,m).*yg(:,gauss)  ...
    + hdelta(:,m).*xg(:,gauss).^2  ...
+ hepsilon(:,m).*yg(:,gauss).^2  ...
+ hphi(:,m).*xg(:,gauss).*yg(:,gauss)  ) ...
.* (hgamma(:,k)+halpha(:,k).*xg(:,gauss)  ...
+ hbeta(:,k).*yg(:,gauss)  ...
    + hdelta(:,k).*xg(:,gauss).^2  ...
+ hepsilon(:,k).*yg(:,gauss).^2  ...
+ hphi(:,k).*xg(:,gauss).*yg(:,gauss)  ) )  );

end
end
end

RTvalueperm=zeros(nel*3*3*icelev,1);
count=1-nel;
for lev=1:icelev
for m=1:3
for k=1:3
count=count+nel;
RTvalueperm(count:count+nel-1)=  area(:).*( (m==k)/6 + (m~=k)/12 );
end
end
end



% Let's reduce the computation of filling A:
rowperm=zeros(nel*36*7,1);
colperm=zeros(nel*36*7,1);
valueperm=zeros(nel*36*7,1);
valueperm2=zeros(nel*36*7,1);
count=-nel+1;
for m=1:6
for k=1:6
for gauss=1:7
count=count+nel;
rowperm(count:count+nel-1)=hindex(:,m);
colperm(count:count+nel-1)=hindex(:,k);
valueperm(count:count+nel-1)=area.*1/dt.*  GaussWeights(gauss).*  ( (hgamma(:,m)+halpha(:,m).*xg(:,gauss)  + hbeta(:,
    + hdelta(:,m).*xg(:,gauss).^2  + hepsilon(:,m).*yg(:,gauss).^2  + hphi(:,m).*xg(:,gauss).*yg(:,gauss)  ) ...
.* (hgamma(:,k)+halpha(:,k).*xg(:,gauss)  + hbeta(:,k).*yg(:,gauss)  ...
    + hdelta(:,k).*xg(:,gauss).^2  + hepsilon(:,k).*yg(:,gauss).^2  + hphi(:,k).*xg(:,gauss).*yg(:,gauss)  ) ) ;
valueperm2(count:count+nel-1)=area.*GaussWeights(gauss).*(   (halpha(:,m)  +2*hdelta(:,m).*xg(:,gauss)  + hphi(:,m).*yg
   .* (halpha(:,k)  + 2*hdelta(:,k).*xg(:,gauss)  + hphi(:,k).*yg(:,gauss))  + ...
   (hbeta(:,m)  + 2*hepsilon(:,m).*yg(:,gauss)  + hphi(:,m).*xg(:,gauss))  ...
   .*(hbeta(:,k)  + 2*hepsilon(:,k).*yg(:,gauss)  + hphi(:,k).*xg(:,gauss))  ) ;
end
end
end


% Set up parsing matrix for thermodynamic step:
```

```
nactive=sum(  ( mt(rocklev*nods+1:nlev*nods)==1  ) );
nspec=icelev*nods-nactive;
rowTI=zeros(nactive,1);
colTI=zeros(nactive,1);
valueTI=zeros(nactive,1);
rowcnt=0;
for n=(rocklev*nods+1):(nlev*nods)
if ( mt(n)==1 )
rowcnt=rowcnt+1;
rowTI(rowcnt)=rowcnt;
colTI(rowcnt)=n-rocklev*nods;
valueTI(rowcnt)=1;
end
end
PTI=sparse(rowTI,colTI,valueTI,nactive,icelev*nods);
clear rowTI colTI valueTI




I=zeros(nel,icelev);
udefbar=zeros(nel,icelev);
vdefbar=zeros(nel,icelev);
uslidebar=zeros(nel,1);
vslidebar=zeros(nel,1);
uelnodes=zeros(nel,3);
velnodes=zeros(nel,3);
u=zeros(nods,icelev);
v=zeros(nods,icelev);
div=zeros(nel,icelev);  % calculated from high-order surface parameterization
w=zeros(nel,icelev);
wnodes=zeros(nods,icelev);
upwind=zeros(nel,icelev);
upwindnodes=zeros(nods,icelev);
diffgauss=zeros(nel,7);
Qnodes=zeros(nods,icelev);
diffbarold=zeros(nel,1);

Tbar=zeros(nel,icelev);
Abar=zeros(nel,icelev);
Tpmpbar=zeros(nel,icelev);
THbar=zeros(nel,icelev);
dz=zeros(nel,icelev-1);



% Time-stepping portion of code:

for n=1:icelev
kice(:,n)=2.1*ones(nods,1);
capice(:,n)=2009*ones(nods,1);
capicebar(:,n)=2009*ones(nel,1);
end
```

```
nsteps=200;
massstep=0;
thermstep=1;
advection=1;
for timestep=1:nsteps
[timestep  (timestep-1)*dt/3.1559e7  T(1039,rocklev+1)-Tnot  T(963,rocklev+1)-Tnot  max(s) ]


sbar=zeros(nel,1);
bbar=zeros(nel,1);
sxbar=zeros(nel,1);
sybar=zeros(nel,1);
sxnodes=zeros(nel,3);
synodes=zeros(nel,3);
grads2=zeros(nel,1);
sgauss=zeros(nel,7);
bgauss=zeros(nel,7);
grads2gauss=zeros(nel,7);
grads2nodes=zeros(nel,3);




for gauss=1:7
for m=1:6

sbar(:)=sbar(:)+GaussWeights(gauss).*(hgamma(:,m)+halpha(:,m).*xg(:,gauss)    + hbeta(:,m).*yg(:,gauss)  ...
    + hdelta(:,m).*xg(:,gauss).^2  + hepsilon(:,m).*yg(:,gauss).^2  + hphi(:,m).*xg(:,gauss).*yg(:,gauss)  ) .* hs(hi
bbar(:)=bbar(:)+GaussWeights(gauss).*(hgamma(:,m)+halpha(:,m).*xg(:,gauss)    + hbeta(:,m).*yg(:,gauss)  ...
    + hdelta(:,m).*xg(:,gauss).^2  + hepsilon(:,m).*yg(:,gauss).^2  + hphi(:,m).*xg(:,gauss).*yg(:,gauss)  ) .* hb(hi
sxbar(:)=sxbar(:)+GaussWeights(gauss).*(halpha(:,m)     ...
    + 2*hdelta(:,m).*xg(:,gauss)   + hphi(:,m).*yg(:,gauss)  ) .* hs(hindex(:,m));
sybar(:)=sybar(:)+GaussWeights(gauss).*(  hbeta(:,m)  ...
     + 2*hepsilon(:,m).*yg(:,gauss)  + hphi(:,m).*xg(:,gauss))  .* hs(hindex(:,m));
sgauss(:,gauss)=sgauss(:,gauss)+(hgamma(:,m)+halpha(:,m).*xg(:,gauss)    + hbeta(:,m).*yg(:,gauss)  ...
    + hdelta(:,m).*xg(:,gauss).^2  + hepsilon(:,m).*yg(:,gauss).^2  + hphi(:,m).*xg(:,gauss).*yg(:,gauss)  ) .* hs(hi
bgauss(:,gauss)=bgauss(:,gauss)+(hgamma(:,m)+halpha(:,m).*xg(:,gauss)    + hbeta(:,m).*yg(:,gauss)  ...
    + hdelta(:,m).*xg(:,gauss).^2  + hepsilon(:,m).*yg(:,gauss).^2  + hphi(:,m).*xg(:,gauss).*yg(:,gauss)  ) .* hb(hi

end
end
%for m=1:3
%sxbar(:)=sxbar(:)+s(index(:,m)).*alpha(:,m);
%sybar(:)=sybar(:)+s(index(:,m)).*beta(:,m);
%sbar(:)=sbar(:)+s(index(:,m))/3;
%end
%for l=1:3
%for k=1:3
%grads2(:)=grads2(:)+s(index(:,k)).*s(index(:,l)).*...
%(alpha(:,l).*alpha(:,k)+beta(:,l).*beta(:,k));
%end
%end
```

```
for gauss=1:7
for m=1:6
for k=1:6
grads2(:)=grads2(:)+GaussWeights(gauss).*(   (halpha(:,m)+2*hdelta(:,m).*xg(:,gauss)+hphi(:,m).*yg(:,gauss))    ...
.*(halpha(:,k)+2*hdelta(:,k).*xg(:,gauss)+hphi(:,k).*yg(:,gauss))    ...
+ (hbeta(:,m)+2*hepsilon(:,m).*yg(:,gauss)+hphi(:,m).*xg(:,gauss))    ...
.*(hbeta(:,k)+2*hepsilon(:,k).*yg(:,gauss)+hphi(:,k).*xg(:,gauss))    ).*hs(hindex(:,m)).*hs(hindex(:,k));
grads2gauss(:,gauss)=grads2gauss(:,gauss)+(   (halpha(:,m)+2*hdelta(:,m).*xg(:,gauss)+hphi(:,m).*yg(:,gauss))    ...
.*(halpha(:,k)+2*hdelta(:,k).*xg(:,gauss)+hphi(:,k).*yg(:,gauss))    ...
+ (hbeta(:,m)+2*hepsilon(:,m).*yg(:,gauss)+hphi(:,m).*xg(:,gauss))    ...
.*(hbeta(:,k)+2*hepsilon(:,k).*yg(:,gauss)+hphi(:,k).*xg(:,gauss))    ).*hs(hindex(:,m)).*hs(hindex(:,k));
end
end
end


for lev=1:icelev-1
dz(:,lev)=(sbar-bbar)*(zetaice(lev+1)-zetaice(lev));   % note that this is an average over the element
end



for n=1:icelev
Tbar(:,n)=(T(index(:,1),n+rocklev)+T(index(:,2),n+rocklev)+T(index(:,3),n+rocklev))/3;
Tpmpbar(:,n)=Tnot*ones(nel,1)-rho*g*(sbar-bbar)*(1-zetaice(n))*Phi;
THbar(:,n)=Tbar(:,n)-Tpmpbar(:,n)+Tnot*ones(nel,1);
% Huybrecht's flow law parameterization:
%Abar(:,n)=((1.14e-5/3.1559e7)*(Tbar(:,n)<(263*ones(nel,1)))   + (5.47e10/3.1559e7)*(Tbar(:,n)>=(263*ones(nel,1))))
%                  .*exp( (-6.0e4*(Tbar(:,n)<(263*ones(nel,1)))  - 13.9e4*(Tbar(:,n)>=(263*ones(nel,1))))./(Rgas*THbar
Abar(:,n)=3.1689e-24*ones(nel,1);
end

diffbar=zeros(nel,1);

for n=2:icelev
I(:,n)=E*grads2.*(Abar(:,n-1)+Abar(:,n))/2*((rho*g)^2)   ...
           .*(((sbar-bbar).*(1-zetaice(n-1))).^3  + ((sbar-bbar).*(1-zetaice(n))).^3).*dz(:,n-1)/2   + I(:,n-1);
udefbar(:,n)=    -2*rho*g*I(:,n).*sxbar;
vdefbar(:,n)=    -2*rho*g*I(:,n).*sybar;
%diffbar(:)=diffbar(:)+2*rho*g*(I(:,n-1)+I(:,n))/2.*dz(:,n-1);
end
%diffbar(:)=diffbar(:)+rho*g*(sbar-bbar).^2.*1.5843e-10.*(Tbar(:,1)>=Tpmpbar(:,1));
diffbar(:)=grads2.*sbar.^5*3.169e-24*(rho*g)^3*2/5;    % Use this expression for EISMINT exact analytical test


diffnodes=zeros(nods,1);
totarea=zeros(nods,1);
for n=1:nel
totarea(index(n,:))=totarea(index(n,:))+area(n);
end
for n=1:nel
diffnodes(index(n,:))=diffnodes(index(n,:))+diffbar(n)*area(n);
end
diffnodes(:)=diffnodes(:)./totarea;
```

```
%if timestep>1
%diffbar=.1*diffbar+.9*diffbarold;
%end

%convergence=[mean(diffbar)  mean(diffbarold)  max(diffbar)  max(diffbarold)]
%diffbarold=diffbar;


Qbar=zeros(nel,icelev);
for n=1:icelev-1
Qbar(:,n)=2*(rho*g)^4*E*Abar(:,n).*grads2.^2.*(sbar*(1-zetaice(n))).^4;    % check the leading 2
end
Qbar(:,1)=Qbar(:,1)-(uslidebar(:).*sxbar   + vslidebar(:).*sybar).*(sbar-bbar)*rho*g;
Qnodes=zeros(nods,icelev);
totarea=zeros(nods,1);
for n=1:nel
totarea(index(n,:))=totarea(index(n,:))+area(n);
end
for lev=1:icelev-1
for n=1:nel
Qnodes(index(n,:),lev)=Qnodes(index(n,:),lev)+Qbar(n,lev)*area(n);
end
Qnodes(:,lev)=Qnodes(:,lev)./totarea;
end



%melted= ( (T(:,rocklev+1)>=(Tnot*ones(nods,1)-rho*g*(s-b)*Phi))   & (Bdot>=zeros(nods,1))  );
melted=(T(:,rocklev+1)>=(Tnot*ones(nods,1)-rho*g*(s-b)*Phi));
heatin=-krock*(T(:,rocklev+1)-T(:,rocklev))/dzrock;
heatout=-(kice(:,2)+kice(:,1))./2.*(T(:,rocklev+2)-T(:,rocklev+1))./(    (s(:)-b(:))*(zetaice(2)-zetaice(1))  );
Bdot=((heatin-heatout)/rho/fusion).*melted;
mb=-Bdot;
mabar(:)=(Adot(index(:,1))+Adot(index(:,2))+Adot(index(:,3)))/3;
mbbar(:)=-(Bdot(index(:,1))+Bdot(index(:,2))+Bdot(index(:,3)))/3;    % Bdot is a melt rate, mbbar is a freeze rate


for n=1:icelev
div(:,n)=2*( hs(hindex(:,1)).*hdelta(:,1)+  ...
                  hs(hindex(:,2)).*hdelta(:,2)+  ...
   hs(hindex(:,3)).*hdelta(:,3)+  ...
   hs(hindex(:,4)).*hdelta(:,4)+  ...
   hs(hindex(:,5)).*hdelta(:,5)+  ...
   hs(hindex(:,6)).*hdelta(:,6)+  ...
   hs(hindex(:,1)).*hepsilon(:,1)+  ...
                  hs(hindex(:,2)).*hepsilon(:,2)+  ...
   hs(hindex(:,3)).*hepsilon(:,3)+  ...
   hs(hindex(:,4)).*hepsilon(:,4)+  ...
   hs(hindex(:,5)).*hepsilon(:,5)+  ...
   hs(hindex(:,6)).*hepsilon(:,6))  ...
   .* (-2*rho*g*I(:,n)   ) ;
end
```

```
%diffgauss=zeros(nel,7);
%for gauss=1:7
%for n=2:icelev  % Warning: not adjusted for FEM vertical spacing yet
%I(:,n)=E*grads2gauss(:,gauss).*(Abar(:,n-1)+Abar(:,n))/2*((rho*g)^2)    ...
%           .*((sgauss(:,gauss)-bgauss(:,gauss)-dzgauss(:,gauss)*(n-2)).^3   + (sbar(:,gauss)-bbar(:,gauss)-dzgauss(:,
%diffgauss(:,gauss)=diffbargauss(:,gauss)+2*rho*g*(I(:,n-1)+I(:,n))/2.*dzgauss(:,gauss);
%end
%diffgauss(:,gauss)=diffbargauss(:,gauss)+rho*g*(sgauss(:,gauss)-bgauss(:,gauss)).^2.*1.5843e-10.*(Tbar(:,1)>=Tpmpbar(
%end
%for gauss=1:7
%diffgauss(:,gauss)=grads2gauss(:,gauss).*sgauss(:,gauss).^5*3.169e-24.*ones(nel,1)*(rho*g)^3*2/5;
%end
% The I calculated above is for gauss points.
for gauss=1:7
diffgauss(:,gauss)=diffnodes(index(:,1)).*(gamma(:,1)+alpha(:,1).*xg(:,gauss)+beta(:,1).*yg(:,gauss))     ...
 + diffnodes(index(:,2)).*(gamma(:,2)+alpha(:,2).*xg(:,gauss)+beta(:,2).*yg(:,gauss))    ...
 + diffnodes(index(:,3)).*(gamma(:,3)+alpha(:,3).*xg(:,gauss)+beta(:,3).*yg(:,gauss))    ;
end


% Computation of the "D-term"

Intdiv=zeros(nel,icelev);
Dterm=zeros(nel,icelev);
Vertvel=zeros(nel,icelev);
for n=2:icelev
Intdiv(:,n)=Intdiv(:,n-1)+(div(:,n-1)+div(:,n))/2*(zetaice(n)-zetaice(n-1));
Vertvel(:,n)=Vertvel(:,n-1)-(div(:,n-1)+div(:,n))/2.*dz(:,n-1);
end


w=zeros(nods,icelev);
upwind=zeros(nods,icelev);

for n=2:icelev
Dterm(:,n)=(sbar-bbar).*(  zetaice(n).*Intdiv(:,icelev)  - Intdiv(:,n) ); % ERROR found here!
end

Dtermnodes=zeros(nods,icelev);
totarea=zeros(nods,1);

for n=1:nel
totarea(index(n,:))=totarea(index(n,:))+area(n);
end
for lev=1:icelev
for n=1:nel
Dtermnodes(index(n,:),lev)=Dtermnodes(index(n,:),lev)+Dterm(n,lev)*area(n);
end
Dtermnodes(:,lev)=Dtermnodes(:,lev)./totarea;
end



for n=1:icelev
w(:,n)=(Dtermnodes(:,n)  - zetaice(n)*Adot  + 0*(1-zetaice(n))*Bdot);   % Error found here
```

```
upwind(:,n)=(w(:,n)>0);  % defines a logical variable to determine upwinding scheme, if used
end




if massstep

value=zeros(nel*36*7,1);
count=-nel+1;
for m=1:6
for k=1:6
for gauss=1:7
count=count+nel;
value(count:count+nel-1)=valueperm(count:count+nel-1)+diffgauss(:,gauss).*valueperm2(count:count+nel-1);
end
end
end
A=sparse(rowperm,colperm,value,nods+nshadow,nods+nshadow);
clear value

Aprime=PDI*A*PDI';



value=zeros(nel*6*6*7,1);
count=1-nel;
for m=1:6
for k=1:6
for gauss=1:7
count=count+nel;
value(count:count+nel-1)=  (hs(hindex(:,k))/dt+0.3/31556926).*  Rvalueperm(count:count+nel-1);
end
end
end
R=sparse(rowperm,ones(nel*36*7,1),value,nods+nshadow,1);


Rprime=PDI*(R-A*(hs.*(~mhs)));   % remember to have current boundary conditions specified in theta
hsprime=Aprime\Rprime;
hs=PDI'*hsprime+hs.*(~mhs);
s=hs(1:nods);


end

% Thermodynamic time step

if thermstep


if advection

ncycles=9*(icelev-1)*nel;
```

```
row=zeros(ncycles,1);
col=zeros(ncycles,1);
value=zeros(ncycles,1);
count=-nel+1;



for lev=1:(icelev-1)
for m=1:3
for k=1:3
count=count+nel;
row(count:count+nel-1)=(lev-1)*nods*ones(nel,1)+index(:,m);
col(count:count+nel-1)=(lev-1)*nods*ones(nel,1)+index(:,k);
value(count:count+nel-1)=dt*area.*(  ( (m==k)/6 + (m~=k)/12 )/dt ...
+ abs(udefbar(:,lev)  +uslidebar(:)).*dx(:).*alpha(:,m).*alpha(:,k)/2   ...
+ abs(vdefbar(:,lev)  +vslidebar(:)).*dy(:).*beta(:,m).*beta(:,k)/2    ...
+ (udefbar(:,lev)/3  +uslidebar(:)/3).*alpha(:,k)   ...
+  (vdefbar(:,lev)/3  +vslidebar(:)/3).*beta(:,k)  );
end
end
end


AT=sparse(row,col,value,icelev*nods,icelev*nods);
ATprime=PTI*AT*PTI';

row=zeros(nel*9*icelev,1);
col=ones(nel*9*icelev,1);
value=zeros(nel*9*icelev,1);
count=1-nel;
for lev=1:icelev
for m=1:3
for k=1:3
count=count+nel;
row(count:count+nel-1)=(lev-1)*nods+index(:,m);
value(count:count+nel-1)=  RTvalueperm(count:count+nel-1).*theta(rocklev*nods+(lev-1)*nods+index(:,k));
end
end
end

R=sparse(row,ones(nel*9*icelev,1),value,icelev*nods,1);

Rprime=PTI*(R-AT*(theta(rocklev*nods+1:nlev*nods).*(~mt(rocklev*nods+1:nlev*nods))));     % remember to have current
thetaprime=ATprime\Rprime;
theta(rocklev*nods+1:nlev*nods)=PTI'*thetaprime+theta(rocklev*nods+1:nlev*nods).*(~mt(rocklev*nods+1:nlev*nods));
count=rocklev*nods-nods+1;
for n=rocklev+1:nlev
count=count+nods;
T(:,n)=theta(count:count+nods-1);
end

end % end advection part of code

% Now for vertical part of the thermal time step:
```

```
% Begin tridiagonal part of code:

for nod=1:nods
dzice=(s(nod)-b(nod))/(icelev-1);

zeta=[linspace(-rockscale/(s(nod)-b(nod)),-rockscale/(s(nod)-b(nod))/rocklev,rocklev)    zetaice']';
al=zeros(nvel,2);
dl=zeros(nvel,1);
for n=1:nvel
X=[zeta(n)  1
zeta(n+1)  1];
phi=inv(X);
al(n,:)=phi(1,:);
dl(n)=zeta(n+1)-zeta(n);
end



K=zeros(nvel,2);
K(:,1)=[krock*ones(rocklev,1)
kice(nod,1:icelev-1)'];
K(:,2)=[krock*ones(rocklev-1,1)
kice(nod,1:icelev)'];
K(rocklev,:)=[krock  krock];

invcaprho=zeros(nvel,2);
invcaprho(:,1)=[1./rhorock./caprock.*ones(rocklev,1)
1./rho./capice(nod,1:icelev-1)'];
invcaprho(:,2)=[1./rhorock./caprock.*ones(rocklev-1,1)
1./rho./capice(nod,1:icelev)'];
invcaprho(rocklev,:)=[1/rhorock/caprock   1/rhorock/caprock];


W=[zeros(1,rocklev)  w(nod,:)];
W=[W(1:nvel)
W(2:nlev)]';
Q=[zeros(1,rocklev)  Qnodes(nod,:)./rho./capice(nod,:)];
Q=[Q(1:nvel)
Q(2:nlev)]';
Q(rocklev,:)=[0  0];
W(rocklev,:)=[0  0];



rowline=zeros(nvel*4,1);
colline=zeros(nvel*4,1);
value=zeros(nvel*4,1);
count=-nvel+1;
for i=1:2
for j=1:2
count=count+nvel;
rowline(count:count+nvel-1)=lindex(:,i);
colline(count:count+nvel-1)=lindex(:,j);
```

```
value(count:count+nvel-1)=dl.*(  ((i==j)/3 + (i~=j)/6)/dt*ones(nvel,1)  + ...
 (W(:,1).*( (i==1)/3 + (i~=1)/6 ).*al(:,j)  ...
+ W(:,2).*( (i==2)/3 + (i~=2)/6 ).*al(:,j) )/(s(nod)-b(nod)) );
for l=1:2
for k=1:2
value(count:count+nvel-1)=value(count:count+nvel-1)+   ...
dl.*( K(:,l).*invcaprho(:,k).*al(:,i).*al(:,j)/(s(nod)-b(nod)).^2.*(   (l==k)/3 + (l~=k)/6)  ...
+ K(:,l).*invcaprho(:,k).*al(:,k).*al(:,j)/(s(nod)-b(nod)).^2.*(   (l==i)/3 + (l~=i)/6) );
end
end
end
end

A=sparse(rowline,colline,value);
A(nlev,nlev)=1;
A(nlev,nlev-1)=0;
A(1,1)=-1;
A(1,2)=1;
A(rocklev+1,rocklev)=0;
A(rocklev+1,rocklev+1)=-1;
A(rocklev+1,rocklev+2)=1;
if (melted(nod))
A(rocklev+1,rocklev)=0;
A(rocklev+1,rocklev+1)=1;
A(rocklev+1,rocklev+2)=0;
end


count=-nvel+1;
for i=1:2
for j=1:2
count=count+nvel;
value(count:count+nvel-1)=dl.*(  T(nod,lindex(:,j))'.*((i==j)/3  + (i~=j)/6)/dt + ...
 Q(:,j).*( (i==j)/3 + (i~=j)/6 )  );
end
end

R=sparse(rowline,ones(nvel*4,1),value);

R(1)=-Geoflux/krock*dzrock;
R(nlev)=T(nod,nlev);
R(rocklev+1)=-Geoflux/kice(nod,1)*(s(nod)-b(nod))*(zetaice(2)-zetaice(1));
if (melted(nod))
R(rocklev+1)=Tnot-rho*g*(s(nod)-b(nod))*Phi;
end

T(nod,:)=(A\R)';

end  % end loop over nods.

% reconstruct  theta:
count=0;
for lev=1:nlev
for n=1:nods
count=count+1;
```

```
theta(count)=T(n,lev);
end
end

end % end thermal part



end % end time step

%diagnostics
veldefbar=sqrt(  (uslidebar(:)+udefbar(:,icelev)).^2  + (vslidebar(:)+vdefbar(:,icelev)).^2);

figure(1)
clg
axis([-755e3  755e3 -755e3 755e3]),axis('image'),shading  flat,hold on,colormap('jet')
for n=1:nel
fill(x(index(n,:)),y(index(n,:)),T(index(n,:),rocklev+1))
end
for n=1:nods
if (ms(n)==0)
plot(x(n),y(n),'wo')
end
end
title('basal  temperature')
plot(x(216),y(216),'bo')
plot(x(215),y(215),'ro')

figure(2)
clg
subplot(1,2,1)
axis([-750e3  750e3 -750e3 750e3]),axis('image'),shading  flat,hold on,colormap('jet')
for n=1:nel
fill(x(index(n,:)),y(index(n,:)),s(index(n,:)))
end
title('surface')
subplot(1,2,2)
axis([-750e3  750e3 -750e3 750e3]),axis('image'),shading  flat,hold on, %,caxis([-1  2])
for n=1:nel
%fill(x(index(n,:)),y(index(n,:)),melted(index(n,:)))
fill(x(index(n,:)),y(index(n,:)),diffnodes(index(n,:)))
%fill(x(index(n,:)),y(index(n,:)),diffbar(n))
end
title('diffbar')



figure(3)
clg
subplot(4,1,1)
axis([0 max(radius)  -100 max(s)]), hold on
plot(radius,s,'r+')
plot(radius,b,'b.')
title('surface  profile')
```

```
xlabel('radius')
ylabel('thickness')
subplot(4,1,2)
axis([0 800e3 0 3.1559e7*max(veldefbar)]),  hold on
plot((radius(index(:,1))+radius(index(:,2))+radius(index(:,3)))/3,3.1559e7*veldefbar(:),'ro')
title('surface x-velocity (elemental)')
subplot(4,1,3)
axis([0 max(radius) min(div(:,icelev))/1  max(div(:,icelev))/1]),  hold on
%plot((radius(index(:,1))+radius(index(:,2))+radius(index(:,3)))/3,3.1559e7*sqrt(udefbar(:,icelev).^2+vdefbar(:,icelev
%title('surface velocity (elemental)')
plot((radius(index(:,1))+radius(index(:,2))+radius(index(:,3)))/3,div(:,icelev),'ro')
title('surface divergence (elemental)')
subplot(4,1,4)
axis([0 max(radius) min(T(:,rocklev+1))  max(T(:,rocklev+1))]),  hold on
plot(radius,T(:,rocklev+1),'r+')
title(' base temperature')
%axis([0 max(radius) min(T(:,nlev))  max(T(:,nlev))]),  hold on
%plot(radius,T(:,nlev),'r+')
%title(' surface temperature')


figure(4)
clg
subplot(3,2,1)
axis([0 31556926*veldefbar(1773)  0 sbar(1773)]),hold  on
plot(31556926*sqrt((uslidebar(1773)+udefbar(1773,:)).^2   +(vslidebar(1773)+vdefbar(1773,:)).^2),zetaice'*sbar(1773),'
plot(31556926*sqrt((uslidebar(1773)+udefbar(1773,:)).^2   +(vslidebar(1773)+vdefbar(1773,:)).^2),zetaice'*sbar(1773),'
title('horizontal  velocity at midpoint')
subplot(3,2,2)
axis([31556926*min(Vertvel(1928,:))  31556926*max(Vertvel(1928,:))  0 sbar(1928)]),hold  on
plot(31556926*Vertvel(1928,:),zetaice'*sbar(1928),'w-')
plot(31556926*Vertvel(1928,:),zetaice'*sbar(1928),'wo')
title('vertical  velocity in z-coordinate at divide element')
%axis([31556926*min(w(1039,:))  31556926*max(w(1039,:))  0 s(1039)]),hold  on
%plot(31556926*w(1039,:),linspace(0,s(1039),icelev),'w-')
%title('vertical  velocity in z-coordinate at divide node')
subplot(3,2,3)
axis([min(T(1039,:))  max(T(1039,:))  -rockscale s(1039)]),hold  on
plot(T(1039,:),[ linspace(-rockscale,-rockscale/rocklev,rocklev)   zetaice'*s(1039)],'w-')
plot(T(1039,:),[ linspace(-rockscale,-rockscale/rocklev,rocklev)   zetaice'*s(1039)],'wo')
title('temperature  at ice divide')
subplot(3,2,4)
axis([min((T(974,:)+T(974,:))/2)  max((T(974,:)+T(974,:))/2)  -50 (s(974)+s(974))/2]),hold  on
plot((T(974,:)+T(974,:))/2,[  linspace(-rockscale,-rockscale/rocklev,rocklev)   zetaice'*s(974)],'w-')
plot((T(974,:)+T(974,:))/2,[  linspace(-rockscale,-rockscale/rocklev,rocklev)   zetaice'*s(974)],'wo')
title('temperature  at midpoint')
subplot(3,2,5)
%axis([31556926*min(Vertvel(1773,:))  31556926*max(Vertvel(1773,:))  0 sbar(1773)]),hold  on
%plot(31556926*Vertvel(1773,:),linspace(0,sbar(1773),icelev),'w-')
%title('vertical  velocity in z-coordinate at midpoint element')
axis([31556926*min(w(963,:))  31556926*max(w(963,:))  0 s(963)]),hold  on
plot(31556926*w(963,:),zetaice'*s(963),'w-')
plot(31556926*w(963,:),zetaice'*s(963),'wo')
title('vertical  velocity in zeta-coordinate  at divide node')
subplot(3,2,6)
```

```
axis([min(Dterm(1940,:))  max(Dterm(1940,:))  0 sbar(1940)]),hold  on
plot(Dterm(1940,:),zetaice*sbar(1940),'w-')
plot(Dterm(1940,:),zetaice*sbar(1940),'wo')
title('Dterm  at divide m/s')

for n=rocklev+1:nlev
Tp(:,n-rocklev)=Tnot*ones(nods,1)-rho*g*(s-b).*(1-zetaice(n-rocklev))*Phi;
TH(:,n-rocklev)=T(:,n)-Tp(:,n-rocklev);
%TH(:,n-rocklev)=T(:,n)-Tnot*ones(nods,1);
end

figure(5)
clg
subplot(1,2,1)
axis([min(TH(962,:))  max(TH(962,:))+20  0 s(962)]),hold  on
plot(TH(962,:),zetaice'*s(962),'y-')
plot(TH(962,:),zetaice'*s(962),'ro')
tmelt=-rho*g*s(962)*Phi;
plot([tmelt  tmelt],[-rockscale  (s(962)+s(962))/2],'g-')
plot([min((TH(962,:)+TH(962,:))/2)  max((TH(962,:)+TH(962,:))/2)],[0  0],'g-')
title('temperature  at midpoint')
subplot(1,2,2)
axis([min(TH(1039,:))  max(TH(1039,:))+20  0  s(1039)]),hold  on
plot(TH(1039,:),zetaice'*s(1039),'y-')
plot(TH(1039,:),zetaice'*s(1039),'ro')
tmelt=-rho*g*s(1039)*Phi;
plot([tmelt  tmelt],[-rockscale  s(1039)],'g-')
plot([min(TH(1039,:))  max(TH(1039,:))],[0  0],'g-')
title('temperature  at divide')
```

Chapter 10

Ice Sheet Thermodynamics: Mixed (Cold and Temperate) Ice Conditions

In circumstances where the strain heating term W is zero, ice temperature can reach a maximum only at the upper or lower boundary. The pressure-melting temperature, if attained anywhere, is attained only at the base of the ice sheet or glacier (or surface, in warm climates). With strain heating present, a temperate (melted) ice layer of finite thickness can be developed. When this happens, the ice sheet or glacier is said to be *polythermal* (*i.e.,* of mixed cold and temperate constitution). Modelling the thermal evolution of a polythermal ice sheet or glacier is complicated because boundary conditions at the cold/temperate ice transition(s) can offer practical difficulties. In some circumstances (when ice from the cold side moves through the transition to the temperate side), for example, both a temperature and a temperature gradient are specified at the cold/temperate ice transition.

In this chapter, techniques are described for predicting the evolution of certain, restricted classes of polythermal ice sheets. In particular, we shall develop a 1-dimensional (vertical heat transfer only) ice-column model of a polythermal ice sheet (with appropriate simplifying assumptions) to illustrate

the methods for predicting the development of a basal temperate-ice layer in circumstances where the upper boundary of the temperate-ice layer constitutes the only cold/temperate transition point. (Circumstances in which the temperate-ice layer has a lower boundary above the base of the ice column are not considered.) The techniques illustrated in the 1-dimensional model developed here are intended to be readily adapted to the 3-dimensional thermodynamic model discussed in the previous chapter.

Excellent treatments of the theoretical and numerical analysis of polythermal ice conditions are provided by Fowler [1984], Hutter and others [1988] and Greve [1995]. A numerical analysis of observed polythermal ice conditions in the Jacobshavns Isbrae, Greenland, is provided by Funk and others [1994]. The development presented here is based on these previous studies and differs only in several technical aspects (*e.g.,* here we use a finite-element formulation of vertical heat flow, a stretched vertical coordinate that spans the cold-ice layer, and a control method to determine the cold-ice to temperate-ice transition rate in circumstances when ice movement through the cold/temperate ice transition is from the cold side to the melted side).

10.1 A Simple Illustrative Problem

To describe the governing equations in a manner that is complementary to the comprehensive work of Hutter and others [1988], we shall adopt a time-evolution point of view and focus on temporal transition points when the governing equations and their boundary conditions shift from one form to another. We begin by setting up a simple (idealized) 1-dimensional, initial-value problem for heat flow in a single ice column (which, as mentioned previously, may be modified for inclusion within a more comprehensive, 3-dimensional ice sheet).

Ice-column geometry is depicted in Figure (10.1). The vertical coordinate, z, ranges from the bed, at $z = b$, to the surface, at $z = s$. Both s and b are assumed to be constant, *i.e.,* the vertical strain rate is assumed to compensate for snow accumulation and basal melting so as to simplify the ice-column geometry. If a cold/temperate ice transition exists, its location is $z = c(t)$.

Cold ice is assumed to exist above $z = c(t)$ and temperate ice is assumed to exist below $z = c(t)$. More complicated geometries, where the temperate ice is confined between two (or more) levels above the bed, is not considered in this chapter. (A temperate ice layer could occur within $c_l(t) \leq z \leq c_u(t)$ if, for example, thermal inertia of the bedrock below the ice keeps a thin zone of basal ice below the melting temperature while strain heating warms ice above to the melting point.) Following convention, a stretched vertical coordinate, ζ which spans the ice column is defined:

$$\zeta = \frac{z - b}{s - b} \tag{10.1}$$

When $c(t) \neq b$, a second stretched vertical coordinate which spans the cold-ice layer is defined:

$$\zeta_c = \frac{z - c(t)}{s - c(t)} \tag{10.2}$$

as depicted in Figure (10.1). Below the ice, within the region $r \leq z \leq b$, a bedrock layer of fixed thickness $(b - r)$ is considered. All notation, unless otherwise stated, is the same as that used in Chapter 9.

The temperature field, $T(z, t)$, and the liquid-water fraction $\mu(z, t)$ (defined as volume of water per unit volume of ice/water mixture), within the ice and bedrock region, $r \leq z \leq s$, is to be determined for $t \geq 0$, where t is time. Boundary conditions are applied at $z = s$ and $z = r$:

$$T(s, t) = T_s < 0 \tag{10.3}$$

$$T_z(r, t) = \frac{-G}{k_r} \tag{10.4}$$

where G is the geothermal flux, k_r is the thermal conductivity of bedrock, and the subscript z denotes partial differentiation with respect to z. Initial conditions are isothermal below the melting point:

$$T(z, 0) = T_s \tag{10.5}$$

$$\mu(z, 0) = 0 \tag{10.6}$$

for the region $r \leq z \leq s$.

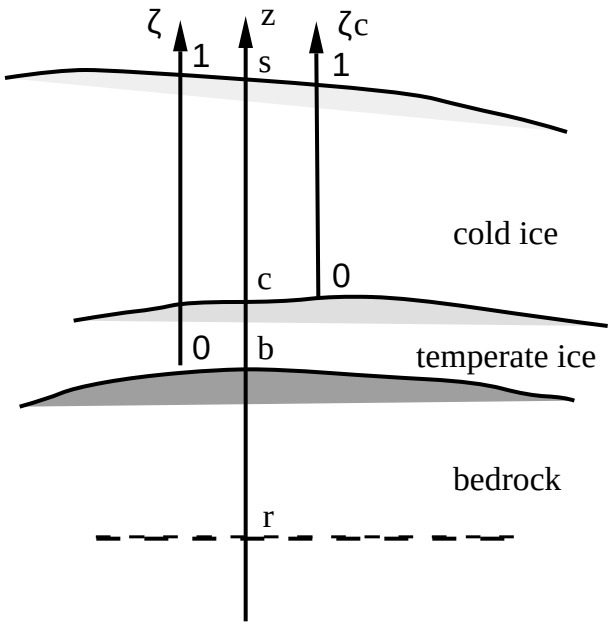

Figure 10.1: Idealized ice-column geometry. Stretched vertical coordinates ζ and ζ_c span the ice column and the cold-ice layer, respectively.

To simplify the illustration of polythermal ice modelling techniques, we shall adopt the following representation of the vertical ice velocity and strain heating:

$$w(\zeta, t) = -\dot{A}\zeta - \dot{B}(1 - \zeta) \tag{10.7}$$

$$W = \begin{cases} \frac{G}{20} \exp(-5\zeta) & \text{if } t < t_h \\ 0 & \text{if } t \geq t_h \end{cases} \tag{10.8}$$

for $b \leq z \leq s$, and where $\dot{A}$ is the snow accumulation rate (assumed positive), $\dot{B}$ is the basal melting rate (positive for *melting*, negative for *freezing*, in meters of ice equivalent), and where the constants 20 and 5 appearing in Eqn. (10.8) are chosen arbitrarily to produce a reasonable example of strain heating (scaled to the geothermal flux) which decays upward from the bed. The strain-heating function W is chosen to permit the development and upward growth of a temperate ice layer during the time prior to $t = t_h$. The strain-heating cutoff time, t_h, is chosen to permit the temperate ice layer to decay and eventually disappear when $t > t_h$. An additional simplification useful for the present illustration is that ice thickness, $h = (s - b)$, is assumed constant (*i.e.,* vertical strain adjusts with changes in $\dot{A}$ and $\dot{B}$).

Constants used in the simple, illustrative problem are listed as follows:

parameter	value
$\dot{A}$	0.3 m a^{-1}
k	2.1 W m^{-1} K^{-1}
c	2009 J kg^{-1} K^{-1}
T_o	273.15 K
Φ	8.7×10^{-4} K m^{-1}
G	42 mW m^{-2}
31556926	s a^{-1}
r	10 m
t_h	1500 a

Other constants not listed above are the same as those listed in the previous chapter describing the EISMINT Level 1 experiment.

10.2 Conceptual Description of Polythermal Temperature Evolution

The goal is to determine the effects of down-gradient heat conduction (advection is not emphasized here, even though it is an equally important process) and strain heating on the temperature and liquid water content of the ice column. For cold ice, a heat diffusion equation is used to determine the evolution of $T(z,t)$. Analysis and numerical techniques associated with heat diffusion in cold ice is presented in previous chapters, so will not be discussed further here.

For temperate ice, $T(z,t)$ is not a freely evolving variable, but rather is a constrained variable. The pressure-melting condition defines temperate ice, $T(z,t) = T_m(z; s(t), \rho(z,t))$ (note: variables appearing after the semicolon denote parameters which implicitly determine the pressure at level z and thus the pressure-melting temperature). Without freedom to vary $T(z,t)$ in the temperate ice layer, heat produced by strain heating is used to melt ice and create a liquid water content (typically, on the order of several percent, see Hutter and others, 1988). The liquid water content is denoted by the variable $\mu(z,t)$. This variable is taken here to be an "ice-equivalent" volume mixing ratio, and is nondimensional. For example, $\mu = 0.01$ indicates that 1% of the ice in a given cubic meter is melted. Ice with $\mu > 0$ possesses a reservior of latent heat given by $\rho L_f \mu$, where ρ is ice density and L_f is the latent heat of fusion for a kilogram of water.

As discussed by Hutter and others [1988], the evolution of $\mu(z,t)$ can be a advective/diffusive process depending on how rigorously water movement within the veins and grain boundaries of the ice matrix is handled. Hutter and others show that a diffusive (D'Arcyian) scheme leads to a profile of $\mu(z,t)$ that can include narrow boundary layers at the cold/temperate ice transition. Greve [1995] points out that, under some circumstances, "drainage functions" must be invoked to prevent the water content from becoming too large, *e.g.,* $\mu = 1$, in some local regions. Relatively little is known about how to formulate the evolution of $\mu(z,t)$ because of the lack of observation of temperate ice water content.

In the present chapter, we shall adopt a simple treatment of liquid water content that emphasizes an accounting of integral (bulk) properties of the temperate layer. We shall assume that liquid water is well-distributed within the temperate layer (*i.e.,* that there is an infinite diffusivity of liquid water fraction) and that its evolution is determined only by the integral of the heat produced by strain heating within the temperate layer:

$$\mu_t = \frac{1}{c-b} \int_c^b \frac{W}{\rho L_f} dz \tag{10.9}$$

To safeguard against the possibility that μ will become too large (*e.g.,* $\mu > 0.1$), we shall introduce a drainage function (following Greve, 1995) to maintain $\mu \leq 0.1$.

To summarize the conceptual dynamics, heat produced by strain heating is used in one of two ways depending on which layer is being considered. In the cold layer, heat is used to maintain or boost the local temperature. In the temperate layer, heat is used to convert ice to liquid water which is held within the ice matrix. The trickiest part of polythermal ice temperature modelling concerns the thermodynamic conditions that must be met at the boundary between cold and temperate ice. These conditions are described next.

10.2.1 The tricky part: cold/temperate transition boundary conditions

Two possibilities exist at the cold/temperate ice transition: ice could be moving through the transition boundary into or out of the temperate layer.

- **Cold ice flow into temperate zone**. When ice moves into the temperate layer (*i.e.,* the cold/temperate transition is expanding into the cold ice layer), continuity of conductive heat flow requires that the temperature gradient T_z be continuous. Temperate ice temperature is constrained at the pressure melting point and this pressure melting temperature changes with z due to the glaciostatic pressure gradient

ρg. Thus, T must equal $-\rho g(s-z)\Phi$ and T_z must equal $\rho g \Phi$ at the cold/temperate transition.

- **Temperate ice flow into cold zone**. When ice moves from the temperate zone into the cold zone (*i.e.,* the cold/temperate transition is contracting into the temperate layer), conductive heat flow need not be continuous. As ice moves from the temperate zone into the cold zone, its water content μ must go to zero. The latent heat released by the process of freezing the water contained within the ice moving through the cold/temperate transition must be conducted away by an appropriate discontinuity of the temperature gradient T_z in the cold layer. As described by Hutter and others [1988], heat flow continuity at the cold/temperate transition is reflected in the following balance:

$$-k(\rho g \Phi) + \rho L_f \mu a_\perp = -k T_z|_+ \tag{10.10}$$

The two terms on the left-hand side of the above equation represent heat flowing into the cold/temperate transition from the temperate layer (assuming $a_\perp$, the ice flow through the transition, is positive for flow from the temperate layer to the cold layer). The first term is heat conduction associated with the pressure-melting temperature gradient. The second term is latent heat release associated with freezing of liquid water content as ice moves through the transition. The term on the right-hand side of the above equation represents heat conduction away from the transition on the cold side (assumed to be above the temperate side, thus denoted by $|_+$). Given that

$$T_z|_- = \rho g \Phi \tag{10.11}$$

the discontinuity in the temperature gradient is given by

$$T_z|_+ - T_z|_- = \frac{-\rho L_f \mu}{k} a_\perp \tag{10.12}$$

The size of the jump in temperature gradient is determined by the rate at which ice moves through the transition and by the amount of liquid water contained in the temperate ice that moves through the transition.

What is tricky about the above conditions? Well, actually, the second possibility (temperate ice flow into cold zone) isn't tricky at all. The $T_z|_-$

and $T_z|+$ are constrained by the Clapyron slope or determined by the heat equation, repectively. The μ is determined by the heat equation in the temperate layer (with associated water-movement dynamics, if applicable). The only unconstrained variable is $a_\perp$, the velocity of ice movement relative to the cold/temperate transition. (Note that by convention, $a_\perp$ is negative for circumstances when cold ice moves through the transition and becomes temperate ice. This way of defining $a_\perp$ is opposite to the definition of basal melting, $\dot{B}$.) The discontinuity expressed in Equation (10.12) provides the means to determine $a_\perp$:

$$a_\perp = \frac{-k\left(T_z|_+ - T_z|_-\right)}{\rho L_f \mu} \tag{10.13}$$

The above expression is similar to the expression used to determine the basal melting rate (with the absence of a interfacial heat source associated with basal sliding). Observe that $a_\perp$ is greater than (or equal to) zero.

The tricky part comes with the first possibility (cold ice flow into temperate zone). In this circumstance, a jump in temperature gradient cannot be supported because movement through the cold/temperate ice transition does not invoke a release or consumption of latent heat. The twin demands imposed by temperature gradient continuity and the pressure-melting point effect an overdetermination of the boundary conditions normally required to solve the heat equation in the cold-ice layer above the cold/temperate transition. The problem of computing $T(z,t)$ in the cold layer comes down to solving an overdetermined boundary value problem similar to the following:

$$T_t + \mathcal{A}(z; a_\perp) = \frac{1}{\rho c}\left(kT_z\right)_z + \frac{W}{\rho c} \qquad \text{for} \quad c < z < s \tag{10.14}$$

$$T(z = s) = T_s \tag{10.15}$$

$$\left\{ \begin{array}{c} T(z = c) = T_m \\ T_z(z = c) = \rho g \Phi \end{array} \right\} \tag{10.16}$$

where $\mathcal{A}(z; a_\perp)$ is an advection term that depends on the velocity of ice movement through the cold/temperate ice transition. What makes it possible to solve the above overdetermined boundary value problem is the fact that $a_\perp < 0$ is *not determined*. In effect, the overdetermination of boundary conditions at $z = c$ allows $a_\perp$ to be determined.

Greve [1995] and Funk and others [1994] describe analytical and numerical techniques for determining $a_\perp$ to satisfy the overdetermination of boundary conditions at $z = c$. The treatment we shall take is slightly different and follows from control theory (see MacAyeal and others, 1991). We formulate the problem as follows: We seek a value of $a_\perp$ to minimize a performance index, J, given by

$$J = \left(T(z = c) - T_m(c)\right)^2 \tag{10.17}$$

subject to the constraint that the temperature at the cold/temperate transition, $T(z = c)$, is a solution of the following well-determined boundary value problem:

$$T_t + \mathcal{A}(z; a_\perp) = \frac{1}{\rho c}\left(kT_z\right)_z + \frac{W}{\rho c} \qquad \text{for} \quad c < z < s \tag{10.18}$$

$$T(z = s) = T_s \tag{10.19}$$

$$T_z(z = c) = \rho g \Phi \tag{10.20}$$

The variable $a_\perp$ is referred to as the control variable. Algorithms for solving this control problem are numerous (see, for example, MacAyeal and others, 1991). We shall employ a rather simple algorithm for use in this chapter that is based on MATLAB 's optimization toolbox.

The upshot of the control method formulation of a solution to the overdetermined boundary value problem (case when $a_\perp < 0$) is that an iterative numerical proceedure is invoked to choose the appropriate $a_\perp$ so as to make the two boundary conditions at $z = c$ compatible. Greve [1995] and Funk and others [1994] use an iterative numerical technique to choose the location (time evolution), $z = c(t)$, of the cold/temperate ice transition. Our technique is virtually the same, except we use the numerical iterations to choose $a_\perp$, and then solve a mass continuity equation, *e.g.,*

$$c_t = -w_z(c - b) - a_\perp - \dot{B} \tag{10.21}$$

where w_z is a vertical strain rate (other influences on c_t may be appropriate too), and where b is assumed time invarient (a simplification), to determine the time-evolution of $c(t)$.

10.3 Temperate-Ice Layer Growth From the Bed, or From Above?

We will now consider how the ice column described above evolves through time $t > 0$ toward the development (and ultimate decay) of a temperate ice layer. Two illustrative (idealized) cases will be considered. They will be referred to as the "low bedrock thermal inertia" case (case 1) and the "high bedrock thermal inertia"case (case 2). In the first case, we shall choose the thickness of thermally "involved" bedrock, $h_r = b - r$, to be sufficiently small (order 10 m) to allow the location of first melting to be at the ice base, $z = b$. In the second case, the value of h_r will be large, and first melting will occur above the ice base at $z = f$. The difference between the two cases considered involves details during the initial transient development of the temperate layer, *e.g.,* where first melting appears and whether the layer initially grows up from the bed or both down and up from some point above the bed. The difference between these two cases is illustrated schematically in Figure (10.2).

- **Low bedrock inertia**. In the first "low inertia" case (Fig.), there will be an initial period when the bed is melted without the immediate development of a temperate layer above the bed. This is because, as we shall discuss below, the temperature gradient at the base of the cold ice needs additional (probably short) time after the bed has melted to reach the Clapyron slope. The temperate layer, in this circumstance, will grow upward from the bed into the cold ice above. Once the temperate layer begins to grow, the focus of the numerical effort will be on appropriate specification and matching of thermodynamic constraints (boundary conditions) at the cold/temperate ice transition, $z = c$.

- **High bedrock inertia**. In the second "high inertia" case (Fig.), first melting is achieved at a distance above the bed, $z = f > b$. The temperature at this point of first melting, $T(f)$, because it is a local maximum, will be tangent to the pressure-melting curve, $T_m(z)$, and will thus allow the rapid, initial development of a temperate ice layer in the region $c_l < z < c_u$. For an initial (short) time period following the

first melting at $z = f$, c_l will advance downward toward the bed (unless, by some arbitrarily controled detail of the strain heating history, the further development of the temperate layer is arrested by a cutoff of heat–in which circumstance, we would need to consider the subsequent upward movement of c_l). When c_l reaches the bed, *i.e.,* $c_l = z$, the subsequent evolution of the thermal layer proceeds according to the same prescription as in the "low inertia" case discussed above.

In the following illustrative time-evolution example, we shall assume low-inertia conditions (because it is most simple). A high-inertia case will be left to the student as an exercise.

10.4 Illustrative Time-Evolution Examples

The illustrative example proceeds through 6 temporal stages listed as follows:

1. Cold ice, frozen bed
2. Cold ice, melted bed
3. Growing temperate layer
4. Shrinking temperate layer
5. Cold ice, melted bed
6. Cold ice, frozen bed

In effect, the ice column is heated up by strain heating, allowed to develop a temperate-ice layer, then cooled down (by artificially eliminating the heat source) and returned to cold, frozen-bed conditions. As explained in the following sections, the governing equations and boundary conditions which determine the evolution of $T(z,t)$ and μ differ from stage to stage.

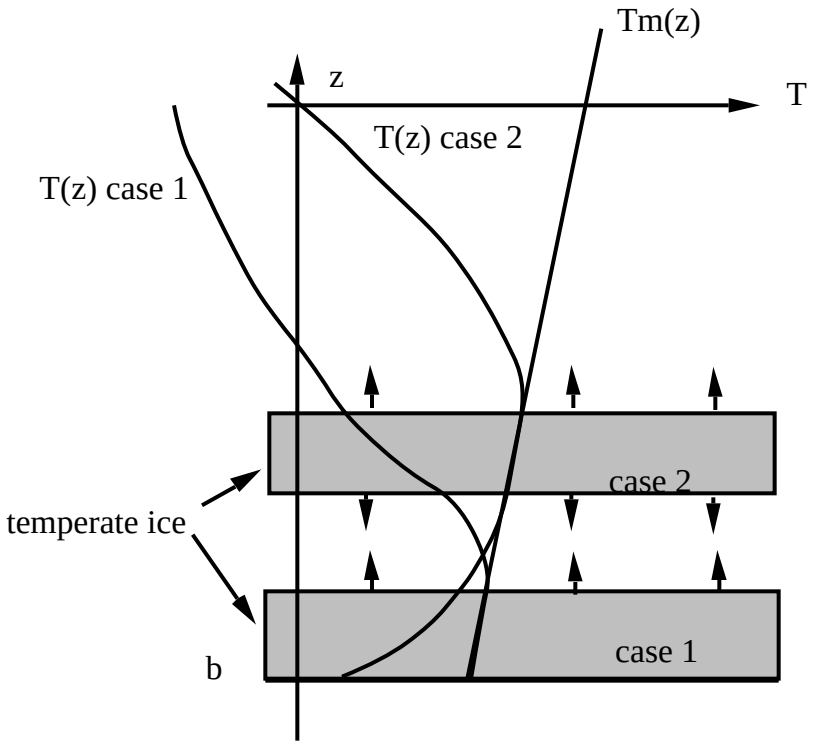

Figure 10.2: The difference between a low thermal inertia bedrock (case 1) and a high thermal inertia bedrock (case 2) concerns where melting first appears in the ice column and whether the temperate layer grows from the bottom up (case 1) or in both directions (up and down) from a point located above the bed (case 2).

10.4.1 Stage 1: cold ice, frozen bed

During the first stage of thermal evolution, $0 \leq t \leq t_m$, the entire ice column is frozen and the basal temperature, $T(z = b, t) = T_b(t)$, is below the pressure melting point $T_m = -\rho g(s - b)\Phi$. The governing equations appropriate for this stage are

$$T_t - \frac{1}{h} T_\zeta \left(\zeta \dot{A} \right) = \frac{1}{\rho c h^2} \left(k T_\zeta \right)_\zeta + \frac{W}{\rho c} \tag{10.22}$$

$$\mu(\zeta) = 0 \tag{10.23}$$

for $b < z < s$ $(0 < \zeta < 1)$, where $h = s - b$, and

$$T_t = \frac{1}{\rho_r c_r} \left(k_r T_z \right)_z \tag{10.24}$$

for $r < z < b$. Boundary conditions are

$$T(z = s, t) = T_s \tag{10.25}$$

$$T_z(z = r, t) = \frac{-G}{k_r} \tag{10.26}$$

At the ice/rock interface, the temperature and heat flux are continuous. The form of Eqn. (10.22) is taken from the previous chapter using the definition of horizontal and temporal derivatives associated with the stretched vertical coordinate $\zeta = \frac{z-b}{s-b}$.

The evolution of $T(z, t)$ through stage 1 is displayed in Figure (10.3). The MATLAB code used to perform the finite-element computation is presented at the end of this chapter.

10.4.2 Transition: stage 1 → stage 2

Stage 1, cold ice with a frozen bed, ends when T_b reaches the pressure melting point. This event defines a time $t = t_m$ when stage 1 dynamics no longer apply (because of an inappropriate boundary condition at $z = b$) and stage

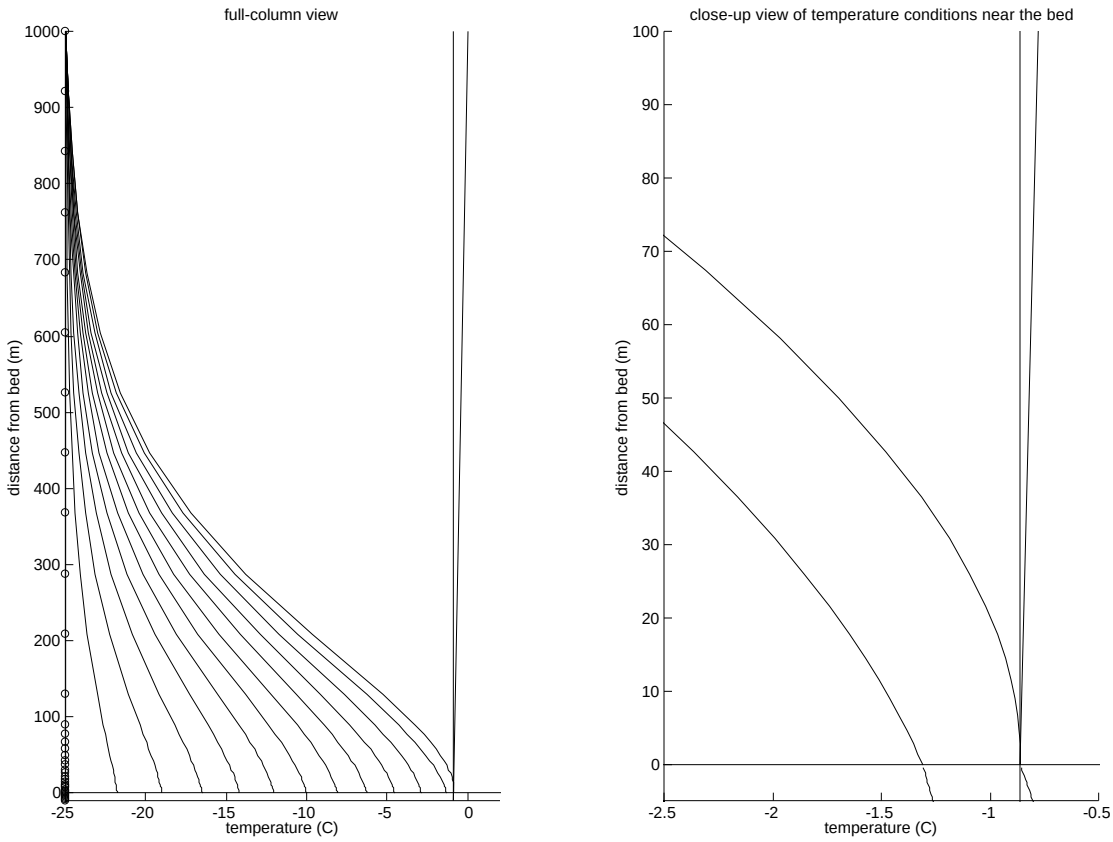

Figure 10.3: Stage 1 thermal evolution of the idealized ice-column temperature (cold ice, frozen bed). The graph on the right-hand side presents a close-up view of basal conditions where the temperate-ice layer will soon develop. Open circles shown on the initial temperature profile display the locations of finite-element nodes (2-node line segments are used as elements to discretize the vertical dimension, node spacing near the bed is logarithmic for added resolution). Each temperature profile represents $T(z,t)$ at 100-year intervals. The sloping line on the right-hand side of the graphs represents the pressure melting temperature, $T_m(z)$.

2 dynamics kick in to determine what happens to $T(z,t)$ next. The circumstance in which the location of first melting is at a level above the bed is left as an exercise to be performed by the student.

10.4.3 Stage 2: cold ice, melted bed

After $t = t_m$, but while the following condition holds true,

$$T_z(z = b) < \rho g \Phi \tag{10.27}$$

the a melted bed will exist at the bottom of an entirely cold ice column (*i.e.,* temperate ice does not exist yet). The governing equations and boundary conditions are:

$$T_t - \frac{1}{h} T_\zeta \left(\zeta \dot{A} + (1-\zeta)\dot{B} \right) = \frac{1}{\rho c h^2} \left(k T_\zeta \right)_\zeta + \frac{W}{\rho c} \tag{10.28}$$

$$\mu(\zeta) = 0 \tag{10.29}$$

for $b < z < s$ $(0 < \zeta < 1)$, and

$$T_t = \frac{1}{\rho_r c_r} \left(k_r T_z \right)_z \tag{10.30}$$

for $r < z < b$. Boundary conditions are

$$T(z = s, t) = T_s \tag{10.31}$$

$$T(z = b, t) = T_m \tag{10.32}$$

$$T_z(z = r, t) = \frac{-G}{k_r} \tag{10.33}$$

and the basal melting rate $\dot{B}$ (meters of ice equivalent per unit time) is given by

$$\dot{B} = \frac{k T_z|_+ - k_r T_z|_-}{\rho L_f} \tag{10.34}$$

For simplification, heat created by basal sliding is disregarded in the determination of $\dot{B}$.

With $\dot{B} \geq 0$, a basal water layer develops at the bed (or may be contained within basal sediments) with a thickness (effective thickness) $h_w(t)$ determined by,

$$(h_w)_t = \frac{\rho}{\rho_w} \dot{B} \tag{10.35}$$

where ρ_w is the density of water, and the water thickness is subject to the constraint that it be positive, *i.e.,* that $h_w \geq 0$. The basal water layer is assumed not to drain via subglacial water or sediment transport processes.

The evolution of $T(z,t)$ through stage 2 is not displayed because it was too short to have been adequately resolved by the 100-year timestep size.

10.4.4 Transition: stage 2 → stage 3

At time $t = t_{gt}$, the condition,

$$T_z(z = b) < \rho g \Phi \tag{10.36}$$

no longer holds true. At this point, the temperature gradient at the bed is equal to the Clapyron slope

$$T_z(z = b) = \rho g \Phi \tag{10.37}$$

and a basal temperate-ice layer begins to grow (develop upward). The boundary between cold and temperate ice, $c(t)$, will move upward from its initial (zero temperate ice thickness) position at $c = b$. (Recall that we do not consider the case where, due to high bedrock thermal inertia, a temperate layer would initially grow upward *and* downward from some finite distance above the bed.)

10.4.5 Stage 3: growing temperate layer

The governing equations during the time period after $t = t_{gt}$, but before the time when the temperate layer stops growing (see below for the conditions

when this holds true), are:

$$T_t - \frac{1}{h_c} T_{\zeta_c} \left(\zeta_c \dot{A} - (1 - \zeta_c) a_\perp \right) = \frac{1}{\rho c h_c^2} \left(k T_{\zeta_c} \right)_{\zeta_c} + \frac{W(z(\zeta_c))}{\rho c} \quad (10.38)$$

$$\mu(\zeta) = 0 \quad (10.39)$$

for $c(t) < z < s$ $(0 < \zeta_c < 1)$, where $\zeta_c = \frac{z - c(t)}{s - c(t)}$ is the stretched vertical coordinate *applicable to the cold layer only.* The strain heating term $W(z(\zeta_c))$ is written as a function of ζ_c to emphasize the point that the stretched vertical coordinate ζ_c does not cover the entire ice column in its variation from 0 to 1 (see Fig. 10.1). The governing equations in the temperate-ice layer are

$$T = T_m(z) \quad (10.40)$$

$$\mu_t(z) = \frac{1}{c - b} \int_b^c \frac{W}{\rho L_f} dz \quad (10.41)$$

for $b < z < c$, (recall that the above formulation for liquid-water content represents a simplified, bulk analysis; see Hutter and others, 1988 for a more detailed theoretical treatment), and

$$T_t = \frac{1}{\rho_r c_r} \left(k_r T_z \right)_z \quad (10.42)$$

for $r < z < b$, with boundary conditions

$$T(z = s, t) = T_s \quad (10.43)$$

$$T(z = c, t) = T_m(z) \quad (10.44)$$

$$T_z(z = c, t) = \rho g \Phi \quad (10.45)$$

$$T_z(z = r, t) = \frac{-G}{k_r} \quad (10.46)$$

The variable $a_\perp(t) < 0$ is the "cold-ice ablation" rate, *i.e.,* the rate at which ice is ejected from the cold-ice layer through the cold/temperate ice boundary into the temperate-ice layer.

The value of $a_\perp(t)$ is determined using the control method discussed previously. In essence, $a_\perp$ is determined interatively by the procedure required to make the otherwise overdetermining thermal boundary conditions at $z = c(t)$, Eqns. (10.44) and (10.45), compatible.

The basal melting rate $\dot{B}$ is given by

$$\dot{B} = \frac{k\rho g\Phi - k_r T_z|_-}{\rho L_f} \tag{10.47}$$

and this leads to additional increase in the basal water layer thickness:

$$(h_w)_t = \frac{\rho}{\rho_w}\dot{B} \tag{10.48}$$

The growth of the temperate-ice layer is determined by a mass-continuity equation that accounts for $a_\perp$, $\dot{B}$ and vertical strain which, in the present idealized example (*i.e.,* constant ice thickness), is given by

$$\dot{e}_{zz} = -\frac{\dot{A} - \dot{B}}{h} \tag{10.49}$$

The resulting equation for temperate-ice layer thickness, $(c(t) - b) = h_T$, is

$$(h_T)_t = \dot{e}_{zz}h_T - a_\perp - \dot{B} \tag{10.50}$$

The shrinkage of the cold-ice layer thickness, $(s - c(t)) = h_c$ is given by

$$(h_c)_t = \dot{e}_{zz}h_c + a_\perp + \dot{A} \tag{10.51}$$

As a consitency check, observe that

$$h_t = (h_T + h_c)_t = \dot{e}_{zz}h + \dot{A} - \dot{B} = 0 \tag{10.52}$$

The evolution of $T(z, t)$ through stage 3 is displayed in Figure (10.4). The MATLAB code used to perform the finite-element computation is presented at the end of this chapter.

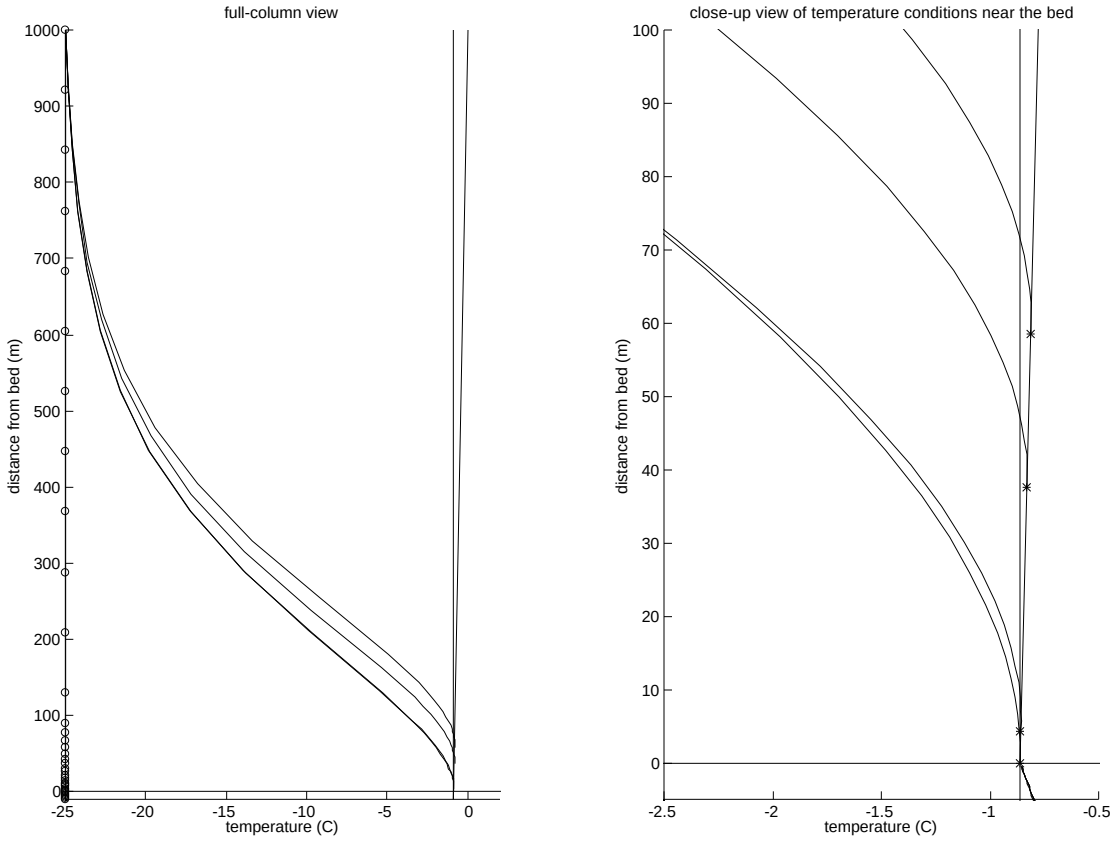

Figure 10.4: Stage 3 thermal evolution of the idealized ice-column temperature (growing temperate-ice layer, melted bed). The graph on the right-hand side presents a close-up view of the development of the temperate-ice layer. Each temperature profile represents $T(z,t)$ at 100-year intervals. The sloping line on the right-hand side of the graphs represents the pressure melting curve, $T_m(z)$. The asterisk (*) symbol denotes the cold/temperate ice transition. Observe that the vertical temperature gradient at the cold/temperate ice transition is tangent to the pressure-melting curve. The first temperature profile (with zero temperate-ice layer thickness) represents the temperature profile at the termination of stage 1 (also shown in Fig. 10.3).

10.4.6 Transition: stage 3 → stage 4

At a point $t = t_h$, strain heating is arbitrarily cut off, and the temperate layer must begin to shrink. This cutoff is arranged arbitrarily in the present example for the sake of illustration. In nature, a shrinking temperate-ice layer can occur as a result of various events including downward advection of cold ice from the ice-sheet surface, a change in flow regime (*e.g.,* onset of basal sliding), or a change in surface temperature.

10.4.7 Stage 4: shrinking temperate layer

The governing equations during the time period after $t = t_h$, but before the time when the temperate layer thickness becomes zero (see below for the conditions when this holds true), are:

$$T_t - \frac{1}{h_c} T_{\zeta_c} \left(\zeta_c \dot{A} - (1 - \zeta_c) a_\perp \right) = \frac{1}{\rho c h_c^2} \left(k T_{\zeta_c} \right)_{\zeta_c} + \frac{W(z(\zeta_c))}{\rho c} \quad (10.53)$$

$$\mu(\zeta) = 0 \quad (10.54)$$

for $c(t) < z < s$ $(0 < \zeta_c < 1)$, where $\zeta_c = \frac{z - c(t)}{s - c(t)}$ is the stretched vertical coordinate *applicable to the cold layer only.* The strain heating term $W(z(\zeta_c))$ is written as a function of ζ_c to emphasize the point that the stretched vertical coordinate ζ_c does not cover the entire ice column in its variation from 0 to 1 (see Fig. 10.1). The governing equations in the temperate-ice layer are

$$T = T_m(z) \quad (10.55)$$

$$\mu_t(z) = \frac{1}{c - b} \int_b^c \frac{W}{\rho L_f} dz \quad (10.56)$$

for $b < z < c$, and

$$T_t = \frac{1}{\rho_r c_r} \left(k_r T_z \right)_z \quad (10.57)$$

for $r < z < b$.

Boundary conditions are

$$T(z = s, t) = T_s \tag{10.58}$$

$$T(z = c, t) = T_m(z) \tag{10.59}$$

$$T_z(z = r, t) = \frac{-G}{k_r} \tag{10.60}$$

The variable $a_\perp(t) < 0$ is the "cold-ice ablation" rate, *i.e.,* the rate at which ice is ejected from the cold-ice layer through the cold/temperate ice boundary into the temperate-ice layer.

Observe that the additional boundary condition applicable at $z = c(t)$ in the previous stage (growing temperate layer), *i.e.,* $T_z|_{z=c} = \rho g \Phi$, is no longer applicable because a discontinuity in conductive heat flux at $z = c$ is allowed with $a_\perp < 0$ to account for latent heat release as ice moves from the temperate side to the cold side of the cold/temperate transition. The value of $a_\perp$ is determined by the jump condition in temperature gradient, as explained previously:

$$a_\perp = \frac{-k\left(T_z|_{c+} - T_z|_{c-}\right)}{\rho L_f \mu} \tag{10.61}$$

The basal melting rate $\dot{B}$ is given by

$$\dot{B} = \frac{k\rho g\Phi - k_r T_z|_-}{\rho L_f} \tag{10.62}$$

and this leads to additional increase in the basal water layer thickness:

$$(h_w)_t = \frac{\rho}{\rho_w}\dot{B} \tag{10.63}$$

The shrinkage of the temperate-ice layer is determined by a mass-continuity equation that accounts for $a_\perp$, $\dot{B}$ and vertical strain which, in the present idealized example (*i.e.,* constant ice thickness), is given by

$$\dot{e}_{zz} = -\frac{\dot{A} - \dot{B}}{h} \tag{10.64}$$

The resulting equation for temperate-ice layer thickness, $(c(t) - b) = h_T$, is

$$(h_T)_t = \dot{e}_{zz} h_T - a_\perp - \dot{B} \tag{10.65}$$

The growth of the cold-ice layer thickness, $(s - c(t)) = h_c$ is given by

$$(h_c)_t = \dot{e}_{zz} h_c + a_\perp + \dot{A} \tag{10.66}$$

Recall that, by definition, $a_\perp < 0$ when temperate ice is converted to cold ice.

The evolution of $T(z, t)$ through stage 4 is displayed in Figure (10.5). The MATLAB code used to perform the finite-element computation is presented at the end of this chapter.

10.4.8 Transition: stage 4 → stage 5

At time $t = t_{m_2}$, the thickness of the shrinking temperate layer becomes zero, *i.e.,* $h_T = 0$. At this point, the bed is melted but the entire ice column above the bed is colder than the pressure-melting point.

10.4.9 Stages 5 & 6: cold ice, melted → frozen bed

The governing equations for stages 5 and 6 are identical to those of stages 2 and 1, respectively. The evolution of $T(z, t)$ through these two stages is displayed in Figure (10.6).

10.4.10 Summary

A summary of the onset of basal melting, growth and decay of a temperate-ice layer, and the ultimate re-freezing of the bed is presented in Figure (10.7).

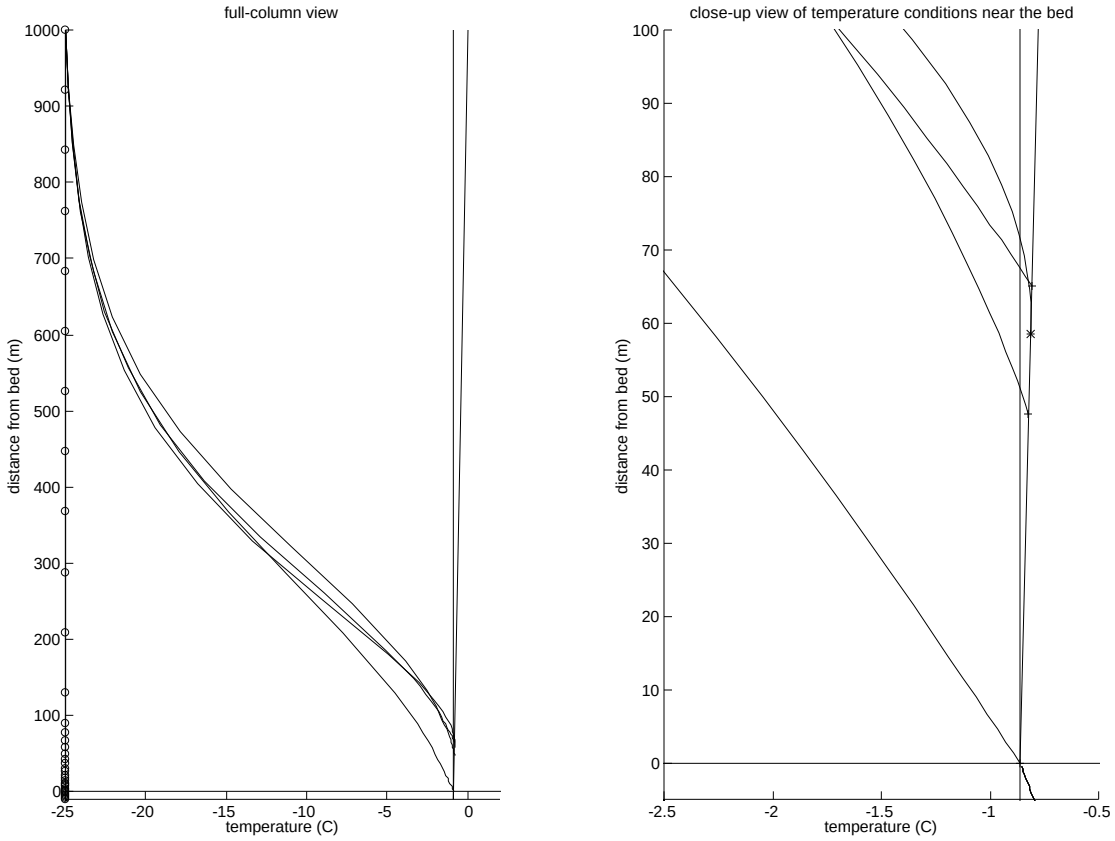

Figure 10.5: Stage 4 thermal evolution of the idealized ice-column temperature (shrinking temperate-ice layer, melted bed). The graph on the right-hand side presents a close-up view of the development of the temperate-ice layer. Each temperature profile represents $T(z,t)$ at 100-year intervals. The sloping line on the right-hand side of the graphs represents the pressure melting curve, $T_m(z)$. The horizontal tick-mark (-) symbol denotes the cold/temperate ice transition while this transition is moving down toward the bed. The asterisk (*) denotes the cold/temperate transition during the last part of stage 3. Observe that the vertical temperature gradient at the cold/temperate ice transition is no longer tangent to the pressure-melting curve. This is because ice flow through the cold/temperate ice transition from below releases latent heat which demands a jump discontinuity in the temperature gradient at the transition.

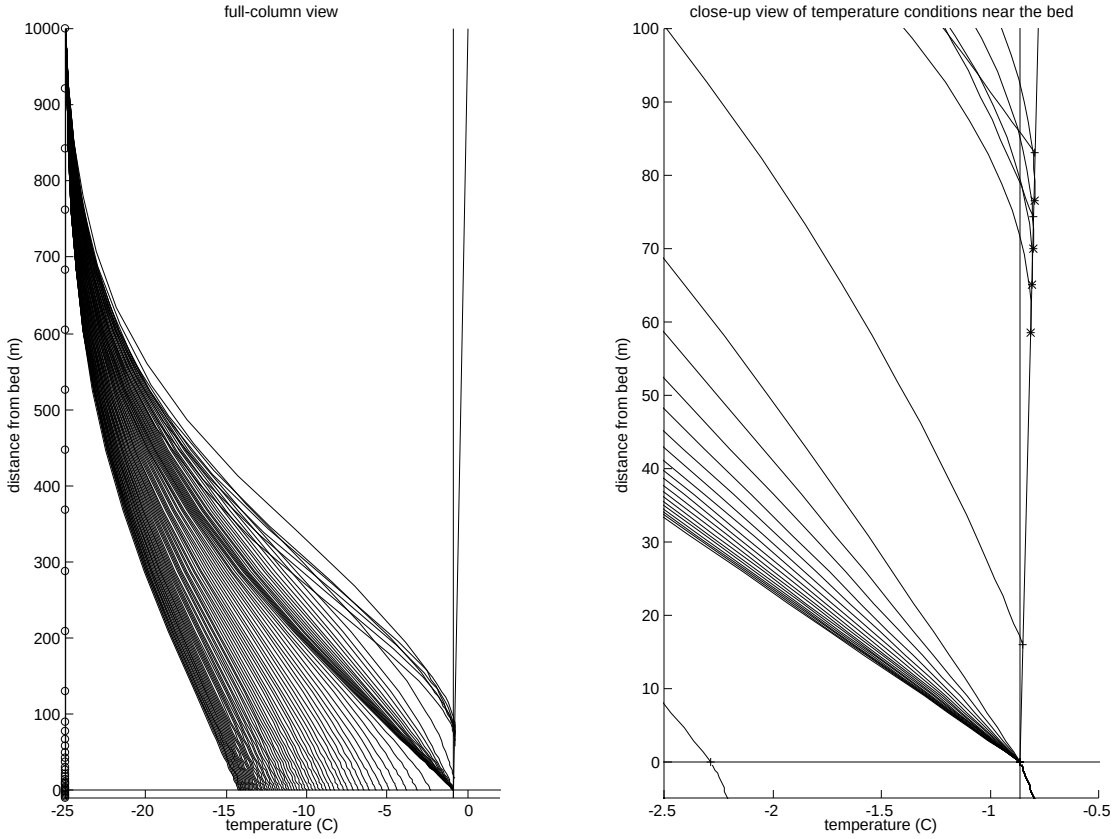

Figure 10.6: Thermal evolution through stages 5 and 6. The bed initially remains melted while the temperature gradient above the bed and the basal water layer thickness decrease. Once the basal water layer thickness goes to zero, latent heat is no longer available to balance upward heat conduction through the ice and the ice column freezes to the bed. Subsequent cooling toward a steady state (frozen bed) temperature profile is partially shown.

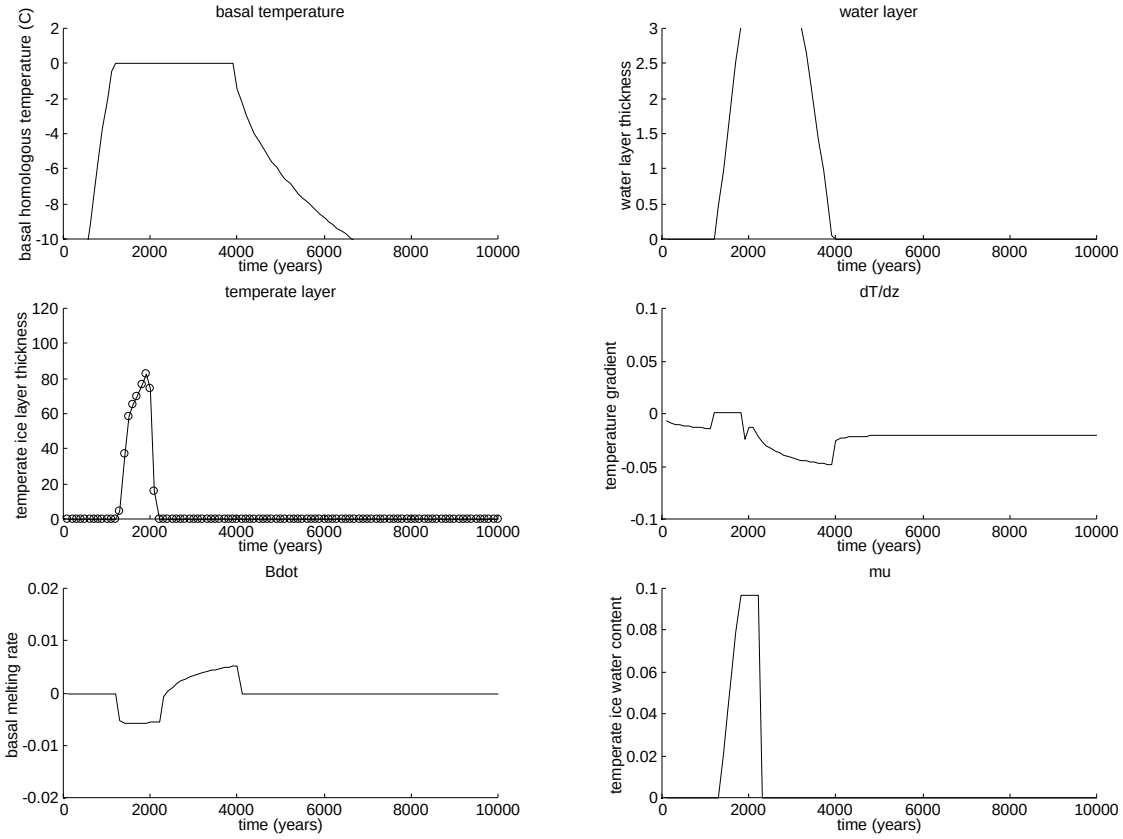

Figure 10.7: Summary of thermal evolution through stages 1 through 6. Units for thickness are meters, for rates are meters per year, and for gradients are degrees Centigrade per meter.

10.5 Exercise: High Bedrock Thermal Inertia

The student is asked to reformulate and solve the thermal-evolution problem treated above in a circumstance where the bedrock thermal inertia is high and strain heating more persistent, *i.e.,* where $r = -500$ and $t_h = 3000$ years. Define any new stages of thermal evolution (*e.g.,* frozen bed, temperate ice conditions) and carefully describe the conditions which determine their onset and termination. Also, formulate the equations which govern temperature and liquid-water content in the temperate layer and basal cold layer during the stage when the temperate-ice layer grows down toward the bed from above. An example of high bedrock inertia thermal evolution (where the temperate ice layer grows down toward the bed from above) is displayed in Figures (10.8) and (10.9).

10.6 MATLAB Scripts Used to Illustrate Polythermal Ice-Column Evolution

The MATLAB scripts used to model polythermal ice-column evolution in the above example (*i.e.,* as shown in Figs. 10.3 - **??**) are provided separately.

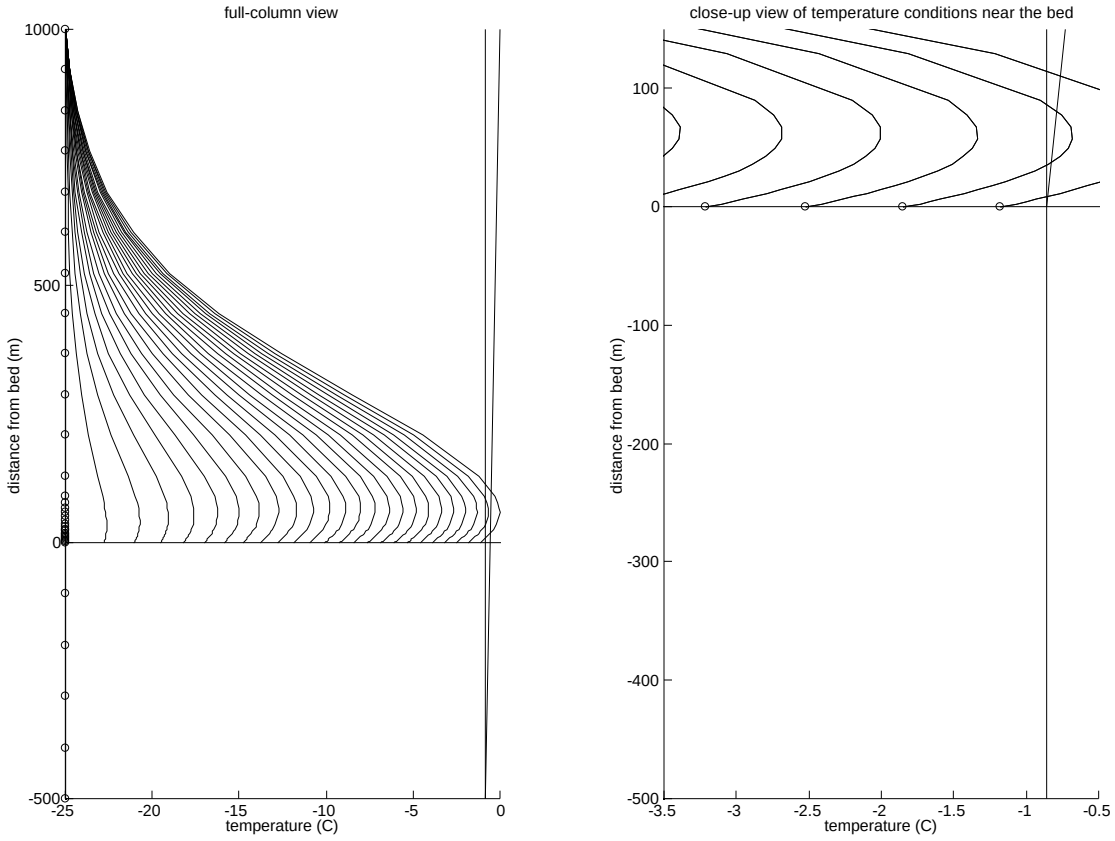

Figure 10.8: Thermal evolution through stage 1 for case in which bedrock thermal inertia is high (*i.e.,* 500 m of bedrock, $r = -500$ m) and in which strain heating is persistant througout the experiment. The time difference between successive temperature profiles is 100 years. Observe that the location of first melting occurs above the bed. The temperate-ice layer thus grows in two directions (up and down) through an initial stage of development. Downward growth of a cold/temperate ice transition is not treated in this chapter, but is left as an exercise for the student.

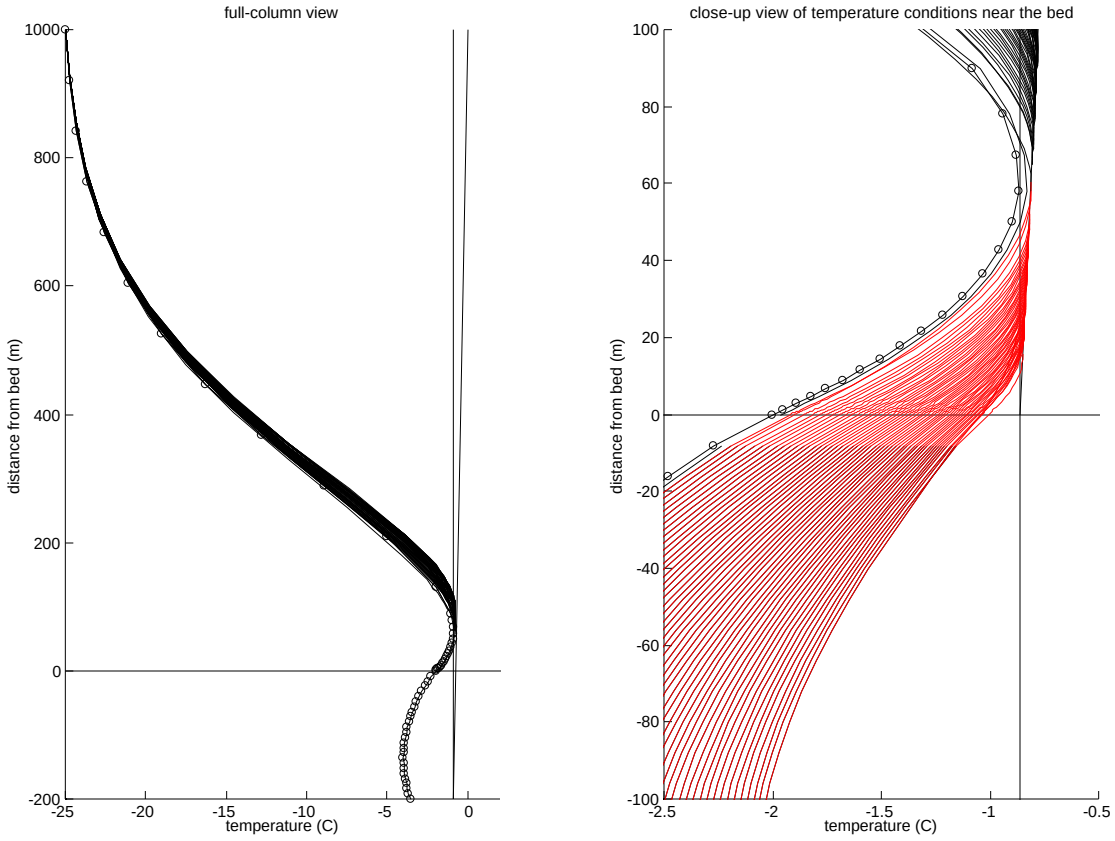

Figure 10.9: Development of temperate-ice layer for case in which bedrock thermal inertia is high (*i.e.,* 200 m of bedrock, $r = -200$ m, note difference with case shown in Figure 10.8) and in which strain heating is persistant througout the experiment. The time difference between successive temperature profiles is 5 years. Observe that the location of first melting occurs above the bed. The temperate-ice layer thus grows in two directions (up and down) through an initial stage of development. Downward growth of a cold/temperate ice transition is not treated in this chapter, but is left as an exercise for the student.

Chapter 11

Ice Sheet Thermodynamics: 2-D (Flowline) Modelling Techniques

In this chapter, we will construct a two-dimensional, "flowline" ice-sheet model to provide a simple alternative to the three-dimensional model developed in Chapter 9. We will complete the EISMINT Level 1 intercomparison test as a means to evaluate the model's performance.

11.1 Governing Equations

The governing equations for the dynamic/thermodynamic flowline ice-sheet model developed here are listed as follows. The equations which govern the horizontal and vertical velocity fields are:

$$u(z) = -\rho g(s-b)C(T)\frac{\partial s}{\partial x} - 2\rho g\frac{\partial s}{\partial x}I(z) \tag{11.1}$$

$$w(z) = -\int_b^z \frac{\partial u}{\partial x}dz' + \frac{\partial b}{\partial t} + u(b)\frac{\partial b}{\partial x} - \dot{B} \tag{11.2}$$

where x is the horizontal coordinate directed down the flowline, z is the vertical coordinate (positive up), u is the horizontal velocity directed down the flowline, w is the vertical velocity, s and b are the surface and basal elevation, respectively, ρ is the ice density (assumed to be constant, 910 kg m^{-3}), g is the acceleration of gravity (9.81 m s^2), and $\dot{B}$ is the basal melting rate (positive for melting, negative for freezing, in meters of ice equivalent per second). To keep the treatment simple, lateral flow convergence along the flowline is not accounted for in the expression for the vertical velocity. Convergence, because it influences the vertical velocity, is expected to be an important influence on the temperature-depth profile computed by the model.

The basal sliding constant $C(T)$ is assumed to be nonzero when the basal temperature $T(b)$ is at the pressure-melting point (Payne, 1995):

$$C(T) = \begin{cases} 5 \times 10^{-3} \text{ m a}^{-1} \text{ Pa}^{-1} & \text{if } T(b) = T_m \\ \\ 0 & \text{if } T(b) < T_m \end{cases} \tag{11.3}$$

The variable $T(x, z, t)$ is the temperature, and T_m is the z-dependent melting point,

$$T_m(z) = T_o - \rho g (s - z) \Phi \tag{11.4}$$

where $\Phi = 8.71 \times 10^{-4}$ K Pa^{-1} (Payne, 1995), and $T_o = 273.15$ K. The factor $I(z)$ determines the deformational velocity of the ice flow, and for Glen's flow law (with exponent 3) is defined by

$$I(z) = \int_b^z E A(T^*)(\rho g)^2 \left(\frac{\partial s}{\partial x} \right)^2 (s - z')^3 \, dz' \tag{11.5}$$

The rate-enhancement factor E is essentially a fudge factor designed to account for empirically determined inadequacies of Glen's law. The value of E is commonly taken to be 1 for "Holocene ice" and 3 for "glacial-period ice", for example, in modelling studies of the Greenland ice sheet. The creep-rate factor $A(T^*)$ is assumed to have a standard thermodynamic form involving a rate constant factor a and an activation energy Q:

$$A(T^*) = a \exp \left(\frac{-Q}{RT^*} \right) \tag{11.6}$$

where $R = 8.31$ J mol^{-1} K^{-1} is the gas constant, and the flow-law parameters (determined from laboratory studies of polycrystaline ice assumed to be isotropic) are commonly taken to be:

$$a = \begin{cases} 7.23 \times 10^{-12} \text{ s}^{-1} \text{ Pa}^{-3} & \text{if } T^* < 263 \text{ K} \\ \\ 3.47 \times 10^{4} \text{ s}^{-1} \text{ Pa}^{-3} & \text{if } T^* \geq 263 \text{ K} \end{cases} \tag{11.7}$$

and

$$Q = \begin{cases} 6.0 \times 10^{4} \text{ J}^{-1} \text{ mol}^{-1} & \text{if } T^* < 263 \text{ K} \\ \\ 13.9 \times 10^{4} \text{ J}^{-1} \text{ mol}^{-1} & \text{if } T^* \geq 263 \text{ K} \end{cases} \tag{11.8}$$

The temperature in the creep-rate factor formula $T^* = T - T_m + T_o$ is the Homologous temperature (temperature relative to the pressure melting point) in degrees Kelvin (note the addition of T_o).

Mass balance of the ice sheet is expressed by the following equation for ice thickness h,

$$\frac{\partial h}{\partial t} = -\frac{\partial q}{\partial x} + \dot{A} - \dot{B} \tag{11.9}$$

where $q = D\frac{\partial s}{\partial x}$ is the horizontal mass flux integrated over the ice column, $\dot{B}$ is the basal melting rate (positive for melting, negative for freezing), and $\dot{A}$ is the surface snow accumulation rate expressed in meters of ice equivalent per year. Note that ice densification effects are disregarded. A common practice is to substitute Equations (12.6) into Equation (12.13) giving,

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial s}{\partial x}\right) + \dot{A} - \dot{B} \tag{11.10}$$

where the effective diffusivity D is defined by,

$$D = \int_b^s 2\rho g I(z')dz' + \rho g(s-b)^2 C \tag{11.11}$$

Kinematic boundary conditions on the free surface $z = s$ and basal surface $z = b$ are recorded for use elsewhere:

$$\frac{\partial s}{\partial t} + u(s)\frac{\partial s}{\partial x} = w(s) + \dot{A} \tag{11.12}$$

and,

$$u(b)\frac{\partial b}{\partial x} = w(b) + \dot{B} \tag{11.13}$$

where we have made the assumption that $\frac{\partial b}{\partial t} = 0$ (no isostatic or bed erosion effects are included).

Heat flow continuity in both ice and underlying bedrock is expressed using the standard advective/diffusive equation with variable heat-flow parameters:

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + w\frac{\partial T}{\partial z} = \frac{1}{\rho c}\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) + \frac{W}{\rho c} \tag{11.14}$$

for $b < z < s$; and

$$\frac{\partial T}{\partial t} = \frac{1}{\rho_r c_r}\frac{\partial}{\partial z}\left(k_r\frac{\partial T}{\partial z}\right) \tag{11.15}$$

for $z_r \leq z \leq b$, and where z_r is a level at a fixed elevation below the ice/bedrock interface (or seabed) where the geothermal flux gradient is applied as a boundary condition (see below). The above equations reflect the standard assumption that horizontal heat conduction is neglegible in comparison with horizontal advection, veritcal advection and conduction, and viscous heat dissipation. The horizontal and vertical velocities referred to in Equations (12.22) and (12.23) are described by Equations (12.6) and (12.7).

Observe that the vertical gradients of k and k_r are acknowledged in the form of the thermal diffusion in Equations (12.22) and (12.23). The horizontal gradients of k and k_r, and all the spatial gradients of ρc and $\rho_r c_r$ are disregarded. This constitutes a simplification that is used widely in ice-sheet thermodynamic studies. It's justification is not provided here (and indeed, there may be an inconsistency to be corrected in future work). See Greve [1995] for further discussion of this matter.

Parameters appearing in Equations (12.22) and (12.23) include the thermal heat capacities for ice and rock, c and c_r, respectively:

$$c = 2115.3 + 7.79293(T - T_o) \tag{11.16}$$

$$c_r = 1000 \tag{11.17}$$

in units of J kg^{-1} K^{-1} (Huybrechts, 1993), and the thermal conductivities for ice and rock, k and k_r, respectively:

$$k = 3.101 \times 10^8 \exp\left(-0.0057T\right) \tag{11.18}$$

$$k_r = 3.3 \tag{11.19}$$

in units of W m K^{-1} (Huybrechts, 1993). The density of rock ρ_r is taken to be that of typical sedimentary rock of interest below the central portions of the Laurentide and West Antarctic ice sheets, 2700 kg m^{-3}. The term W in Eqn. (12.22) is the viscous heating term formally defined by

$$W = \sum_i \sum_j \dot{e}_{ij} T'_{ij} \tag{11.20}$$

where $\dot{e}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ is the strain rate, and T'_{ij} is the deviatoric stress. According to the common assumptions associated with grounded ice-sheet stress balance (*i.e.,* neglecting longitudinal stress), W can be written as:

$$W(z) = (\rho g)^4 \, 2 \left(\frac{\partial s}{\partial x}\right)^4 (s - z)^4 \, EA(T^*) \tag{11.21}$$

Boundary conditions are necessary at the top $z = s$ and bottom $z = z_r$ of the ice/rock column. At the top, a surface temperature (depending on atmospheric conditions) is specified

$$T(s) = T_s \tag{11.22}$$

At the bottom $z = z_r$, well below the ice/bedrock interface, a geothermal temperature gradient is specified

$$\frac{\partial T}{\partial z} = -\frac{\Gamma}{k_r}. \tag{11.23}$$

The geothermal flux Γ is typically taken to be one geophysical heat flow unit (0.42 W m^{-2}). The value of z_r is typically chosen to be a fixed distance below $z = b$. A useful consideration for choosing this distance $(b - z_r)$ is the e-folding penetration depth λ for temperature oscillations forced at the ice/bedrock interface of frequency ω (Carlsaw and Jaeger, 1954):

$$\lambda = \left(\frac{2\kappa_r}{\omega}\right)^{\frac{1}{2}} \tag{11.24}$$

where the thermal diffusivity of bedrock is defined by $\kappa_r = \frac{k_r}{\rho_r c_r}$. For Heinrich-event frequency oscillations (periodicity of about 10,000 years), the penetration depth λ is 375 m.

Unless there are inflow boundaries around the horizontal edges of the ice sheet, thermal boundary conditions are not required around the edges of the ice sheet. This is because the horizontal mode of heat transfer in an ice sheet is advection, a process that demands boundary conditions only where ice flow into the domain.

A material constraint on the ice-sheet thermodynamics is that ice never warm above the presssure-melting point. When layers of finite thickness reach the pressure-melting point, the ice sheet is said to become "polythermal" (Hutter, 1982; Hutter, Blatter and Funk, 1988). A review of polythermal ice-sheet modelling is provided by Greve (1995). Polythermal conditions typically occur as a result of viscous dissipation, which heats the ice column internally and permits the locus of a temperature maximum (the pressure-melting point, by definition) to move away from the boundary (a constraint that is otherwised forced by a consideration similar to the maximum principle of solutions of Laplace Equation). When ice becomes temperate (at the pressure melting point), its vertical temperature gradient becomes that dictated by the "Clapyron slope" of water (the change of the melting temperature with pressure). In this circumstance, the vertical heat flux is fixed (near zero), so further heat transfer (via internal heating and flux from non-temperate portions of the ice sheet) acts only to modify the liquid water content of the ice. In the ice-sheet model by Greve (1995), for example, the heat-flow continuity equation in the temperate ice is replaced with a diffusive-type equation for water content, and an internal free surface (called the CTS, cold/temperate ice transition surface) must be monitored through the use of energy flux and mass flux matching conditions.

In the examples described in this chapter, polythermal ice conditions are not treated; thus, we shall make a standard (but not necessarily well justified) assumption that the melting point is achieved only at the ice base. In this simplification, we augment the heat equation developed above with logical conditions determining when the basal ice boundary condition is fixed at the melting point, or when the basal ice temperature (frozen) is allowed to freely

vary according to the combined ice/bedrock heat equation. When the base is at the pressure-melting point, we compute a basal melting rate, $\dot{B}$ (positive for melting, meters of ice equivalent per year), to balance the heat budget at the basal ice interface. We define heat-flux terms H_o and H_i as the outward- and inward-directed heat fluxes to the interface $z = b$, rexpectively, by

$$H_o = -k\frac{\partial T}{\partial z}|_{z=b+} \tag{11.25}$$

and,

$$H_i = -k_r\frac{\partial T}{\partial z}|_{z=b-} + u(b)\tau_b + \begin{cases} \frac{\rho\dot{B}}{L_f} & \text{if } \dot{B} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{11.26}$$

where τ_b is the vector-valued basal stress which, in the absence of longitudinal stress, is $-\rho g h\frac{\partial s}{\partial x}$, and $u(b)$ is the basal sliding velocity (which can be nonzero when the bed is melted). The product $u(b)\tau_b$ is assumed positive. With the above definitions for H_o and H_i, $\dot{B}$ is determined by

$$\dot{B} = \frac{H_i - H_o}{\rho L_f} \tag{11.27}$$

where $L_f = 3.35 \times 10^5$ J kg^{-1} is the latent heat of fusion for pure water. In circumstances, where $\dot{B}$ is nonzero, a basal water layer can develop. When $\dot{B} > 0$ (melting), the presence or absence of basal water does not influence the thermal evolution of the ice base. When $\dot{B} < 0$ (freezing), a basal water layer must be present to supply the ice that is accumulated on the base. In this circumstance, if there is insufficient basal water to supply the required basal freezing, the base of the ice will freeze to the bed and the basal temperature will drop below the pressure melting point.

The above expressions describing the basal melting condition are interlinked; so, it is not simple to determine when a frozen bed should be converted to a melted bed, or a melted bed to a frozen bed. In brief, a frozen bed is melted when the solution of the heat equation dictates that $T(b)$ rise to the pressure-melting point; and a melted bed will freeze when the vertical heat flux into the ice is greater than the heat source into the bed by the combination of (1) conduction from the rock below, (2) heat generation by basal sliding, and (3) latent heat released by freezing of available water stored in the bed.

11.2 Contour-Following Vertical Coordinate

To handle the variable vertical dimension of the ice sheet, a new vertical coordinate ζ is defined such that $\zeta = 1$ at $s(x, y, t)$ and $\zeta = 0$ at $b(x, y, t)$. The relation between ζ and z is

$$\zeta = \frac{z - b}{s - b} \tag{11.28}$$

and

$$z = (s - b)\zeta + b \tag{11.29}$$

Using the above definitions, z-derivatives are converted to ζ-derivatives:

$$\frac{\partial \cdot}{\partial z} = \frac{1}{(s - b)} \frac{\partial \cdot}{\partial \zeta} \tag{11.30}$$

$$\frac{\partial^2 \cdot}{\partial z^2} = \frac{1}{(s - b)^2} \frac{\partial^2 \cdot}{\partial \zeta^2} \tag{11.31}$$

The conversion of x- and t-derivatives evaluated at fixed z to their counterparts evaluated at fixed ζ is somewhat more complicated. Following the considerations described in Chapter 9, the derivative of T with respect to x on a surface of constant z is

$$T_x|_{z=\text{constant}} = T_x|_{\zeta=\text{constant}} - \frac{1}{h} T_\zeta \left(\zeta \frac{\partial s}{\partial x} + (1 - \zeta) \frac{\partial b}{\partial x} \right) \tag{11.32}$$

where subscripts x, z and ζ denote partial derivatives of T, and $h = (s - b)$ is the ice thickness. The expression for T_t is

$$T_t|_{z=\text{constant}} = T_t|_{\zeta=\text{constant}} - \frac{1}{h} T_\zeta \left(\zeta \frac{\partial s}{\partial t} + (1 - \zeta) \frac{\partial b}{\partial t} \right) \tag{11.33}$$

With the above conversion formulae for T_x, T_z and T_t, we rewrite Equation (12.22) as follows:

$$T_t + uT_x + \frac{w}{h} T_\zeta$$

$$-\frac{1}{h}T_\zeta\Big(u\zeta\frac{\partial s}{\partial x}+u(1-\zeta)\frac{\partial b}{\partial x} + \zeta\frac{\partial s}{\partial t}+(1-\zeta)\frac{\partial b}{\partial t}\Big) = \frac{1}{\rho c h^2}\frac{\partial}{\partial\zeta}\left(k\frac{\partial T}{\partial\zeta}\right)+\frac{W}{\rho c} \tag{11.34}$$

Observe that

$$\frac{\partial s}{\partial t}+u\frac{\partial s}{\partial x}=w(s)+\dot{A} \tag{11.35}$$

and,

$$\frac{\partial b}{\partial t}+u\frac{\partial b}{\partial x}=w(b)+\dot{B} \tag{11.36}$$

Substitution of the above equations into Equation (11.34) gives,

$$T_t+uT_x +\frac{1}{h}T_\zeta\left(w(z)-\zeta\left(w(s)+\dot{A}\right)-(1-\zeta)\left(w(b)+\dot{B}\right)\right) = \frac{1}{\rho c h^2}\frac{\partial}{\partial\zeta}\left(k\frac{\partial T}{\partial\zeta}\right)+\frac{W}{\rho c} \tag{11.37}$$

This equation may be rewriten as follows:

$$T_t+uT_x +\frac{1}{h}T_\zeta\left(w(z)-w(b)-\zeta\left(w(s)-w(b)\right)-\zeta\dot{A}-(1-\zeta)\dot{B}\right) = \frac{1}{\rho c h^2}\frac{\partial}{\partial\zeta}\left(k\frac{\partial T}{\partial\zeta}\right)+\frac{W}{\rho c} \tag{11.38}$$

Observe, however, that

$$w(z)-w(b)=-\int_b^z\frac{\partial u}{\partial x}dz' \tag{11.39}$$

and,

$$w(s)-w(b)=-\int_b^s\frac{\partial u}{\partial x}dz' \tag{11.40}$$

Substitution of Eqns. (11.39) and (11.40) into Eqn. (11.38) gives

$$\begin{aligned} & T_t + uT_x \\ & + \frac{1}{h} T_\zeta \left(\mathcal{D}(\zeta) - \zeta \dot{A} - (1-\zeta)\dot{B} \right) \\ & = \frac{1}{\rho c h^2} \frac{\partial}{\partial \zeta} \left(k \frac{\partial T}{\partial \zeta} \right) + \frac{W}{\rho c} \end{aligned} \tag{11.41}$$

where $\mathcal{D}(\zeta)$ is an ice-divergence parameter defined by,

$$\mathcal{D}(\zeta) = h \left(\zeta \int_0^1 \frac{\partial u}{\partial x} d\zeta - \int_0^\zeta \frac{\partial u}{\partial x} d\zeta' \right) \tag{11.42}$$

11.3 Discretization

According to the diagnostic relations between u, w and s for a grounded ice-sheet with "inland ice" type flow (Eqns. 12.6 and 12.7), the computation of the vertical velocity w and the $\mathcal{D}$-term as a function of ζ requires that the second spatial derivative of s, *i.e.*, $\frac{\partial^2 s}{\partial x^2}$, be evaluated on the computational domain. This requirement necessitates the use of high-order element interpolation using Hermite polynomial basis functions.

Variables s, b, $h = s - b$ are represented on each element's horizontal span, $x_l \leq x \leq x_r$, by the sum

$$s(x) = \sum_{j=1}^{4} s_j H_j(x) \tag{11.43}$$

where the $H_j(x)$, $j = 1, \ldots, 4$, are interpolation functions related to Hermite polynomials:

$$H_1(x) = \frac{1}{4} \left(2 - 3\xi + \xi^3 \right) \tag{11.44}$$

$$H_2(x) = \frac{L_e}{8} \left(1 - \xi - \xi^2 + \xi^3 \right) \tag{11.45}$$

$$H_3(x) = \frac{1}{4}\left(2 + 3\xi - \xi^3\right) \tag{11.46}$$

$$H_4(x) = \frac{L_e}{8}\left(-1 - \xi + \xi^2 + \xi^3\right) \tag{11.47}$$

$$\tag{11.48}$$

$L_e = x_r - x_l$ is the element's length, and $\xi = \frac{2(x-x_l)}{L_e} - 1$ is a scaled horizontal coordinate that is -1 at $x = x_l$ and $+1$ at $x = x_r$. The interpolation functions convey information about the interpolated function's value and first-derivatives at the two endpoints (nodes) of the line-segment element, and are displayed on the interval $-1 < \xi < 1$ in Figure (12.1). The coefficients s_j, for $j = 1, 3$, represent nodal values of $s(x)$ and s_j, and coefficients s_j, for $j = 2, 4$, represent information about the nodal values of $\frac{\partial s}{\partial x}$. Observe that, in the above notation, the subscript j refers to a local node-numbering scheme and that coefficients s_j vary from element to element according to the global element connectivity scheme.

The variation of T with x and z is represented by bi-directional linear interpolation (*i.e.*, linear in each separate direction):

$$T(x, y) = \sum_{j=1}^{2} \left(\sum_{k=1}^{2} T_{jk} L_j(x) \right) L_j(z) \tag{11.49}$$

where T_{jk} is the nodal value of $T(x, z)$ at the j'th node (line-segment endpoint) in the horizontal element and at the k'th node (line-segment endpoint) in the vertical element. Because a split timestep is used (see below), updates to T_{jk} are made first holding j fixed and next holding k fixed. This scheme avoids the necessity of integrating interpolation functions $L_j(x)L_k(z)$ over the 3-dimensional "brick" volume determined by the 8 corners where nodal values of T are resolved, *i.e.,* at the 8 points (x_j, z_k), $j = 1, 2$ and $k = 1, 2$. The linear interpolation functions $L_j(x)$ and $L_k(z)$ are of the form

$$L_j(x) = \alpha_j x + \beta_j \tag{11.50}$$

$$L_k(z) = \alpha_k z + \beta_k \tag{11.51}$$

where the coefficients α_m and β_m are determined by

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix}^{-1} \tag{11.52}$$

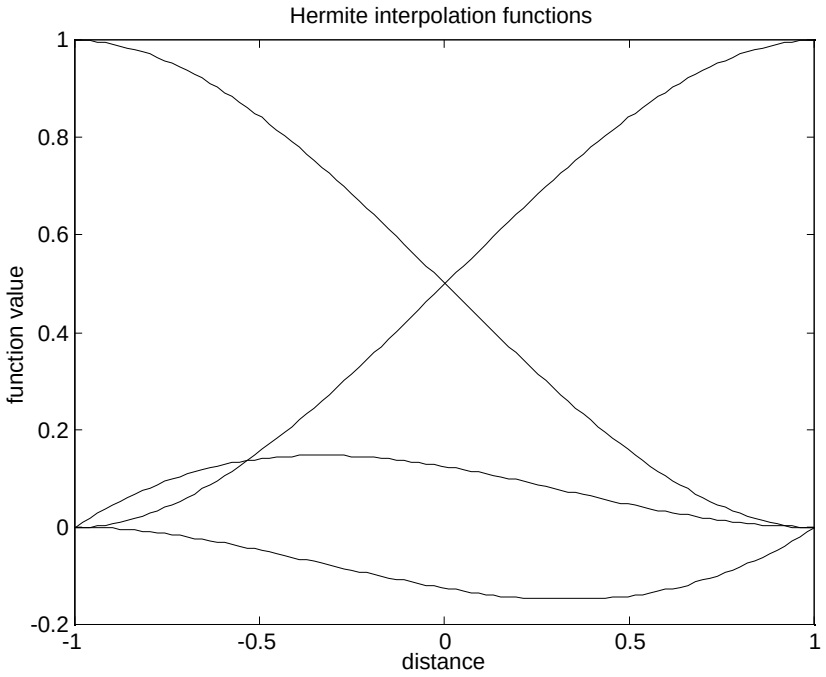

Figure 11.1: To permit evaluation of $\frac{\partial^2 s}{\partial x^2}$, interpolation functions based on the Hermite polynomials are employed. Four functions are used on each element. Two functions convey the value of the interpolated field at each endpoint (node) of the line segment (element), and two functions convey information about the value of the first-derivative of the interpolated field at each endpoint. The two interpolation functions that are zero at endpoints (nodes) are scaled by L_e, the length of the element, in this particular diagram.

or

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{bmatrix} = \begin{bmatrix} z_1 & 1 \\ z_2 & 1 \end{bmatrix}^{-1} \tag{11.53}$$

and where (x_1, x_2) and (z_1, z_2) are the nodal coordinates (line segment endpoints) in the horizontal and vertical discretizations, respectively.

Integrals of products of interpolation functions, *e.g.*, $\int_{x_1}^{x_2} H_j(x)H_l(x)dx$ are evaluated using either exact integration formulae or Gaussian quadrature (see Chapter 9). When Gaussian quadrature is used, the integrals are approximated by sums as follows:

$$\int_{x_1}^{x_2} H_j(x)H_l(x)dx \approx \frac{L_e}{2} \sum_{g=1}^{N_g} \mathcal{W}_g H_j(\xi_g)H_l(\xi_g) \tag{11.54}$$

The $N_g = 5$ quadrature points are, in the example presented in this chapter, $\xi_g \in \{-.9061798459\ -.5384693101\ 0.0\ .5384693101\ .9061798459\}$, and the weighting factors are $\mathcal{W}_g \in \{.2369268851\ .4786286705\ .5688888888\ .4786286705\ .2369268851\}$. Exact integration formulae are listed in Hulbe (personal communication).

Time stepping of the heat- and mass-balance equations is accomplished using a finite-difference representation of the time derivatives, *e.g.*,

$$T_t = \frac{T^{n+1} - T^n}{\Delta t} \tag{11.55}$$

where n is the discrete time level, and Δt is the timestep size. Vertical and horizontal derivatives are involved in the heat-balance equation, thus we employ a split timestep to simplify integration of Eqn. (12.43):

$$\frac{\tilde{T}^{n+1}}{\Delta t} + u^n(\zeta_l)\tilde{T}_x^{n+1} + \mathcal{S}(\tilde{T}^{n+1}; \zeta_l) = \frac{T^n}{\Delta t} \quad \forall l \tag{11.56}$$

$$\frac{T^{n+1}}{\Delta t} - \frac{1}{\rho c h^2} \left(k T_\zeta^{n+1} \right)_\zeta + \frac{\omega(\zeta)}{h} T_\zeta^{n+1} = \frac{\tilde{T}^{n+1}}{\Delta t} + \frac{W(\zeta)}{\rho c} \quad \forall k \tag{11.57}$$

where in the first of the above equations, l is the vertical level number, ζ_l is the l'th vertical level in the ice, and $\omega(\zeta)$ is the vertical velocity in the "ζ-system" given by:

$$\omega = D(\zeta) - \zeta \dot{A} - (1 - \zeta)\dot{B} \tag{11.58}$$

The term $\mathcal{S}(\tilde{T}^{n+1};\zeta_l)$ refers to an artificial horizontal diffusion term that can be either "upwinding" related or the "streamline upwind Petrov-Galerkin" (SUPG) term described in Chapter 9. This artificial diffusion term is used to supress numerical noise (wiggles) generated in high Peclet number flows (see Chapter 9).

11.4 EISMINT Level 1 Fixed Margin Intercomparison Benchmark

To test the above described flowline model, and to compare it with other finite-element and finite-difference models of a similar nature, we rerun the fixed margin intercomparison benchmark described in Chapters 2 and 9. Recall that the fixed-margin intercomparison test [now described fully by Huybrechts and others, in press, Annals of Glaciology 23] imposes fixed $\dot{A}$, k, c and a surface temperature (in Kelvin) given by

$$T_s = 239 + 8 \times 10^{-8} d^3 \tag{11.59}$$

where $d = |x|$, in a domain centered at $x = 0$ that extends 1500 km to the sides, with zero ice thickness at the margins. Thermomechanical coupling, bedrock, and polythermal ice treatments are artificially supressed. Horizontal divergence in the direction transverse to the flowline is assumed zero. This assumption constitutes an important constraint on the vertical velocity field and, through advection, the temperature field.

The value of parameters used in this intercomparison experiment are summarized as follows:

parameter	value
$\dot{A}$	0.3 m a^{-1}
A	10^{-16} Pa^{-3} a^{-1}
g	9.81 m s^{-2}
ρ	910 kg m^{-3}
k	2.1 W m^{-1} K^{-1}
c	2009 J kg^{-1} K^{-1}
T_o	273.15 K
Φ	8.7×10^{-4} K m^{-1}
G	42 mW m^{-2}
31556926	s a^{-1}

Horizontal resolution is fixed at 50 km. Vertical resolution is variable, and is designed to represent the temperature profile in the bottom 10% of the ice column with a logarithmic spacing of nodes (finest spacing at the bottom). Spacing is linear in the upper 90% of the ice column. In the present test, 31 levels are used to resolve the vertical variation of temperature, velocity and other parameters. Twenty levels are used to resolve the lowest 10% of the ice column.

11.4.1 Results

According to the model intercomparison instructions, the model is run until steady state conditions prevail. Temperature, velocity, mass flux and ice thickness data are then sampled at specific points and along the flowline. In particular, temperature and vertical velocity profiles at the ice divide and "midpoint" node are recorded for the intercomparison. Ice thickness, horizontal ice flux, and homologous basal temperature are presented along the flowline and are recorded for intercomparison with the results of the 3-dimensional model in Chapter 9.

Flowline sections of ice thickness and horizontal ice flux are displayed in Figs. (11.2) and (11.3). Homologous basal temperature along the same cross sections is displayed in Fig. (11.4). Ice flux at the midpoint of the flowline (half way to the ice margin from the divide) is 1.1261×10^5 m^2 a^{-1}, and

basal and homologous basal temperatures at the ice divide are -9.9625 C and -6.8830 C, respectively. Temperature profiles at the divide and midpoint are shown in Fig. (11.5).

Numerical values of various derived quantities are provided in the following table (other numerical data is available on request):

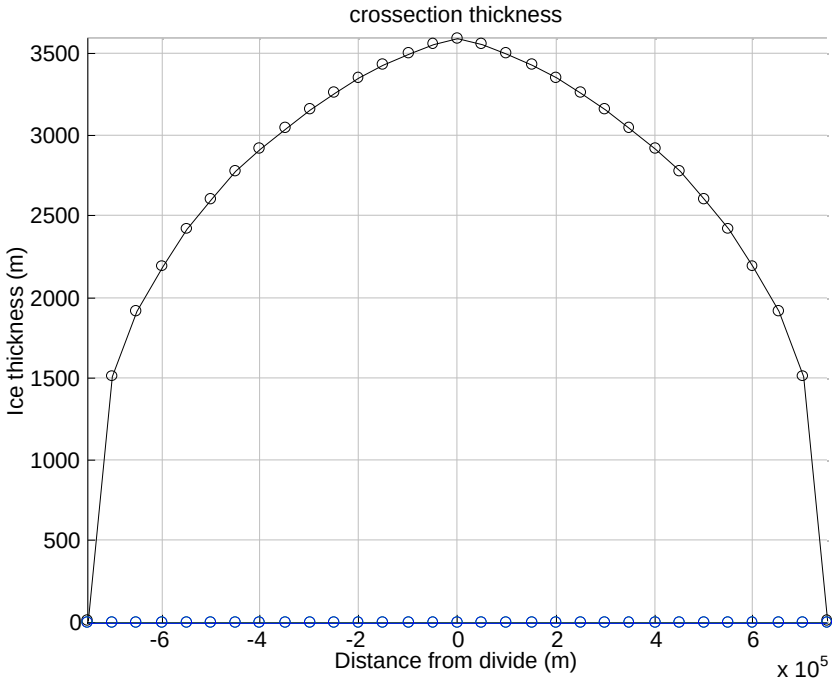

Figure 11.2: Ice thickness along the flowline. Maximum ice thickness at the ice divide is 3599.0 m, a value larger than the 3387.4 m thickness produced by the 3-dimensional ice-sheet model in Chapter 9. This thickness difference is due primarily to the fact that lateral ice divergence along the flowline is zero, whereas cross-sections through a square, plan view, domain are not. The analytical value of ice thickness at the divide, according to Huybrechts and others (in press), is 3575.1 m.

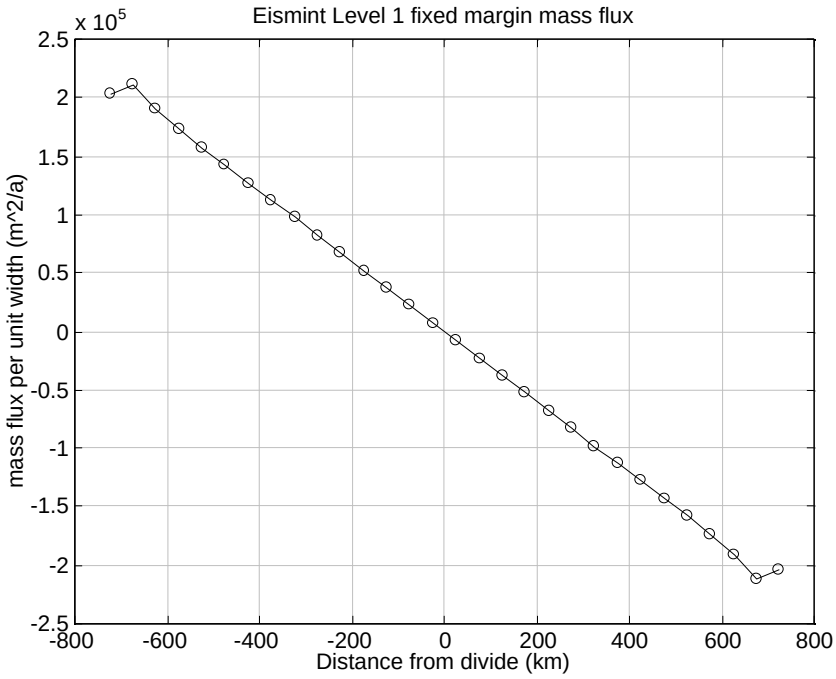

Figure 11.3: Horizontal depth-integrated ice flux. The results compare favorably with the exact analytic flux, which should be 2.25 $\times 10^5$ m^2 a^{-1} at the ice margin. Ice flux is resolved at element midpoints in the present example.

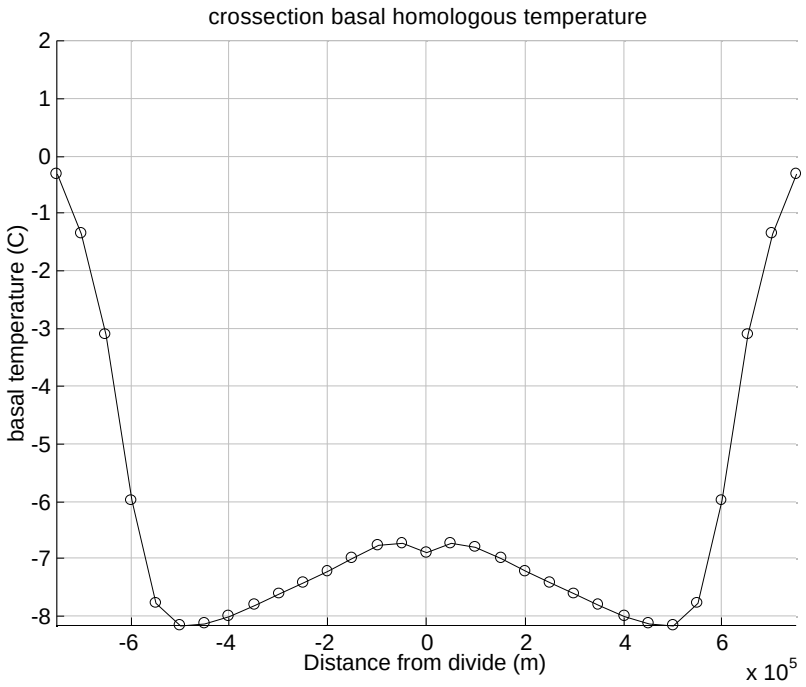

Figure 11.4: Homologous basal temperature along flowline.

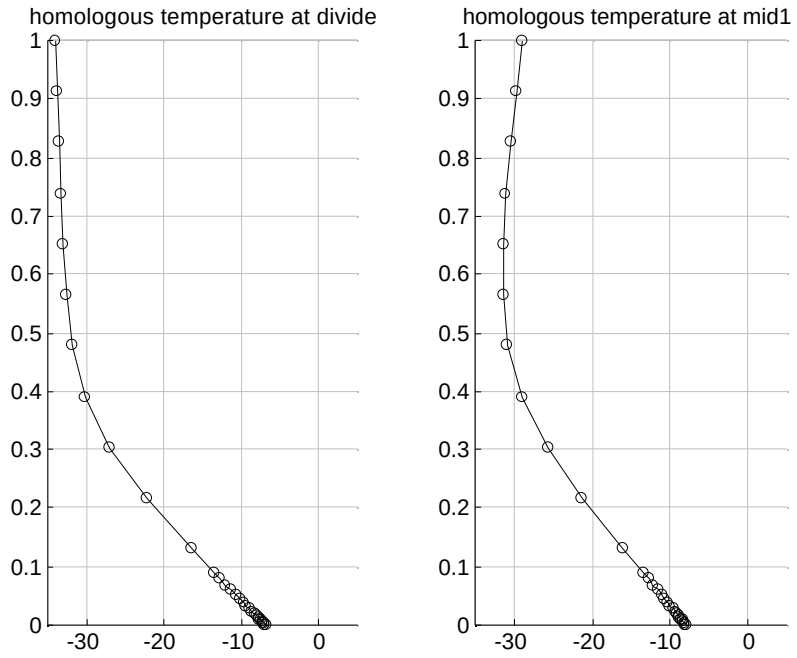

Figure 11.5: Homologous temperature profiles at ice divide, midpoint and ice margin.

ζ	T^*_{divide}	T^*_{midpoint}
0	-6.8830	-7.9853
0.0013	-6.9797	-8.0639
0.0027	-7.0889	-8.1526
0.0044	-7.2122	-8.2526
0.0062	-7.3513	-8.3656
0.0083	-7.5084	-8.4932
0.0107	-7.6857	-8.6372
0.0134	-7.8858	-8.7998
0.0164	-8.1117	-8.9834
0.0198	-8.3667	-9.1908
0.0236	-8.6544	-9.4250
0.0279	-8.9792	-9.6897
0.0328	-9.3457	-9.9887
0.0383	-9.7593	-10.3266
0.0446	-10.2258	-10.7087
0.0516	-10.7519	-11.1408
0.0595	-11.3450	-11.6295
0.0685	-12.0131	-12.1825
0.0786	-12.7650	-12.8082
0.0900	-13.6100	-13.5161
0.1300	-16.5227	-16.0011
0.2170	-22.3486	-21.2358
0.3040	-26.9724	-25.7363
0.3910	-30.0642	-28.9767
0.4780	-31.7823	-30.8134
0.5650	-32.6143	-31.5160
0.6520	-33.0397	-31.5121
0.7390	-33.3375	-31.1299
0.8260	-33.6101	-30.5258
0.9130	-33.8800	-29.7449
1.0000	-34.1500	-29.0300

where T^* refers to homologous temperature (temperature relative to the local pressure melting temperature).

Model Intercomparisons

Huybrechts and others [in press] present an intercomparison of a number of finite-difference model results using the intercomparison benchmark we present above. The model results presented by Huybrechts and others are similar to those developed here. There are several points to be noted in the comparison:

1. Basal temperatures in the 2-dimensional model differ significantly from the the 3-dimensional model results presented in Chapter 9 and by Huybrechts and others [in press] (see Fig. 11.5). Under the ice divide, basal temperature is warmer than in the 3-dimensional result. This difference is due to the fact that the ice divide is thicker in the 2-dimensional result than in the 3-dimensional result by about 200 m. More interestingly, the 2-dimensional results display frozen basal conditions all the way to the ice margin, whereas the 3-dimensional results have frozen conditions persisting only about half the distance to the ice margin (see Chapter 9). This difference results from the fact that horizontal velocity (which influences advective cooling) is larger in the 2-dimensional case than in the 3-dimensional case (see Fig. 11.6).

2. Ice divide thickness in the flowline model presented here is greater than that in the 3-dimensional model of Chapter 9. This is due to the lack of lateral divergence along the flowline which is present in the 3-dimensional model due to the square domain.

11.5 Comparisons: Strain Heating and Polythermal Ice

To illustrate the influence of strain heating, we repeat the EISMINT level 1 (fixed margin) experiment with strain heating and polythermal ice conditions accounted for in the customary manner (*e.g.*, see Chapter 10; Greve, 1995). Because bedrock thermal coupling is disregarded and the intercomparison

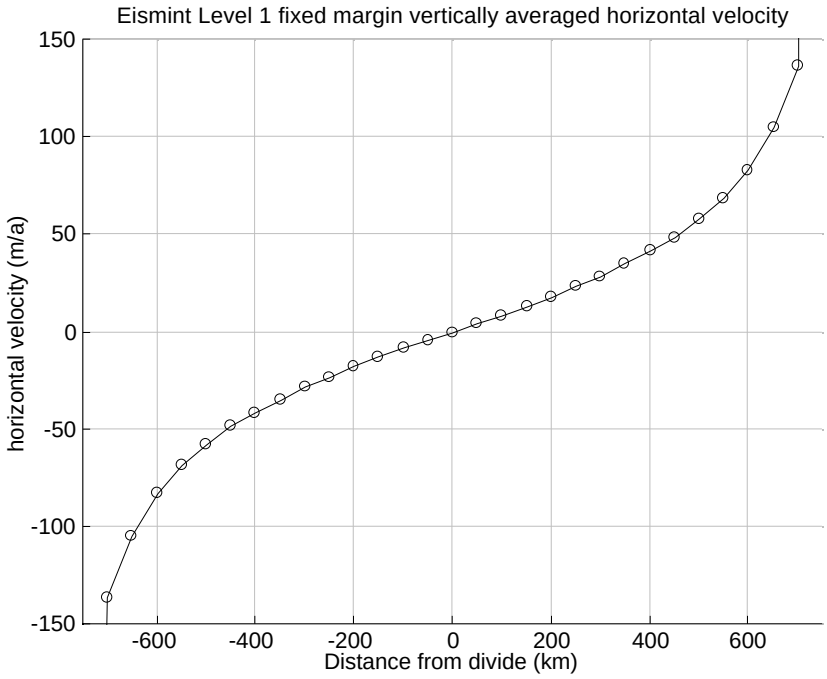

Figure 11.6: Depth-averaged horizontal velocity as a function of distance from the ice divide. This profile can be compared with Fig. (2) of Huybrechts and others (in press) to show that the horizontal velocity in 2-dimensional flowline models is greater than that of the 3-dimensional models in the EISMINT Level 1 test. The greater horizontal velocity introduces larger advective cooling of the bed (vertical velocity is larger, horizontal velocity is larger) which explains the difference between the 2-dimensional and 3-dimensional results for basal temperature in the EISMINT Level 1 fixed margin test.

results are to be in steady state, the development of a temperate ice layer is assumed to grow from the bed of the ice sheet upward into the ice column. (Recall that bedrock thermal inertia permits the possibility of temperate ice developing first above the bed, and subsequently growing down to the bed from above.) In keeping with the notion of no thermomechanical coupling, the mass balance of the ice sheet is not adjusted for liquid water drainage from the temperate layer. Basal melting, furthermore, is artificially taken to be zero.

11.5.1 Governing Equations

Polythermal ice conditions are computed using techniques of Chapter 10 modified to allow for horizontal advection (and appropriate specification of velocity and strain heating fields). To facilitate advection across junctions between cold and polythermal ice columns, we employ two stretched vertical coordinates and two temperature functions: $\zeta = \frac{z-b}{s-b}$ spans the entire ice column, $\zeta_c = \frac{z-c}{s-c}$ spans the portion of the ice column above the cold/temperate ice transition at $z = c(x,t)$, $T(x,\zeta,t)$ represents the temperature in the region $b < z < s$, and $T_c(x,\zeta_c,t;c)$ represents the temperature in the region $c(x,t) < z < s$.

In regions (in the present example, near the ice divide) where $c(x,t) = b$, *i.e.,* where there is no temperate ice layer, the following equations are solved for $T(x,\zeta,t)$ and $T_c(x,\zeta_c,t;c=b)$:

$$\begin{aligned} & T_t + uT_x \\ & + \frac{1}{h} T_\zeta \left(\mathcal{D}(\zeta) - \zeta \dot{A} - (1-\zeta)\dot{B} \right) \\ & = \frac{1}{\rho c h^2} \frac{\partial}{\partial \zeta} \left(k \frac{\partial T}{\partial \zeta} \right) + \frac{W}{\rho c} \end{aligned} \tag{11.60}$$

and

$$T_c(x,\zeta_c,t;c=b) = T(x,\zeta,t) \tag{11.61}$$

Boundary conditions for the above equations consist of specified surface temperature and either specified basal temperature gradient (if the basal temperature is below the freezing point) or specified basal temperature.

In regions (in the present example, near the ice margin) where $c(x,t) > b$, it i.e., where there is a basal temperate ice layer, the governing equations are:

$$(T_c)_t - \frac{1}{h_c}(T_c)_{\zeta_c}\left(\zeta_c \dot{A} - (1-\zeta_c)a_\perp\right) = \frac{1}{\rho c h_c^2}\left(k(T_c)_{\zeta_c}\right)_{\zeta_c} + \frac{W(z(\zeta_c))}{\rho c} \quad (11.62)$$

$$T(x,\zeta(\zeta_c),t) = T_c(x,\zeta_c,t;c) \quad (11.63)$$

for $c(t) < z < s$ $(0 < \zeta_c < 1)$, and

$$T(x,\zeta,t) = T_m(x,\zeta,t) \quad (11.64)$$

for $\zeta < \frac{c-b}{s-b}$, where T_m is the pressure melting temperature. Surface temperature is applied as a boundary condition at $\zeta_c = 1$ and

$$T_c(z=c,t) = T_m(z=c,t) \quad (11.65)$$

at $\zeta_c = 0$. An additional requirement is made of the solution at $z = c$ if ice is transported from the cold region to the temperate region through the cold/temperate ice transition surface, *i.e.,* $a_\perp < 0$:

$$(T_c)_z(z=c,t) = \rho g \Phi \quad (11.66)$$

To satisfy this requirement, the model uses a control method (see Chapter 10) to choose the value of $a_\perp$ (which is less than 0) so as to make the specification of temperature and temperature gradient at $z = c$ possible. If ice is transported from the temperate region toward the cold region through the cold/temperate ice transition surface, *i.e.,* $a_\perp > 0$, then no additional requirement on the temperature gradient at $z = c$ is made. In this circumstance, the value of $a_\perp$ is chosen to satisfy energy-continuity constraints associated with freezing of liquid water at a surface where there is a jump condition in temperature gradient (and, hence, in heat flow):

$$a_\perp = \frac{-k\left(T_z|_{c_+} - T_z|_{c_-}\right)}{\rho L_f \mu} \quad (11.67)$$

where $\left(T_z|_{c_+} - T_z|_{c_-}\right)$ expresses the temperature-gradient jump condition at $z = c$, and μ is the liquid water fraction of the temperate ice passing through the cold/temperate ice transition. In the example presented here, $a_\perp < 0$;

thus the liquid-water content of temperate ice is not computed because it is not needed to determine the temperature in the cold ice region (*i.e.,* a jump condition in temperature gradient is replaced by the requirement $(T_c)_z(z = c, t) = \rho g \Phi$).

To compute the location of the cold/temperate ice transition, $c(x, t) = b(x) + h_T(x, t)$, we use the following kinematic equation:

$$c_t + u(z = c)c_x = -a_\perp + w(z = c) \tag{11.68}$$

As remarked above, $a_\perp$ is chosen using a control method to satisfy the twin (overdetermined) boundary conditions at $z = c$.

11.5.2 Results

Results of the test are displayed in Figs. (11.7) - (11.9). The effect of strain heating, *i.e.,* $W \neq 0$ according to Eqn. (12.25), is to produce melted basal conditions and to introduce temperate ice layers at the margins of the ice sheet. The difference between the results shown here and those of the case of no strain heating (previous sections) are large and point to the importance of strain heating in determining the degree to which melting occurs at the margins of large ice sheets.

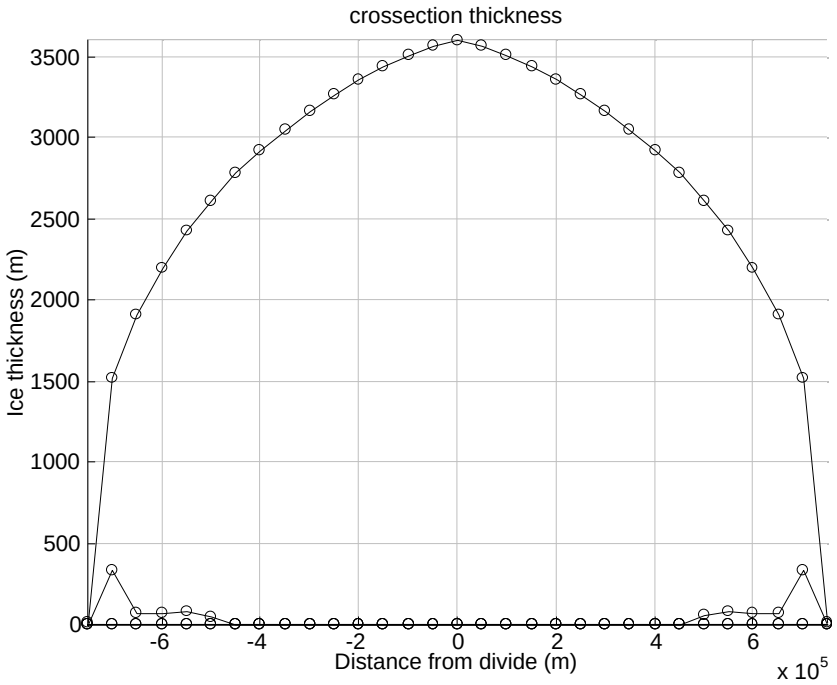

Figure 11.7: Ice and temperate-ice thickness for the EISMINT Level 1 (fixed margin) test in which polythermal ice conditions were allowed to develop.

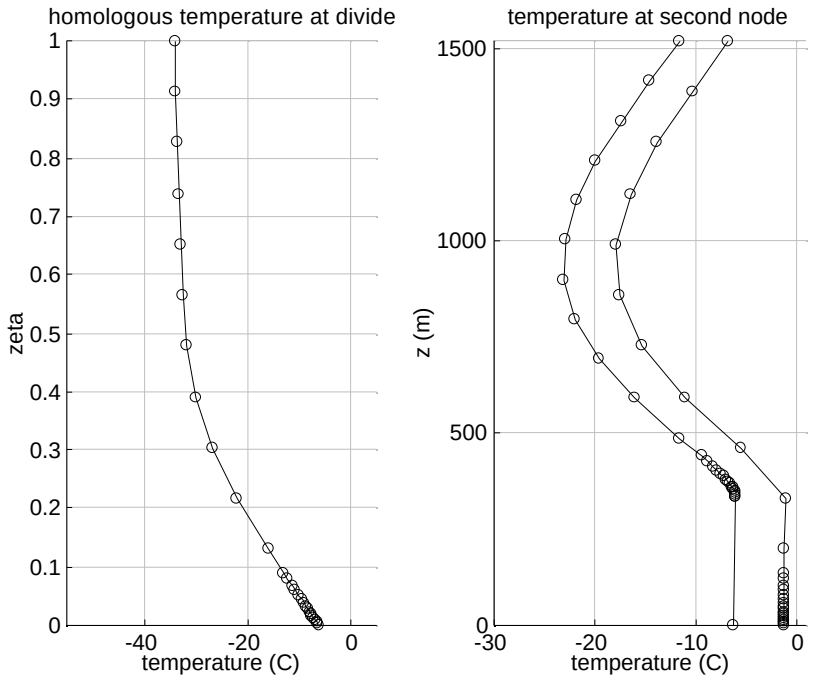

Figure 11.8: Ice divide temperature profile (left) and temperature profile at the second node (where the temperate ice layer is 333 m thick) (right). In the right panel, two temperature profiles are shown. The left-most profile in the right panel represents T_c where node configuration is optimized for the cold ice region (and is concentrated at the cold/temperate ice transition), and is offset by -5 C for display. The right-most profile in the right panel represents T where node configuration is distributed through the ice column (and is concentrated at the base of the ice). Observe that the vertical temperature gradient of T_c at $z = c$ is equal to the Clapyron slope, $\rho g \Phi$.

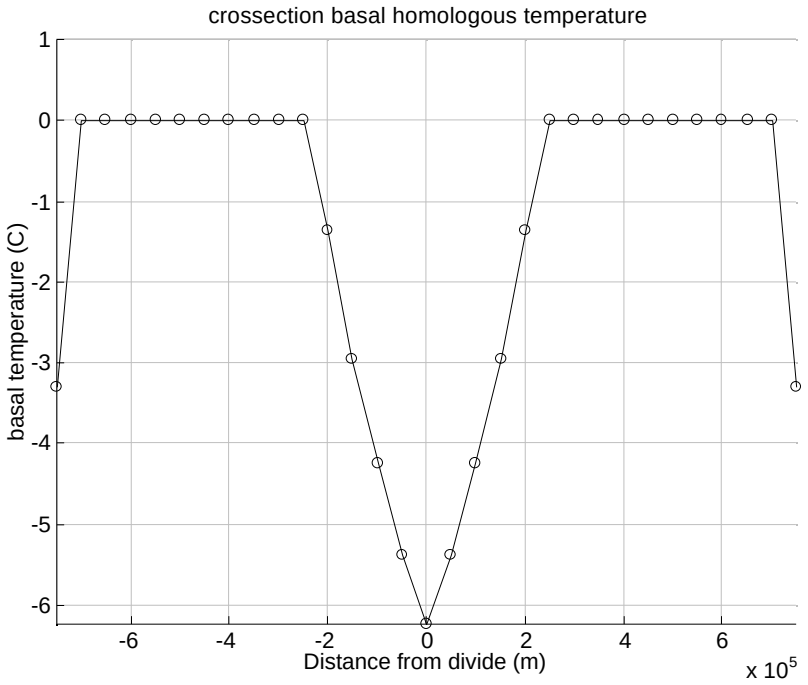

Figure 11.9: Basal homologous temperature in circumstances where strain heating and polythermal ice development are accounted for in the EISMINT Level 1 (fixed margin) experiment.

Chapter 12

Greenland Model: 2-D (Flowline) Techniques

In this chapter, we report the reconstruction and testing of a new and improved two-dimensional, "flowline" ice-sheet model developed jointly by a Pennsylvania State University/University of Chicago team during Spring of 1997 (Byron Parizek and Doug MacAyeal). This model is an improvement of the one described in Chapter 11. The new psu/uc model addresses several bugs and misconceptions associated with Chapter 11, and makes substantial improvements to the "vectorization" of the matlab code.

The psu/uc model is tested here using the EISMINT level 2 "Greenland" suit of tests developed by Catherine Ritz and the Grenoble team.

12.1 Synopsis (Info File)

Following the list of questions provided by Catherine Ritz, the EISMINT Greenland test coordinator, we offer the following summary of the flowline model design.

1. In referring to our model as a "flowline"-model, we mean that it is 2-dimensional in x and z, the horizontal and vertical coordinates. All transverse horizontal structure is assumed zero, in particular: $v = 0$ and $\frac{\partial v}{\partial y} = 0$ where v is the transverse (y) velocity, and y is the horizontal coordinate transverse to x. Being 2-dimensional, the x-axis defines the horizontal projection of flowlines in the model. This does not mean that the particular transect through the ice sheet (in the present study, the GRIP transect) is a flowline for a fully 3-dimensional ice-sheet model or in real life.

2. The model domain is the GRIP transect ($J = 77$ in the EISMINT data sets provided by the testing coordinator, Catherine Ritz).

3. We have not published a paper describing this model. This write-up is intended for MacAyeal's *Lessons in Ice-Sheet Modelling*, and will be revised to appear in Byron Parizek's MS thesis for Pennsylvania State University (Richard Alley, advisor). The numerical methodology is a high-order finite-element method, however much of the spirit of the flowline model is based on the finite-difference model described by Payne (1995). A few notes:

   - This model is capable of dealing with polythermal ice conditions in the manner described by Greve (1996); (see previous chapter of MacAyeal's *Lessons*). We have not implemented or tested the polythermal aspects of this model in the context of the Greenland Eismint experiments.
   - We do not treat floating ice. Thus, if conditions permit, the flowline model presented here could advance the margin of the Greenland ice sheet into the bottom of the Labrador Sea! We limit the ice-sheet advance to the contour where present depth below sea level is 250 m.
   - We compute temperatures with horizontal and vertical advection, strain heating and vertical conduction. We allow ice to reach the melting point within the ice column (temperate ice), but we do not incorporate the conditions at the Cold/Temperate Transition Surface that are derived by Greve and his teachers. (The CTS in our model has an unphysical discontinuity in temperature gradient

from the pressure-melting depression "β" in the temperate ice, to the value predicted by the vertical heat conduction/advection equation above the temperate ice.)

- We believe that we treat snow and superimposed ice differently from other EISMINT investigators (we are not sure). In particular, we solve advective continuity equations for snow thickness and superimposed ice thickness, h_s and h_x, respectively:

$$\frac{\partial h_s}{\partial t} + \frac{\partial u_s h_s}{\partial x} = \dot{S} \tag{12.1}$$

and,

$$\frac{\partial h_x}{\partial t} + \frac{\partial u_s h_x}{\partial x} = \dot{X} \tag{12.2}$$

where we assume that both snow and superimposed ice advect horizontally with the surface flow, $u_s = u(z = s)$, of the ice sheet. (Note: The EISMINT Level 1 test has an error in the code which we have just caught. Instead of solving the above equations, we actually solved the following equations:

$$\frac{\partial h_s}{\partial t} + u_s \frac{\partial h_s}{\partial x} = \dot{S} \tag{12.3}$$

and,

$$\frac{\partial h_x}{\partial t} + u_s \frac{\partial h_x}{\partial x} = \dot{X} \tag{12.4}$$

this error will lead to slightly thicker snow layers due to lack of flow divergence as a thinning effect. We will correct this mistake later. We do not believe that our treatment of snow and superimposed ice will yield significant differences from other treatments which do not include horizontal movement of the surface of the ice sheet; we are thus not compelled to correct the mistake noted above for the EISMINT intercomparison tests.)

- We believe that we avoid "ill-posedness" in the specification of surface boundary conditions in the heat balance equation. In ablation areas where ice is moving upward through the ice sheet surface, we do *not* specify the surface temperature in circumstances where the ice is sufficiently thick (over 250 m) that the expected thermal

boundary layer at the surface could not be resolved. Information flow, *i.e.*, characteristics of the hyperbolic problem, is directed out of the ice sheet; thus, the equations without specification of surface temperature are well-posed, and the solution is smooth (albeit colder than the actual surface temperature achieved with a boundary layer). In particular we specify:

$$T(s) = \begin{cases} T_s & \text{if } \dot{A} > 0 \\ \\ \text{not specified} & \text{if } \dot{A} \leq 0 \text{ or } h \geq 250 \text{ m} \end{cases} \tag{12.5}$$

where $h = s - b$ is the ice thickness, s is surface elevation of the ice-sheet surface, T is temperature, and T_s is the atmospheric annual average temperature.

- In the Level 1 experiment, we assume that the bed is perfectly drained and that water is not available for basal freezing.

4. We have not submitted Level 3 results yet.

5. Horizontal resolution is 20 km. Flowline divergence in the transverse direction is assumed zero. We do not use staggered grids. All matrix equations are solved with exact matrix factorization and backsubstitution. We do not use upwind differencing or artificial diffusion in the ice balance and thermal balance equations. We do use upwind-like artificial diffusion in the snow and superimposed ice advection equations (to avoid ill-posed boundary conditions needed to keep snow and superimposed ice positive at the edges of the snow covered domain).

6. Vertical resolution is variable. We use 25 nodes to cover the ice column, 10 of which are located in the lowest 10% of the ice column with logarithmic spacing (the first ice node is located $0.0029 \times h$ above the bed, and the second to last ice node is located $0.09379 \times h$ below the ice surface, where h is the ice thickness). We cover 750 m of bedrock with 6 nodes, and with a vertical resolution of 150 m.

7. Thermodynamic coupling: advection, diffusion, strain heating and a crude variant of polythermal ice conditions (one which does not conserve energy) are employed. The geothermal heat flux boundary condition is applied at the bottom of the bedrock, which is assumed 750

m thick. We do not have "upwind" differencing type artificial diffusion in any part of our code. We note that this can sometimes yield noisy results; however, we believe that the results with such noise are of interest (*i.e.,* both the noisy results and the artificially smoothed results have equal, yet offsetting, inadequacies. When looked at together, they form complementary "pictures" of the solution.)

8. We follow Huybrechts (1991) in developing our ablation parameterization, however, we extend his parameterization by presuming that the snow layer is perrenial and thus subject to the advection and vertical strain rate associated with ice-sheet flow. (In our simulations, we made an error and only accounted for advection. Vertical strain rate in the snow layer, which contributes to thinning was not forgotten by accident.) We do not expect this possible improvement in the Huybrechts (1991) parameterization to have much effect on model results (since the transition between snow-covered ice and snow-free ice is so abrupt that it is not resolved by the 20 km node spacing). Nevertheless, we decided to impliment the idea as a means of contributing toward the further development of Greenland ice-sheet modelling (particularly if models are required to predict snow facies or ice-core firn effects, such as pore close off depths).

9. We follow Huybrechts (1991) and the EISMINT intercomparison instructions provided by Catherine Ritz, coordinator, for our accumulation parameterization.

10. Bedrock is modelled with a diffusive equation following Payne (1995) using his asthenospheric diffusivity $D_a = 10^8$ m^2 a^{-1}.

11. Sea level drops only influence the accumulation and ablation parameterizations by "moving the atmosphere " up and down. (The ice surface instantly becomes higher when the sealevel instantly drops.)

12. We do not parameterize calving. We require that the ice not advance beyond the -250 m bed contour of the present bed elevation, and make no adjustment of this criterion for changes in sea level.

12.2 Dynamics

The governing equations for the dynamic/thermodynamic flowline ice-sheet model developed here for the purpose of performing the EISMINT Level 2 Greenland Experiments are listed as follows. The equations which govern the horizontal and vertical velocity fields are:

$$u(z) = -2\rho g \frac{\partial s}{\partial x} I(z) \tag{12.6}$$

$$w(z) = -\int_b^z \frac{\partial u}{\partial x} dz' + \frac{\partial b}{\partial t} + u(b)\frac{\partial b}{\partial x} - \dot{B} \tag{12.7}$$

where x is the horizontal coordinate directed down the flowline, z is the vertical coordinate (positive up), u is the horizontal velocity directed down the flowline, w is the vertical velocity, s and b are the surface and basal elevation, respectively, ρ is the ice density (assumed to be constant, 910 kg m^{-3}), g is the acceleration of gravity (9.81 m s^2), and $\dot{B}$ is the basal melting rate (positive for melting, negative for freezing, in meters of ice equivalent per second). To keep the treatment simple, lateral flow divergence along the flowline is not accounted for in the expression for the vertical velocity. Lateral divergence influences the vertical velocity (makes it larger) and is expected to yield a cooler temperature-depth profile than that which is computed by our model, which lacks lateral divergence. Basal sliding is suppressed following the EISMINT Level 2 instructions.

The temperature $T(x, z, t)$ is assumed to cover both ice and a 750 m layer of bedrock (in ice-free zones, the temperature at vertical grid points reserved for ice are set automatically to the surface atmospheric temperature). We do not account for polythermal conditions, however we require T/geT_m where T_m is the z-dependent melting point,

$$T_m(z) = T_o - \rho g(s - z)\Phi \tag{12.8}$$

where $\Phi = 8.7 \times 10^{-4}$ K Pa^{-1}, and $T_o = 273.15$ K.

The factor $I(z)$ determines the deformational velocity of the ice flow, and

for Glen's flow law (with exponent 3) is defined by

$$I(z) = \int_b^z EA(T^*)(\rho g)^2 \left(\frac{\partial s}{\partial x}\right)^2 (s - z')^3 \, dz' \tag{12.9}$$

The rate-enhancement factor E is designed to account for empirically determined inadequacies of Glen's law as applied to Greenland Ice Sheet modelling. The value of E is held fixed at 3 for the EISMINT Greenland experiments. The creep-rate factor $A(T^*)$ is assumed to have a standard thermodynamic form involving a rate constant factor a and an activation energy Q:

$$A(T^*) = a \exp\left(\frac{-Q}{RT^*}\right) \tag{12.10}$$

where $R = 8.314$ J mol^{-1} K^{-1} is the gas constant, and the flow-law parameters (determined from laboratory studies of polycrystaline ice assumed to be isotropic) are commonly taken to be:

$$a = \begin{cases} 1.14 \times 10^{-5} \text{ a}^{-1} \text{ Pa}^{-3} & \text{if } T^* < 263.15 \text{ K} \\ \\ 5.47 \times 10^{10} \text{ a}^{-1} \text{ Pa}^{-3} & \text{if } T^* \geq 263.15 \text{ K} \end{cases} \tag{12.11}$$

and

$$Q = \begin{cases} 6.0 \times 10^4 \text{ J}^{-1} \text{ mol}^{-1} & \text{if } T^* < 263.15 \text{ K} \\ \\ 13.9 \times 10^4 \text{ J}^{-1} \text{ mol}^{-1} & \text{if } T^* \geq 263.15 \text{ K} \end{cases} \tag{12.12}$$

The temperature in the creep-rate factor formula $T^* = T - T_m + T_o$ is the Homologous temperature (temperature relative to the pressure melting point) in degrees Kelvin (note the addition of T_o).

Mass balance of the ice sheet is expressed by the following equation for ice thickness h,

$$\frac{\partial h}{\partial t} = -\frac{\partial q}{\partial x} + \dot{A} - \dot{B} \tag{12.13}$$

where $q = D\frac{\partial s}{\partial x}$ is the horizontal mass flux integrated over the ice column, $\dot{B}$ is the basal melting rate (positive for melting, negative for freezing), and $\dot{A}$ is the surface snow accumulation rate expressed in meters of ice equivalent per

year. Note that ice densification effects are disregarded. A common practice is to substitute Equations (12.6) into Equation (12.13) giving,

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial s}{\partial x}\right) + \dot{A} - \dot{B} \tag{12.14}$$

where the effective diffusivity D is defined by,

$$D = \int_b^s 2\rho g I(z')dz' \tag{12.15}$$

Kinematic boundary conditions on the free surface $z = s$ and basal surface $z = b$ are recorded for use elsewhere:

$$\frac{\partial s}{\partial t} + u(s)\frac{\partial s}{\partial x} = w(s) + \dot{A} \tag{12.16}$$

and,

$$u(b)\frac{\partial b}{\partial x} = w(b) + \dot{B} \tag{12.17}$$

where we have made the assumption that $\frac{\partial b}{\partial t} = 0$ (no isostatic or bed erosion effects are included).

Ice thickness is assumed to be positive, *i.e.*, $h \geq 0$. To handle the advance and retreat of glacial ice, we define a nodal information variable that is equal to 1 for barren ground (or seabed, in this circumstance) and 0 for ice-covered ground. To define this variable, and to effectively determine the advance and retreat of ice-sheet margins (disconnected, separate ice sheets are allowed within the model domain), we perform the mass-balance time step twice. First, we assume that all nodes can contain glacial ice and that h is free to become negative. Second, we perform a time step with this assumption. Third, we then identify those nodes where $h \leq 0$ at the end of this tentative time step. Fourth, we retake the time step, but this time with specified $h = 0$ as boundary conditions for the nodes identified in the third step. The barren-ground nodal information variable is then updated for use elsewhere in the model (notably the thermodynamic subroutines, where nodes reserved for ice temperature profiles are set to the surface temperature over barren ground).

12.3 Firn-Layer Dynamics

To parameterize the surface accumulation and ablation rates according to the EISMINT Greenland Level 2 experiment design (see Huybrechts, 1991), it is necessary to account for surface snow and superimposed ice. We assume that "ice equivalent" snow thickness h_s and superimposed ice thickness h_x are governed by the following mass balance equations (we do not account for snow density, *i.e.,* all snow is delt with as ice equivalent material):

$$\frac{\partial h_s}{\partial t} + \frac{\partial u_s h_s}{\partial x} = \dot{S} \tag{12.18}$$

and,

$$\frac{\partial h_x}{\partial t} + \frac{\partial u_s h_x}{\partial x} = \dot{X} \tag{12.19}$$

where we assume that both snow and superimposed ice advect horizontally with the surface flow, $u_s = u(z = s)$, of the ice sheet.

The snow and superimposed ice thicknesses are assumed to be positive. This necessitates the following procedure for marching Equations (12.18) and (12.19) forward in time. First, we take a tentative time step assuming that h_s and h_x are free to become negative at any node. Second, we identify those nodes where h_s and/or h_x are less than or equal to zero. Third, we retake the time step with h_s and h_x specified to be zero as boundary conditions at these identified nodes. To avoid numerical noise in specification of a boundary condition were one is not required to solve the purely hyperbolic equation, we add artificial diffusion equivalent to the SUPG procedure discussed in previous chapters to both equations (this term is not represented above).

To prevent the thickness of snow and superimposed ice from exceeding a reasonable firn-layer thickness (ice equivalent, assumed each to be 25 m, for a combined firn/superimposed ice layer maximum thickness of 50 m ice equivalent), we artificially limit snow thickness to 25 m and parameterize the snow and superimposed ice accumulation rates, $\dot{S}$ and $\dot{X}$ respectively, according to the climate parameterization specified in Huybrechts (1991). We admit that this maximum thickness assumption is a crude idealization of actual firn-layer dynamics. To limit snow thickness, we set $h_s = 25$ m

whenever Equation (12.18) predicts a value of $h_s > 25$ m. The accumulation rate for glacial ice, $\dot{A}$, is adjusted (see below) to receive all snow accumulated as a "firn-densification" pass-through when $h_s = 25$ m.

The expressions for $\dot{S}$ and $\dot{X}$ are:

$$\dot{S} = \dot{P} - \alpha_s H_{pdd} \frac{\rho_{water}}{\rho} \tag{12.20}$$

and

$$\dot{X} = \begin{cases} \alpha_s H_{pdd} \frac{\rho_{water}}{\rho} & \text{if } h_s > 0 \text{ and } \alpha_s H_{pdd} \frac{\rho_{water}}{\rho} \leq \gamma \dot{P} \\ \gamma \dot{P} & \text{if } h_s > 0 \text{ and } \alpha_s H_{pdd} \frac{\rho_{water}}{\rho} > \gamma \dot{P} \\ \alpha_x H_{pdd} \frac{\rho_{water}}{\rho} & \text{if } h_s = 0 \end{cases} \tag{12.21}$$

where $\dot{P}$ is the precipitation rate (assumed to all fall as snow, in m a^{-1} of ice equivalent), H_{pdd} is the positive-degree-day factor (units d C) given by the climate parameterization (see below), and the factor $\frac{\rho_{water}}{\rho}$ is necessary to convert from units of ice equivalent to water equivalent. The factors $\alpha_s = 0.003$ m d^{-1} C^{-1} and $\alpha_x = 0.008$ m d^{-1} C^{-1} denote positive degree day ablation rates for snow and superimposed ice, respectively.

The idea behind the above equations is as follows (see Huybrechts, 1991). Snow is always able to accumulate or ablate according to the difference between the snow precipitation rate and the positive-degree-day parameterization of the snow melting rate. Superimposed ice accumulates only as the residual between the snow melting rate and the snow melt runoff rate. When the snow melt rate falls below $\gamma \dot{P}$, where $\gamma = 0.6$ is an empirical nondimensional factor, all snow melt is assumed to accumulate as superimposed ice (zero runoff). When the snow layer thickness is zero, superimposed ice is allowed to ablate at the rate specified by the positive-degree-day method using a factor α_x that is greater than α_s.

12.4 Thermodynamics

Heat flow continuity in both ice and underlying bedrock is expressed using the standard advective/diffusive equation with variable heat-flow parameters:

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + w\frac{\partial T}{\partial z} = \frac{1}{\rho c}\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right) + \frac{W}{\rho c} \tag{12.22}$$

for $b < z < s$; and

$$\frac{\partial T}{\partial t} = \frac{1}{\rho_r c_r}\frac{\partial}{\partial z}\left(k_r\frac{\partial T}{\partial z}\right) \tag{12.23}$$

for $z_r \leq z \leq b$, and where z_r is a level at a fixed elevation below the ice/bedrock interface (or seabed) where the geothermal flux gradient is applied as a boundary condition (see below). In the present study, we take z_r to be 750 m below the ice/rock interface (or the atmosphere/rock interface in ice-free regions). The above equations reflect the standard assumption that horizontal heat conduction is negligible in comparison with horizontal advection, vertical advection and conduction, and viscous heat dissipation. The horizontal and vertical velocities referred to in Equations (12.22) and (12.23) are described by Equations (12.6) and (12.7).

For the EISMINT Greenland tests, we take $k = 2.1$ W m^{-1} K^{-1} and $k_r = 3$ W m^{-1} K^{-1} and $c = 2009$ J kg^{-1} K^{-1} and $c_r = 1000$ J kg^{-1} K^{-1}. We assume constant density in the ice, $\rho = 910$ kg m^{-3}, and a dry homogeneous bedrock $\rho_r = 2700$ kg m^{-3}.

The term W in Eqn. (12.22) is the viscous heating term formally defined by

$$W = \sum_i \sum_j \dot{e}_{ij} T'_{ij} \tag{12.24}$$

where $\dot{e}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ is the strain rate, and T'_{ij} is the deviatoric stress. According to the common assumptions associated with grounded ice-sheet stress balance (*i.e.,* neglecting longitudinal stress), W can be written as:

$$W(z) = (\rho g)^4\, 2\left(\frac{\partial s}{\partial x}\right)^4 (s-z)^4\, EA(T^*) \tag{12.25}$$

Boundary conditions are specified at the top $z = s$ and bottom $z = z_r$ of the ice/rock column in the following manner. At the top, a surface temperature (depending on atmospheric conditions) is specified only under circumstances when the finite-element discretization in the vertical is capable of resolving any surface thermal boundary layers that may exist at the ice surface

$$T(s) = \begin{cases} T_s & \text{if } \dot{A} > 0 \\ \\ \text{not specified} & \text{if } \dot{A} \leq 0 \text{ or } h \geq 250 \text{ m} \end{cases} \tag{12.26}$$

where $h = s - b$ is the ice thickness. After considerable testing, we found that specification of a surface temperature in ablation zones ($\dot{A} \leq 0$) when ice was too thick ($h > 250$ m) led to numerical noise, because the vertical resolution of the finite-element node points was unable to resolve the very thin conductive boundary layer at the ice-sheet surface. The model is able to run adequately without specification of a surface temperature in ablation zones because the *characteristics* of the mixed parabolic/hyperbolic thermal equation are directed from the interior of the ice sheet outward toward the atmosphere.

At the ice/bedrock interface, a geothermal temperature gradient is specified

$$\frac{\partial T}{\partial z} = -\frac{\Gamma}{k_r}. \tag{12.27}$$

The geothermal flux Γ is typically taken to be one geophysical heat flow unit (0.5 W m^{-2}). The value of z_r is chosen to be 750 m below $z = b$. A useful consideration for choosing this distance $(b - z_r)$ is the e-folding penetration depth λ for temperature oscillations forced at the ice/bedrock interface of frequency ω (Carlsaw and Jaeger, 1954):

$$\lambda = \left(\frac{2\kappa_r}{\omega}\right)^{\frac{1}{2}} \tag{12.28}$$

where the thermal diffusivity of bedrock is defined by $\kappa_r = \frac{k_r}{\rho_r c_r}$.

Unless there are inflow boundaries around the horizontal edges of the ice sheet, thermal boundary conditions are not required around the edges of the ice sheet. This is because the horizontal mode of heat transfer in an ice sheet

is advection, a process that demands boundary conditions only where ice flow into the domain.

A material constraint on the ice-sheet thermodynamics is that ice never warm above the pressure-melting point. When layers of finite thickness reach the pressure-melting point, the ice sheet is said to become "polythermal" (Hutter, 1982; Hutter, Blatter and Funk, 1988). A review of polythermal ice-sheet modelling is provided by Greve (1995). Polythermal conditions typically occur as a result of viscous dissipation, which heats the ice column internally and permits the locus of a temperature maximum (the pressure-melting point, by definition) to move away from the boundary (a constraint that is otherwise forced by a consideration similar to the maximum principle of solutions of Laplace Equation). When ice becomes temperate (at the pressure melting point), its vertical temperature gradient becomes that dictated by the "Clapyron slope" of water (the change of the melting temperature with pressure). In this circumstance, the vertical heat flux is fixed (near zero), so further heat transfer (via internal heating and flux from non-temperate portions of the ice sheet) acts only to modify the liquid water content of the ice. In the ice-sheet model by Greve (1995), for example, the heat-flow continuity equation in the temperate ice is replaced with a diffusive-type equation for water content, and an internal free surface (called the CTS, cold/temperate ice transition surface) must be monitored through the use of energy flux and mass flux matching conditions.

In the examples described in this chapter, polythermal ice conditions are treated in a very simple manner. When ice reaches the local pressure melting point, it is assumed to be fixed at the pressure melting point. To accomplish this specification, we take each thermal time step twice (actually, just the vertical portion of the split thermal time step, not the horizontal advection portion). We first assume that there is no melting at any level of the ice, including the ice/rock interface. We take a vertical conduction/advection time step (with strain heating) and determine if T exceeds $T_m(z)$ for any level z within the ice. If the condition is true, *i.e.,* there is pressure melting at the base or within the ice, we retake the time step (starting from the same previous time step temperature) with all nodes where $T \geq T_m$ in the first time step specified to have $T = T_m$.

12.4.1 Drained Bed Conditions

When the base is at the pressure-melting point in Level 2 experiments, we compute a basal melting rate, $\dot{B}$ (positive for melting, meters of ice equivalent per year), to balance the heat budget at the basal ice interface. We define heat-flux terms H_o and H_i as the outward- and inward-directed heat fluxes to the interface $z = b$, respectively, by

$$H_o = -k\frac{\partial T}{\partial z}|_{z=b+} \tag{12.29}$$

and,

$$H_i = -k_r\frac{\partial T}{\partial z}|_{z=b-} \tag{12.30}$$

where $z = b+$ and $z = b-$ denote evaluation of the derivatives on one side of the basal boundary. With the above definitions for H_o and H_i, $\dot{B}$ is determined by

$$\dot{B} = \begin{cases} \frac{H_i - H_o}{\rho L_f} & \text{if } H_i > H_o \\ \\ 0 & \text{otherwise, assuming drained bed conditions} \end{cases} \tag{12.31}$$

where $L_f = 3.35 \times 10^5$ J kg^{-1} is the latent heat of fusion for pure water.

12.5 Climate Parameterization

Surface temperature and mass balance conditions are required at the upper surface of the ice sheet. These are provided by the EISMINT Greenland Level 1 specifications as follows.

Let,

$$Z(t) = \max \begin{cases} s(t) - (-34.83(\delta^{18}\mathrm{O}_{ds}(t) + 1.93)) \\ \\ 20(\phi - 65^\circ) \end{cases} \tag{12.32}$$

where $\delta^{18}\mathrm{O}_{ds}(t)$ (units $^o/_{oo}$) is the SPECMAP value of the deep-sea oxygen-isotope value at time t used to account for elevation changes associated with

changing sea level, the constant -34.83 m $(^o/_{oo})^{-1}$ is the assumed conversion rate between oxygen-isotope value of the ocean and sea level, 1.93 $^o/_{oo}$ is the assumed value of $\delta^{18}O_{ds}(t)$ at present, $t = 0$, t is time, with zero at present, and ϕ is latitude in units of degrees. Annual average surface temperature T_s is given by:

$$T_s = T_{sl} - \Gamma_z Z(t) - \Gamma_\phi \phi + 1.5(\delta^{18}O_{ic}(t) + 35.27) + T_{CO_2}(t) \tag{12.33}$$

where $T_{sl} = 49.13$ C, $\Gamma_z = 7.992 \times 10^{-3}$ C m^{-1}, $\Gamma_\phi = 0.7576$ C degree^{-1}, $1.5(\delta^{18}O_{ic}(t) + 35.27)$ C is the climate effect provided by the GRIP ice-core oxygen isotope ratio $\delta^{18}O_{ic}(t)$ (units $^o/_{oo}$), and $T_{CO_2}(t)$ is the greenhouse effect for $t > 0$ given by:

$$T_{CO_2} = \begin{cases} \int_0^t 0.035 dt & \text{if } t < 80 \text{ a} \\ \int_0^{80} 0.035 dt + \int_{80}^t 0.0017 dt & \text{if } t \geq 80 \text{ a} \end{cases} \tag{12.34}$$

The values of $\delta^{18}O_{ds}(t)$ and $\delta^{18}O_{ic}(t)$ are assumed constant at their present values for times $t/ge0$, and are interpolated from SPECMAP and GRIP data provided by the EISMINT Greenland Level 2 experiment (see Johnsen *et al.*, 1995; Imbrie *et al.*, 1984).

The summer mean temperature is given by

$$T_{sum} = (T_{sl})_{sum} - (\Gamma_z)_{sum} Z(t) - (\Gamma_\phi)_{sum} \phi + 1.5(\delta^{18}O_{ic}(t) + 35.27) + T_{CO_2}(t) \tag{12.35}$$

where $(T_{sl})_{sum} = 30.78$ C is the summer mean sealevel temperature, $(\Gamma_z)_{sum} = 6.277 \times 10^{-3}$ C m^{-1} is the summer value of Γ_z, and $(\Gamma_\phi)_{sum} = 0.3262$ C degree^{-1} is the summer value of Γ_ϕ.

The positive-degree-day factor, H_{pdd}, is given by (see Huybrechts, 1991):

$$H_{pdd} = \frac{1}{\sigma\sqrt{2\pi}} \int_0^{t_a} \left(-0.04394\,\sigma^2 + 1.23775 T_d \sigma + \sigma^2 \exp\left(\frac{{T_d}^2}{2\sigma^2}\right) + T_d \sigma \sqrt{\frac{\pi}{2}} \mathrm{erf}\left(\frac{T_d}{\sigma\sqrt{2}}\right) \right) dt \tag{12.36}$$

where t_a is one year, $\sigma = 5$ C is the standard deviation of the daily temperature, and T_d is the daily temperature (idealization of an annual cycle) given by:

$$T_d = T_s + (T_{sum} - T_s) \cos\left(\frac{2\pi t}{t_a}\right) \tag{12.37}$$

The units of H_{pdd} are (C day). All temperatures in the above two equations are in units of degrees Centigrade.

The glacial ice accumulation rate $\dot{A}$ is given by

$$\dot{A} = \begin{cases} \dot{P} - \alpha_s H_{pdd} \frac{\rho_w}{\rho} & \text{if } h_s = 25 \text{ m} \\ -\alpha_x H_{pdd} \frac{\rho_w}{\rho} & \text{if } h_s = h_x = 0 \\ 0 & \text{if } 0 < h_s < 25 \text{ and/or } 0 < h_x < 25 \end{cases} \tag{12.38}$$

The idea behind the above representation of the glacial ice accumulation rate (which drives ice-sheet growth and decay) is that ice accumulates only after a full thickness firn layer has developed ($h_s = 25$ m). Glacial ice will only ablate when all snow and superimposed ice have been eliminated by ablation. During the periods where the snow layer thickness is less than 25 m, glacial ice does not accumulate, because precipitation minus ablation is busy causing the snow and superimposed ice layers to change their thickness.

12.6 Bedrock Isostacy

We treat bedrock elevation using the scheme developed by Oerlemans and van der Veen as described by Payne (1995):

$$\frac{\partial b}{\partial t} = D_a \frac{\partial^2}{\partial x^2} \left(b - b_o + \frac{\rho h}{\rho_m} \right) \tag{12.39}$$

where $D_a = 10^8$ m^2 a^{-1} is the asthenospheric diffusivity, b_o is the equilibrium bedrock elevation in the absence of an ice load, and $\rho_m = 3300$ kg m^{-3} is the asthenospheric density. For b_o, we assume that the present bed elevation of greenland is in isostatic equilibrium with the present ice load, *i.e.*, $b_o = b + \frac{\rho h}{\rho_m}$. We do not account for the effects of changing sea level and consequent isostatic compensation of the bed in water-covered portions of the domain.

For boundary conditions, we require that $\frac{\partial b}{\partial x} = 0$ at the extreme endpoints of our domain. The unphysical aspect of this boundary condition is not

a concern because bed elevations at the extreme endpoints of the flowline domain are below sea level and do not influence the ice-sheet simulation elsewhere.

12.7 Calving, Sea-Level Effects

For simplicity, we do not consider floating ice. In fact, our ice sheet is able to advance to the bottom of the Labrador Sea as a grounded ice sheet if climatic conditions permit (*i.e.,* the Labrador Sea is assumed to be empty of seawater and, instead, filled with atmosphere). Calving is thus not predicted by the model. To avoid ice-sheet advance into the bottom of the Labrador Sea (a rather humorous circumstance which came up in our initial tests), we force the ice sheet to have zero thickness at nodes where the bed elevation is currently below -250 m.

Our experience with flowline ice-shelf models (see previous chapters) suggests that an ice-thickness constraint at marine termini is an adequate approach to simulating the effects of ice flotation in circumstances where there are no transverse constraints on ice-shelf flow.

12.8 Sliding

For the Greenland experiments, basal sliding is taken to be zero even when the bed is melted. We anticipate using the model in a mode where sliding is permitted, using the parameterization described by Payne (1995).

12.9 Contour-Following Vertical Coordinate

To handle the variable vertical dimension of the ice sheet, a new vertical coordinate ζ is defined such that $\zeta = 1$ at $s(x, y, t)$ and $\zeta = 0$ at $b(x, y, t)$.

The relation between ζ and z is

$$\zeta = \frac{z-b}{s-b} \tag{12.40}$$

and

$$z = (s-b)\zeta + b \tag{12.41}$$

Using the above definitions, z-derivatives are converted to ζ-derivatives, and the governing equation for thermodynamics becomes:

$$\begin{aligned} & T_t + uT_x \\ & + \frac{1}{h} T_\zeta \left(\mathcal{D}(\zeta) - \zeta \dot{A} - (1-\zeta)\dot{B} \right) \\ & = \frac{1}{\rho c h^2} \frac{\partial}{\partial \zeta} \left(k \frac{\partial T}{\partial \zeta} \right) + \frac{W}{\rho c} \end{aligned} \tag{12.42}$$

where $\mathcal{D}(\zeta)$ is an ice-divergence parameter defined by,

$$\mathcal{D}(\zeta) = h \left(\zeta \int_0^1 \frac{\partial u}{\partial x} d\zeta - \int_0^\zeta \frac{\partial u}{\partial x} d\zeta' \right) \tag{12.43}$$

We note the fact that in previous EISMINT Level 1 tests, the computation of the $\mathcal{D}$-term did not require analysis of horizontal gradients in $I(z)$. This was because a constant flow-law parameter $A_o = 10^{-16}$ Pa^{-3}/yr was applied. Here, it is necessary to be more general about the computation of horizontal divergence fields, caution is advised because it is not now possible to express the $\mathcal{D}$-term in terms of second derivatives of s. (We learned this the hard way.)

12.10 Discretization

According to the diagnostic relations between u, w and s for a grounded ice-sheet with "inland ice" type flow (Eqns. 12.6 and 12.7), the computation of the vertical velocity w and the $\mathcal{D}$-term as a function of ζ requires that

the second spatial derivative of s, *i.e.*, $\frac{\partial^2 s}{\partial x^2}$, be evaluated on the computational domain. This requirement necessitates the use of high-order element interpolation using Hermite-polynomial basis functions.

Variables s, b, $h = s - b$ are represented on each element's horizontal span, $x_l \leq x \leq x_r$, by the sum

$$s(x) = \sum_{j=1}^{4} s_j H_j(x) \tag{12.44}$$

where the $H_j(x)$, $j = 1, \ldots, 4$, are interpolation functions related to Hermite polynomials:

$$H_1(x) = \frac{1}{4}\left(2 - 3\xi + \xi^3\right) \tag{12.45}$$

$$H_2(x) = \frac{L_e}{8}\left(1 - \xi - \xi^2 + \xi^3\right) \tag{12.46}$$

$$H_3(x) = \frac{1}{4}\left(2 + 3\xi - \xi^3\right) \tag{12.47}$$

$$H_4(x) = \frac{L_e}{8}\left(-1 - \xi + \xi^2 + \xi^3\right) \tag{12.48}$$

$$\tag{12.49}$$

$L_e = x_r - x_l$ is the element's length, and $\xi = \frac{2(x - x_l)}{L_e} - 1$ is a scaled horizontal coordinate that is -1 at $x = x_l$ and $+1$ at $x = x_r$. The interpolation functions convey information about the interpolated function's value and first-derivatives at the two endpoints (nodes) of the line-segment element, and are displayed on the interval $-1 < \xi < 1$ in Figure (12.1). The coefficients s_j, for $j = 1, 3$, represent nodal values of $s(x)$ and s_j, for $j = 2, 4$, coefficients s_j provide information about the nodal values of $\frac{\partial s}{\partial x}$ (they are not the derivatives, however). Observe that, in the above notation, the subscript j refers to a local node-numbering scheme and that coefficients s_j vary from element to element according to the global element connectivity scheme.

The variation of T with x and z is represented by bi-directional linear

Figure 12.1: To permit evaluation of $\frac{\partial^2 s}{\partial x^2}$, interpolation functions based on the Hermite polynomials are employed. Four functions are used on each element. Two functions convey the value of the interpolated field at each endpoint (node) of the line segment (element), and two functions convey the value of the first-derivative of the interpolated field at each endpoint. The two interpolation functions that are zero at endpoints (nodes) are scaled by L_e, the length of the element, in this particular diagram.

interpolation (*i.e.*, linear in each separate direction):

$$T(x,y) = \sum_{j=1}^{2} \left(\sum_{k=1}^{2} T_{jk} L_j(x) \right) L_j(z) \tag{12.50}$$

where T_{jk} is the nodal value of $T(x,z)$ at the j'th node (line-segment endpoint) in the horizontal element and at the k'th node (line-segment endpoint) in the vertical element. Because a split timestep is used (see below), updates to T_{jk} are made first holding j fixed and next holding k fixed. This scheme avoids the necessity of integrating interpolation functions $L_j(x)L_k(z)$ over the 3-dimensional "brick" volume determined by the 8 corners where nodal values of T are resolved, *i.e.,* at the 8 points (x_j, z_k), $j = 1, 2$ and $k = 1, 2$. The linear interpolation functions $L_j(x)$ and $L_k(z)$ are of the form

$$L_j(x) = \alpha_j x + \beta_j \tag{12.51}$$

$$L_k(z) = \alpha_k z + \beta_k \tag{12.52}$$

where the coefficients α_m and β_m are determined by

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix}^{-1} \tag{12.53}$$

or

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{bmatrix} = \begin{bmatrix} z_1 & 1 \\ z_2 & 1 \end{bmatrix}^{-1} \tag{12.54}$$

and where (x_1, x_2) and (z_1, z_2) are the nodal coordinates (line segment endpoints) in the horizontal and vertical discretizations, respectively.

Integrals of products of interpolation functions, *e.g.,* $\int_{x_1}^{x_2} H_j(x) H_l(x) dx$ are evaluated using either exact integration formulae or Gaussian quadrature (see Chapter 9). When Gaussian quadrature is used, the integrals are approximated by sums as follows:

$$\int_{x_1}^{x_2} H_j(x) H_l(x) dx \approx \frac{L_e}{2} \sum_{g=1}^{N_g} \mathcal{W}_g H_j(\xi_g) H_l(\xi_g) \tag{12.55}$$

The $N_g = 5$ quadrature points are, in the example presented in this chapter, $\xi_g \in \{-.9061798459 \ -.5384693101 \ 0.0 \ .5384693101 \ .9061798459\}$, and the

weighting factors are $\mathcal{W}_g \in \{.2369268851\ .4786286705\ .5688888888\ .4786286705\ .2369268851\}$. Exact integration formulae are listed in Hulbe (personal communication). Further analysis demonstrates to us that a 3-point quadrature method yields equal results.

Time stepping of the heat- and mass-balance equations is accomplished asynchronously using a finite-difference representation of the time derivatives, *e.g.*,

$$T_t = \frac{T^{n+1} - T^n}{\Delta t_{therm}} \tag{12.56}$$

and,

$$h_t = \frac{h^{n+1} - h^n}{\Delta t_{mass}} \tag{12.57}$$

For the EISMINT intercomparison experiments, $\Delta t_{therm} = 50$ a, and $\Delta t_{mass} = 5$ a. Climate, ice velocity and other information are updated every mass balance time step.

Vertical and horizontal derivatives are involved in the heat-balance equation, thus we employ a split timestep to simplify integration of Eqn. (12.43):

$$\frac{\tilde{T}^{n+1}}{\Delta t} + u^n(\zeta_l)\tilde{T}_x^{n+1} = \frac{T^n}{\Delta t} \quad \forall l \tag{12.58}$$

$$\frac{T^{n+1}}{\Delta t} - \frac{1}{\rho c h^2}\left(kT_\zeta^{n+1}\right)_\zeta + \frac{\omega(\zeta)}{h}T_\zeta^{n+1} = \frac{\tilde{T}^{n+1}}{\Delta t} + \frac{W(\zeta)}{\rho c} \quad \forall k \tag{12.59}$$

where in the first of the above equations, l is the vertical level number, ζ_l is the l'th vertical level in the ice, and $\omega(\zeta)$ is the vertical velocity in the "ζ-system" given by:

$$\omega = \mathcal{D}(\zeta) - \zeta\dot{A} - (1-\zeta)\dot{B} \tag{12.60}$$

We note the fact that there is *no* artificial horizontal diffusion term or "upwinding" necessary to suppress numerical noise (wiggles) generated in high Peclet number flows (see Chapter 9). This is because we do not need to specify boundary conditions in Equation (12.58) when flow characteristics are directed out of all boundaries.

For regions of the domain that are free of glacial ice, the temperature for nodes that are reserved for ice in the vertical discretization of ice/rock temperature profiles are set to T_s. When ice first appears at a previously ice-free node, the initial temperature profile of the ice is taken to be uniform and equal to the minimum of T_s or -5 C.

12.11 EISMINT Greenland Intercomparison Benchmarks

To apply our flowline model to the Greenland model intercomparison experiment described by Catherine Ritz, we selected the GRIP transect (row 77 of the 2-dimensional input data) as the model domain. This transect corresponds only approximately to a flowline. We thus expect our results to differ substantially from the results of full, 3-dimensional models and from the observations of the actual ice sheet itself. Output data supplied to Grenoble is in the natural units for flowline models (volumes and areas are calculated assuming a 1 m width of the ice sheet transverse to the x-axis).

12.11.1 Level 2 Intercomparison Benchmarks

Select figures from the level 2 climate cycle experiment are provided below for information.

12.11.2 Level 3 Intercomparison Benchmarks

The only difference between the level 2 and level 3 Greenland experiments is that, in level 2, we assume an undrained bed (with unlimited water supply for basal freezing); whereas in level 3, we assume a drained bed (the bed freezes as soon as heat flux upward into the ice exceeds the the heat flux upward from the rock).

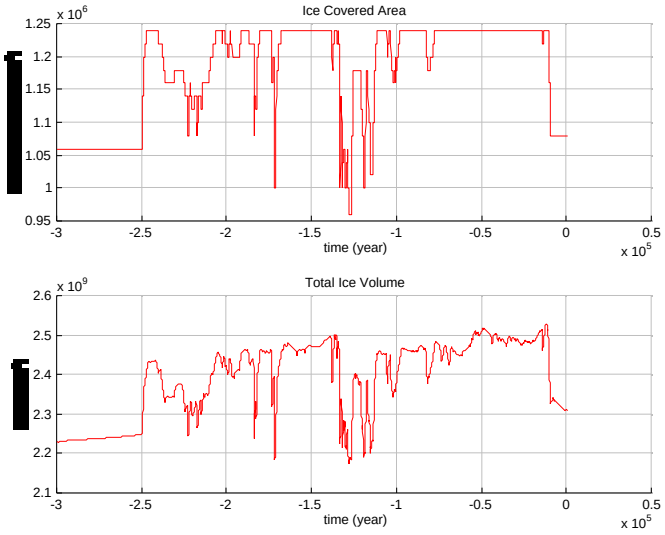

Figure 12.2: Ice covered area vs. time (upper panel) and ice volume vs. time (lower panel) for the Level 2 test (250,500-year climate run). The model is started from and "initial steady-state" condition at t=-300 ka to allow further adjustment of model fields toward steady state. Units of area and volume assume a 1-m width.

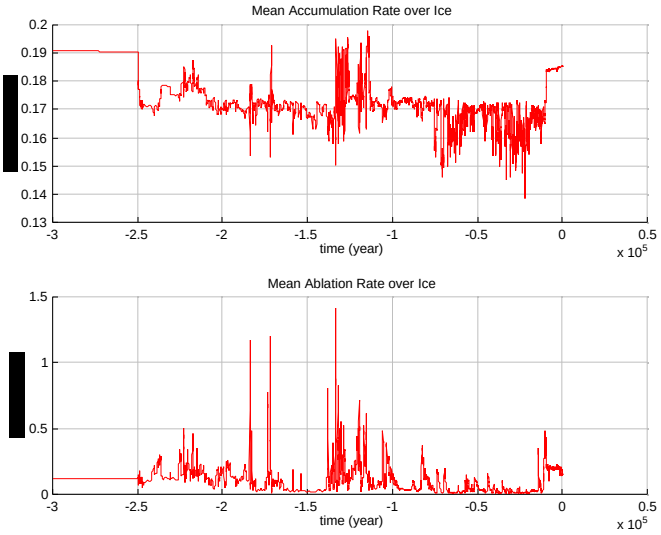

Figure 12.3: Mean accumulation rate (m a^{-1}) vs. time (upper panel) and mean ablation rate (m a^{-1}) vs. time (lower panel) for the Level 2 test (250,500-year climate run). Values are averaged for glacial-ice-covered areas only.

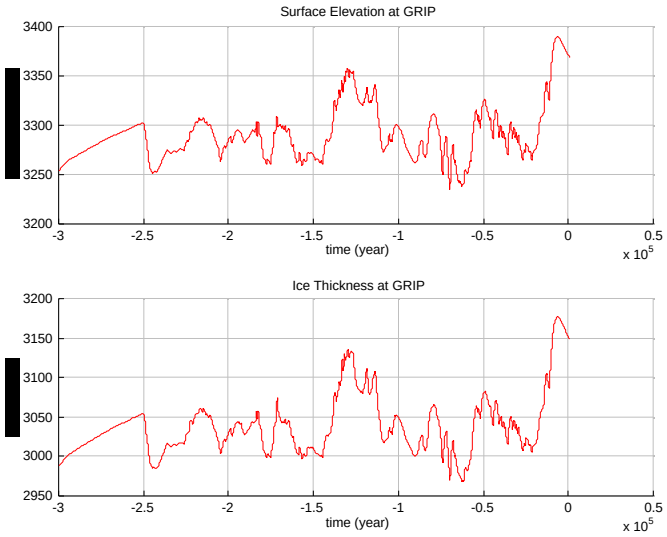

Figure 12.4: Surface elevation vs. time (upper panel) and thickness vs. time (lower panel) at the GRIP ice-core site for the Level 2 test (250,500-year climate run).

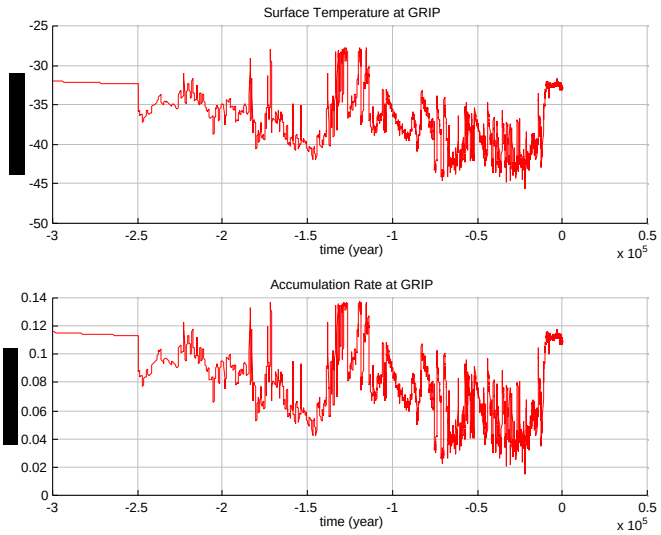

Figure 12.5: Surface Temperature vs. time (upper panel) and accumulation rate (m a^{-1}) vs. time (lower panel) for the Level 2 test (250,500-year climate run) at the GRIP ice-core site.

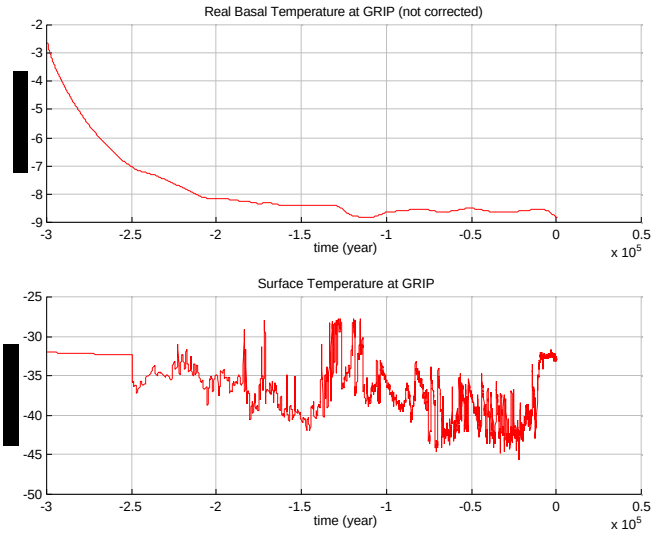

Figure 12.6: Basal temperature vs. time (upper panel) and surface temperature vs. time (lower panel) for the Level 2 test (250,500-year climate run) at the GRIP ice-core site.

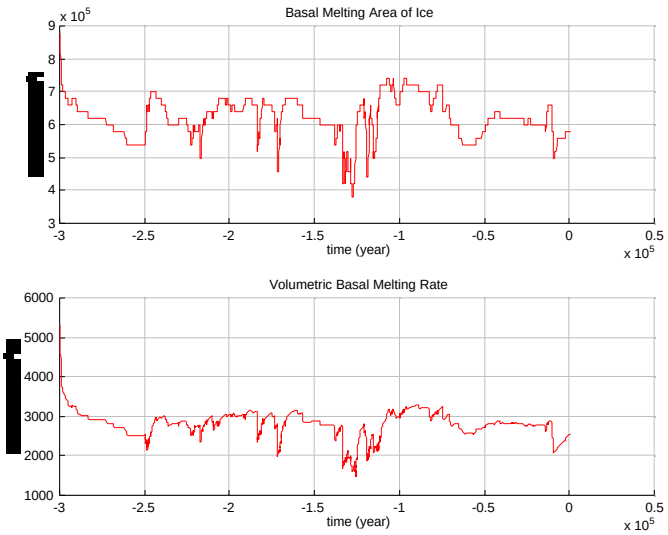

Figure 12.7: Basal temperature vs. time (upper panel) and area of basal melting vs. time (lower panel) for the Level 2 test (250,500-year climate run).

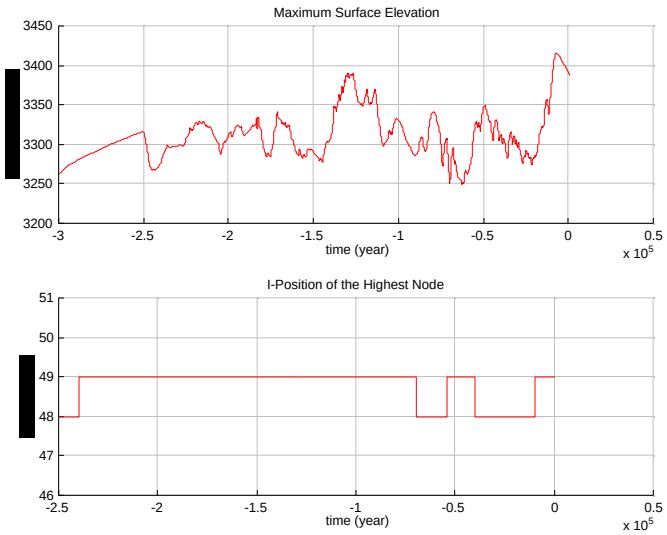

Figure 12.8: Maximum surface elevation vs. time (upper panel) and node of maximum surface elevation vs. time (lower panel) for the Level 2 test (250,500-year climate run).

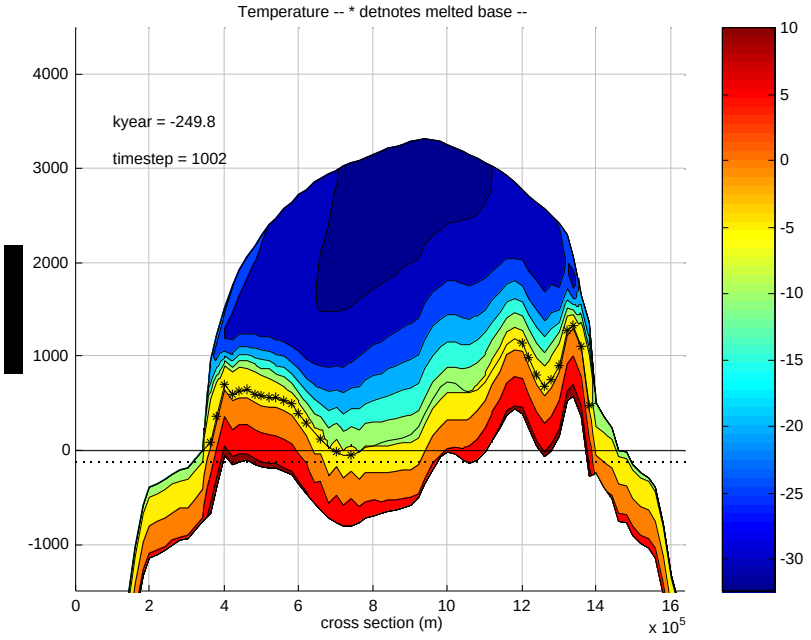

Figure 12.9: Sample model results. Ice-sheet and bed elevation at the beginning of the 250,000-year time integration of the climate cycle (*i.e.,* at present time, $t = 0$). This represents nearly steady-state conditions associated with present climate conditions. Basal melting is denoted by *-symbols. Also shown are the observed surface and bed elevations. Our ice sheet is significantly larger than that observed due to the fact that transverse divergence of the flowline is not allowed. Wiggles in the temperature field are due to the fact that we do not use "upwind" differencing type artificial diffusion.

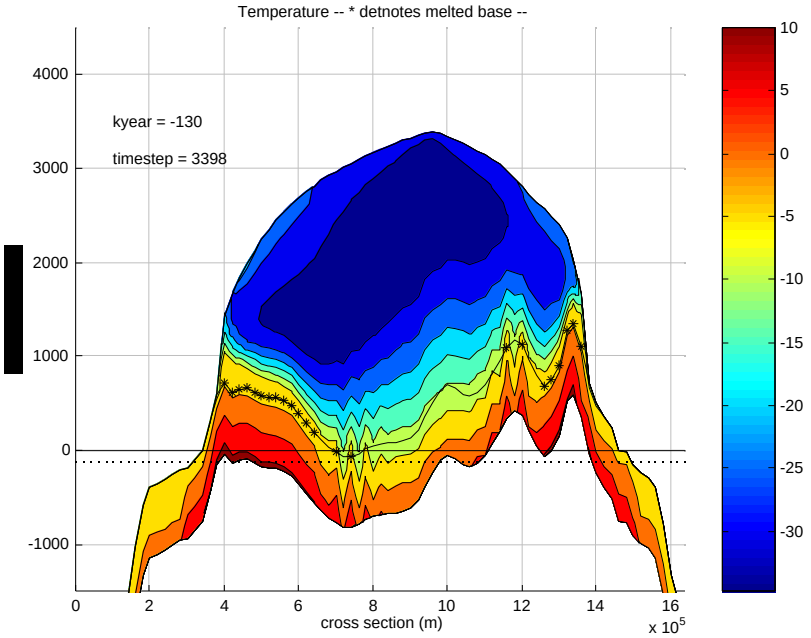

Figure 12.10: Sample model results. Ice-sheet and bed elevation at $t = -130$ ka. Basal melting is denoted by *-symbols.

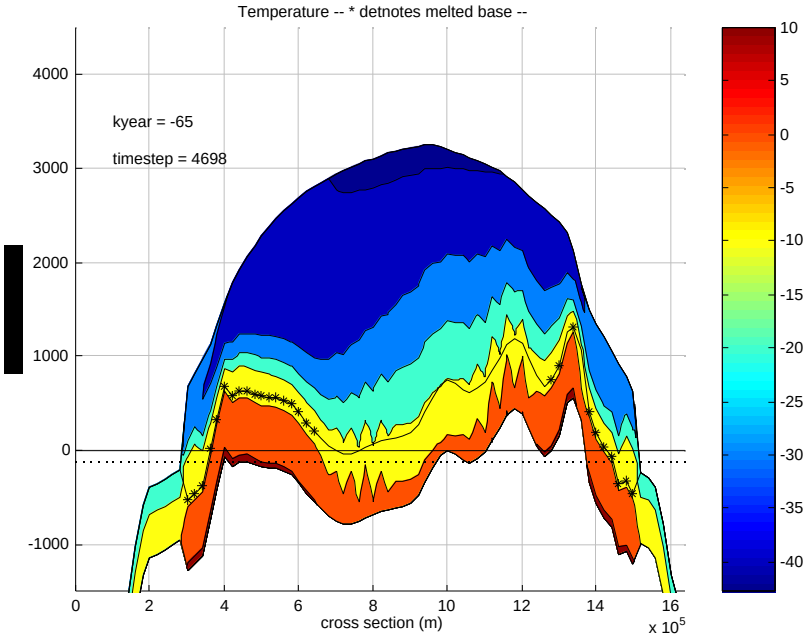

Figure 12.11: Sample model results. Ice-sheet and bed elevation at $t = -130$ ka. Basal melting is denoted by *-symbols.

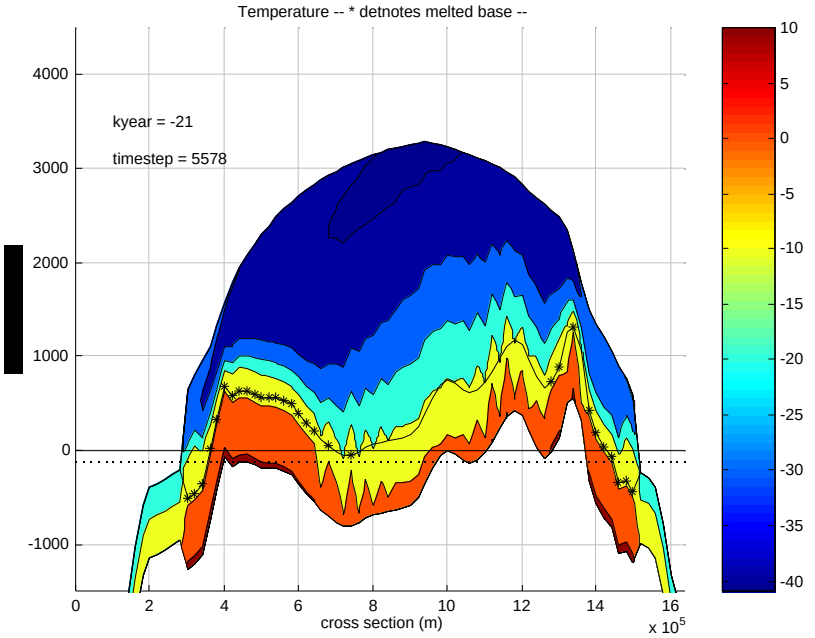

Figure 12.12: Sample model results. Ice-sheet and bed elevation at $t = -21$ ka. Basal melting is denoted by *-symbols.

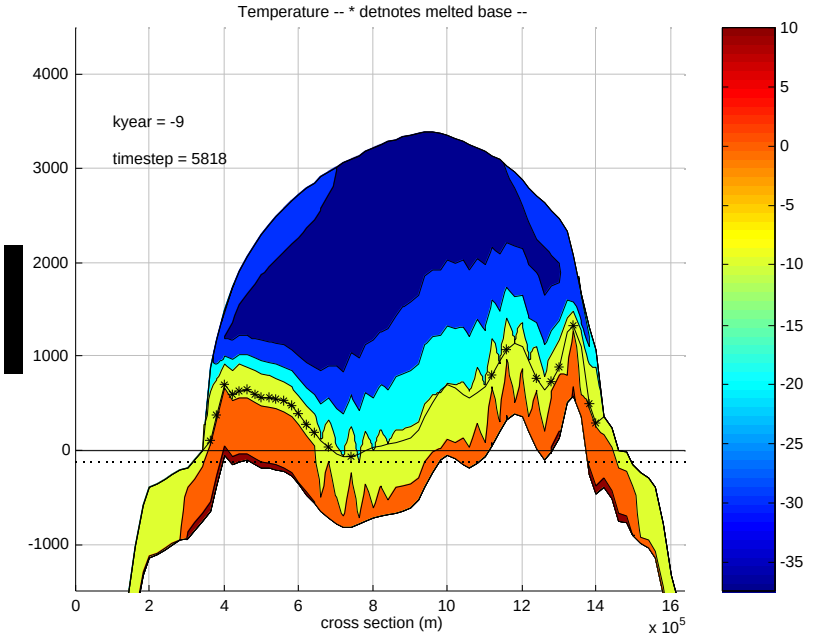

Figure 12.13: Sample model results. Ice-sheet and bed elevation at $t = -9$ ka. Basal melting is denoted by *-symbols.

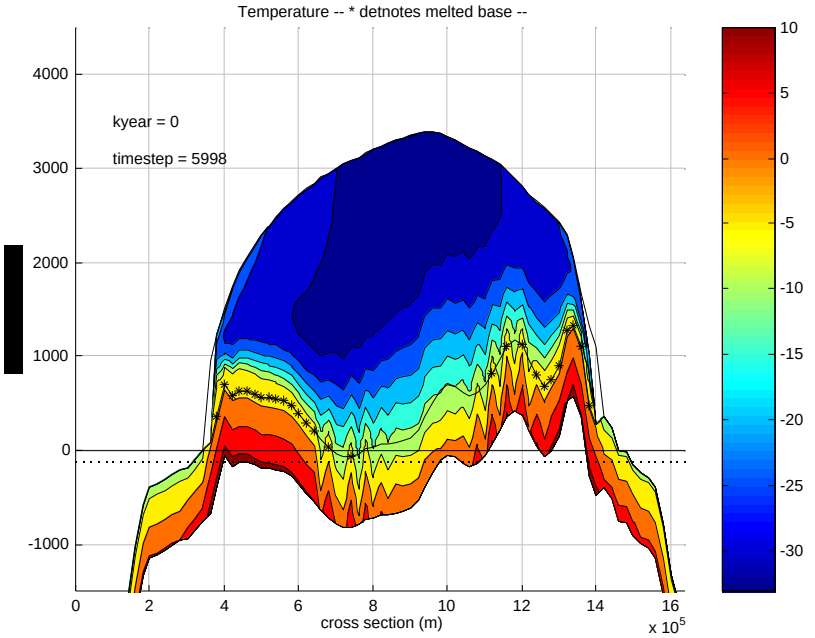

Figure 12.14: Sample model results. Ice-sheet and bed elevation at $t = 0$ ka. Basal melting is denoted by *-symbols.