NUMERICAL APPROXIMATION OF A TWO–DIMENSIONAL THERMOMECHANICAL MODEL FOR ICE FLOW

ED BUELER

ABSTRACT. In section one I describe the model problem. It is a close copy of [20, 22, 14, 21, 23] in the two–dimensional case.

Several finite difference schemes for the decoupled continuity equation are compared in the second section. An incomplete "experimental" comparison is given. *Matlab* codes for several methods are given (`iceA.m`). [Latrice Bowman [1] will be continuing with more sophisticated comparisons of more sophisticated methods. Also, careful consideration of the local truncation error of some of these methods is needed.] Based on the tentative results of section 2, and on my understanding of the literature, a semi–implicit "type II" method ("method 4") is fixed upon for treating the continuity equation in sections 3 and 4 of these notes. However, I also propose a semi–implicit "extrapolated diffusivity" type II method of higher order (method 7) (`iceB.m`).

A finite difference "free boundary" scheme for the temperature equation is described in section three. It avoids the Jenssen [15] change of variables, and in this sense the free–boundary nature of the problem for temperature is respected and not traded for a singular temperature equation. I do a first experiment in which heat flows only in an isolated column (i.e. there is no horizontal advection). See `iceC.m`.

In section four a scheme for the coupled equations is described. That is, the continuity and temperature equations are solved simultaneously and are coupled through the Arrhenius relation. Advection in both the horizontal and vertical directions is included, as is a dissipation heat source and the appropriate basal and surface temperature boundary conditions. A semi–implicit method computes only the heat conduction term implicitly, giving stability. In fact, no stability problems are observed which do not already arise in the uncoupled continuity equation. The vertical grid is allowed to have any given (unequal) spacing. Second–order upwinding in both vertical and horizontal is used for the advection terms. A *Matlab* code (`iceD.m`) and the outputs of several numerical experiments are given.

Date: August 23, 2002. *Address*: Dept. Math. Sciences UAF, Fairbanks AK 99775. *Email*: `ffelb@uaf.edu`.

Contents

1. The mathematical model

I make the following physical assumptions, corresponding roughly to the two–dimensional *EISMINT* [14, 21, 23] experiment but with thermomechanical coupling. The assumptions are also informed by [20, 22] and [10, 19, 27] generally. Though the equations are quite simplified, I claim (or at least, *hope*) that the resulting mathematical model still contains the representative structure of any reasonable cold and shallow thermomechanical model for ice [11, 6].

(i) Flow is only in the horizontal (x) and vertical (z) directions;
(ii) flow occurs only by horizontal shear stress and a nonlinear Glen flow law [5] relates the stresses and strain rates;
(iii) temperature change is by conduction in the vertical direction, by advection in vertical and horizontal, and by dissipation of flow as heat energy;
(iv) ice is incompressible;
(v) a Arrhenius relation determines the coefficient in the Glen flow law [27] as a function of temperature;
(vi) the thickness is held zero at the two ends of an interval (i.e. fixed margin);
(vii) the bed is flat, nondeformable and at zero elevation; the contribution of ice melt to vertical velocity at the base is ignored and thus vertical velocity at the base is assumed zero;
(viii) the snow accumulation rate is constant;
(ix) no sliding occurs at the base;
(x) the temperature at the surface increases as the cube of the distance from the center;
(xi) and the temperature boundary condition at the base is the geophysical heat flux.

The unknown functions are (notation roughly follows [27])

$h(x,t)$	surface elevation (m) [= thickness by (vii)],
$T(x,z,t)$	temperature (K),
$u(x,z,t)$	horizontal component of velocity (m/s),
$w(x,z,t)$	vertical component of velocity (m/s).

Here $x \in [-L, L]$, $z > 0$, $t \geq 0$. The following equations correspond *roughly* to items (i) through (xi) above:

$$\frac{\partial h}{\partial t} = B - \frac{\partial}{\partial x}\left(\int_0^h u\,dz\right) \tag{1}$$

$$u = -2(\rho g)^n \left|\frac{\partial h}{\partial x}\right|^{n-1} \frac{\partial h}{\partial x} \int_0^z A(h-\zeta)^n\,d\zeta + u_b \tag{2}$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + w\frac{\partial T}{\partial z} = \frac{k}{\rho C_p}\frac{\partial^2 T}{\partial z^2} + \frac{g(h-z)}{C_p}\left|\frac{\partial h}{\partial x}\frac{\partial u}{\partial z}\right| \tag{3}$$

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0 \tag{4}$$

$$A = A_0 \exp\left(\frac{-Q}{RT} + \frac{3c}{(T_r - T)^\kappa}\right) \qquad \text{(from Hooke [9])} \tag{5}$$

$$h(-L,t) = h(L,t) = 0, \quad L = 750\text{km} \tag{6}$$

$$w_b = w(x,0,t) = 0 \tag{7}$$

$$B = 0.3\frac{\text{m}}{\text{a}} = 9.51\times 10^{-9}\frac{\text{m}}{\text{s}}\text{(ice–equivalent rate)} \tag{8}$$

$$u_b = u(x,0,t) = 0 \tag{9}$$

$$T(x,h,t) = 239 + (8\times 10^{-8})|x| \tag{10}$$

$$\frac{\partial T}{\partial z}(x,0,t) = -\frac{G}{k} \text{ if } T(x,0,t) \le \tilde{T}(x,0,t), \text{ and otherwise} \tag{11}$$
$$T(x,0,t) = \tilde{T}(x,0,t) \text{ where } \tilde{T}(x,z,t) = T_r - \beta(h-z)$$

The function $\tilde{T}(x,z,t)$ is the melting temperature of ice at depth. The constants appearing above have the following values ([14] and [27], pp. 16, 180):

$\rho = 910\,\frac{\text{kg}}{\text{m}^3}$ (density of ice)	$R = 8.321\,\frac{\text{J}}{\text{mol K}}$ (gas constant)
$g = 9.81\,\frac{\text{m}}{\text{s}^2}$ (acceleration of gravity)	$\kappa = 1.17$
$k = 2.10\,\frac{\text{J}}{\text{m K s}}$ (thermal conductivity of ice)	$c = 0.16612\,\text{K}^\kappa$
$C_p = 2009\,\frac{\text{J}}{\text{kg K}}$ (specific heat capacity of ice)	$T_r = 273.39\,\text{K}$
$A_0 = 2.948\times 10^{-9}\,\frac{1}{\text{Pa}^3\text{s}}$	$G = .042\frac{\text{J}}{\text{m}^2\text{s}}$ (geothermal heat flux)
$Q = 7.88\times 10^4\,\frac{\text{J}}{\text{mol}}$ (activation energy for creep)	$\beta = 8.7\times 10^{-4}\,\frac{\text{K}}{\text{m}}$ (change of melting point with depth)

Note B is the ice (snow) accumulation rate and L is the half–width of the sheet.

When comparing to *EISMINT* results, and in sections 2 and 3 for concreteness, I choose $n = 3$.

Figure 1 shows the boundary value problem for the ice thickness and temperature.

2. Choices: numerical methods for the continuity equation

For this and the next section I consider the decoupled system which follows from the replacement of (5) by the assumption:

Temporary Assumption 1 (Sections 2, 3 only)**.** The flow parameter A is assumed constant. It is given the value 10^{-16} $\text{Pa}^{-3}\text{a}^{-1}$ which is the *EISMINT* [14] value.

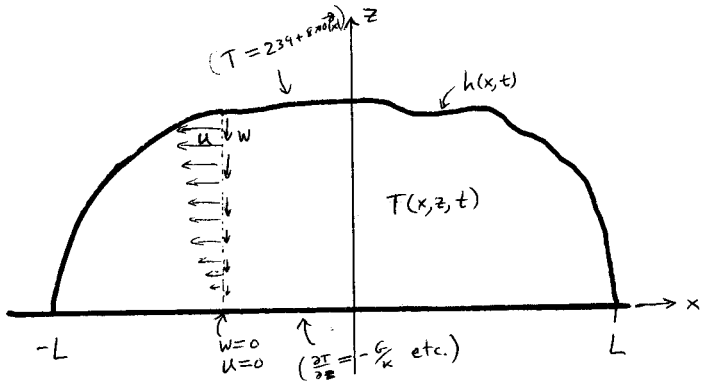

FIGURE 1. Region and boundary conditions for h, T, u and v.

I also assume for this section that $n = 3$.

These assumptions simplify the comparison of numerical methods for the continuity equation (1).

Let $\alpha(x,t) = \left|\frac{\partial h}{\partial x}(x,t)\right|$ be the surface slope. By assumption 1, both u and w can be found from the geometry (i.e. h and $\frac{\partial h}{\partial x}$) alone. In particular,

$$u = -2(\rho g)^3 A\alpha^2 \frac{\partial h}{\partial x}\int_0^z (h-\zeta)^3 d\zeta = \frac{1}{2}(\rho g)^3 A\alpha^2 \frac{\partial h}{\partial x}((h-z)^4 - h^4) \tag{12}$$

from (2). For the purposes of section 3, I also note:

$$\begin{aligned} w &= -\int_0^z \frac{\partial u}{\partial x}(x,\zeta,t)d\zeta \\ &= -2(\rho g)^3 A\alpha^2\left[\frac{3}{4}\frac{\partial^2 h}{\partial x^2}\left(\frac{h^5}{5} - \frac{(h-z)^5}{5} - h^4 z\right) + \alpha^2\left(\frac{h^4}{4} - \frac{(h-z)^4}{4} - h^3 z\right)\right] \end{aligned} \tag{13}$$

from incompressibility (4) and the boundary condition $w(x,0,t) = 0$ in (7). Note that with the decoupling assumption 1, the value of w is immaterial for modelling h though it is needed for modelling T.

From (12),

$$\int_0^h u\,dz = \frac{1}{2}(\rho g)^3 A\alpha^2 \frac{\partial h}{\partial x}\left(\int_0^h (h-z)^4 - h^4\,dz\right) = -\frac{2}{5}(\rho g)^3 A\alpha^2 \frac{\partial h}{\partial x} h^5,$$

which is a familiar expression for $\bar{u}h$ where $\bar{u}$ is the vertically–averaged velocity. Thus (1) becomes a "nonlinear diffusion equation" form of the continuity equation

$$\boxed{\frac{\partial h}{\partial t} = B + \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x}\right)} \tag{14}$$

where

$$\boxed{D(x,t) = \frac{\Gamma}{5}\alpha^2 h^5 \quad \text{and} \quad \Gamma = 2(\rho g)^3 A.}$$

I regard D as a "diffusivity coefficient" though this is only a heuristic label because D depends on h. Note D is positive if and only if $h_x \neq 0$ and $h > 0$. Thus the boundary where the thickness goes to zero and ridges where the slope goes to zero should be regarded as singular points for equation (14), and presumably for the general case (1) as well.

The goal for this section[1] is to describe, analyze and experiment with some choices of finite difference methods for the initial boundary value problem consisting of (14), (6) and some initial condition. Convenient initial conditions for preliminary experiments include $h(x,0) = 0$ or $h(x,0) = h_{\text{exact}}(x)$ where h_{exact} is the analytical steady solution described in appendix A.

See [17, 18, 24, 25, 26] for general information about finite difference schemes. I think [24], chapter 19, is the most accessible introduction. I think [25] is the most complete reference.

Let $x_j = -L + (j-1)\Delta x$ be a grid for x ($j = 1, \ldots, N_x + 1$ where $N_x \Delta x = 2L$). Let $0 \le t \le t_{end}$ and $t_l = 0 + (l-1)\Delta t$ ($l = 1, \ldots, M_t + 1$ where $M_t \Delta t = t_{end}$).[2] Let $h_{j,l}$ be an approximation to $h(x_j, t_l)$ and $D_{j,l}$ be an approximation to $D(x_j, t_l)$.

Method 1 (Explicit, "type I")**.** Inspired by the suggestions of van der Veen [27], approximate (14) by

$$\frac{h_{j,l+1} - h_{j,l}}{\Delta t} = B + \frac{1}{\Delta x}\left(\bar{D}_{j+1/2,l}\left(\frac{h_{j+1,l} - h_{j,l}}{\Delta x}\right) - \bar{D}_{j-1/2,l}\left(\frac{h_{j,l} - h_{j-1,l}}{\Delta x}\right)\right) \tag{15}$$

where

$$\bar{D}_{j+1/2,l} = \frac{\Gamma}{5}\left|\frac{h_{j+1,l} - h_{j,l}}{\Delta x}\right|^2 \left(\frac{h_{j+1,l} + h_{j,l}}{2}\right)^5 \tag{16}$$

and similarly for $\bar{D}_{j-1/2,l}$. The "diffusivity" approximation (16) is the "type I" approximation referred to in *EISMINT* [14].

It is not obvious that equations (15), (16) produce a method with local truncation error $O(\Delta x^2, \Delta t)$ as $\Delta x, \Delta t \to 0$, but that is the case. Indeed, the precise local truncation error claim is that

$$\left[\frac{\partial h}{\partial t} - B - \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x}\right)\right]_{x_j,t_l} = [\text{approximation (15) and (16)}] + O(\Delta x^2, \Delta t)$$

where I suppose $h_{j,l+1} = h(x_j, t_{l+1})$, $h_{j,l} = h(x_j, t_l)$, etc., and where the bracketed expression on the right is fully expanded as a function of $\{h_{j+1,l}, h_{j-1,l}, h_{j,l}, h_{j,l+1}\}$.

[1] The project goal for Latrice [1] is to extend this analysis and comparison to the more sophisticated methods.

[2] The indexing starting with $j = 1$, $l = 1$ matches the *Matlab* convention.

I was only able to verify this by using *Mathematica* to manipulate the Taylor expansions. On the other hand, the combination of (15), (16) is symmetric in the index j (around the point x_j) so second–order–in–space performance is not surprising once one determines that the method is consistent (i.e. that the zeroth order term is correct).

Figure 2 shows the "stencil" of the method. Note the (at least notional) "staggered grid". Open circles are used to indicate the new (unknown) values.

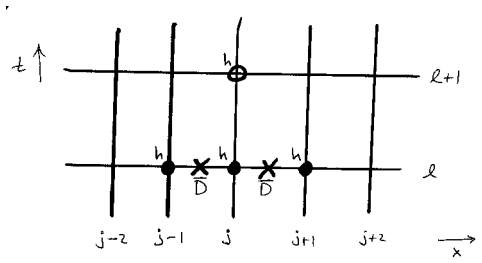

FIGURE 2. Method 1 for (14).

Method 2 (Explicit, "type II")**.** Further inspired by [27, 14], approximate (14) by

$$\frac{h_{j,l+1}-h_{j,l}}{\Delta t} = B + \frac{1}{\Delta x}\left(D_{j+1/2,l}\left(\frac{h_{j+1,l}-h_{j,l}}{\Delta x}\right) - D_{j-1/2,l}\left(\frac{h_{j,l}-h_{j-1,l}}{\Delta x}\right)\right) \tag{17}$$

but approximate $D_{j+1/2,l}$ by

$$D_{j+1/2,l} = \frac{1}{2}\left(D^*_{j+1,l} + D^*_{j,l}\right) \tag{18}$$

and

$$D^*_{j,l} = \frac{\Gamma}{5}\left|\frac{h_{j+1,l}-h_{j-1,l}}{2\Delta x}\right|^2 h^5_{j,l}. \tag{19}$$

The "diffusivity" approximation (18), (19) is the "type II" approximation referred to in [14].

Methods 1 and 2 are easy to program and easy to generalize to three dimensions because they are explicit. Also, because they are explicit and one step, their accuracy is limited to first order in time (i.e. $O(\Delta t)$). One should consider the possibility of multistep explicit schemes, with the obvious caveat that such schemes are less stable for stiff problems. The continuity equation for ice flow, as a "nonlinear diffusion" presumably generates stiff systems when the spatial terms are discretized.

Figure 3 shows the stencil, which emphasizes that this explicit method actually depends on *five* values of h at the current time. In fact, this raises an extremely important issue, namely, that more boundary conditions seem necessary for type II. In particular, $D^*_{1,l}$ and $D^*_{N_x+1,l}$, which are located at the boundaries $x = -L$ and

$x = L$, need to be given values. In the analytical steady solution one sees that $D(x = \pm L) = 0$ (see appendix A), and this is why I have used $D^* = 0$ as a boundary condition. For type I schemes this is not an issue, as only a boundary condition on h is needed. A more sophisticated condition might be required at the boundary in the future, perhaps by calculating the boundary D by type I even if all others are calculated by type II.

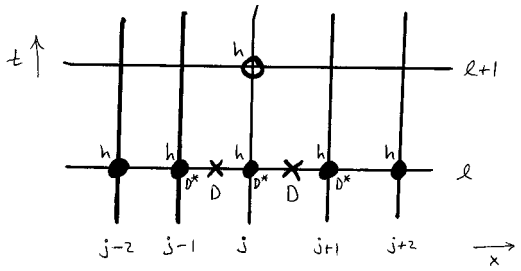

FIGURE 3. Method 2 for (14).

Method 3 (Semi–implicit, "type I")**.** For greater stability one may approximate (14) by

(20)
$$\frac{h_{j,l+1} - h_{j,l}}{\Delta t} = B + \frac{1}{\Delta x}\left(\bar{D}_{j+1/2,l}\left(\frac{h_{j+1,l+1} - h_{j,l+1}}{\Delta x}\right) - \bar{D}_{j-1/2,l}\left(\frac{h_{j,l+1} - h_{j-1,l+1}}{\Delta x}\right)\right)$$

where $\bar{D}_{j+1/2,l}, \bar{D}_{j-1/2,l}$ are calculated by (16). Note that the h terms on the right are at time t_{l+1} but that the "old" values of $\bar{D}$ are used. Thus the label *semi*–implicit.

Experience with the heat/diffusion equation motivates such implicit treatment. Any description of numerical methods for PDEs ([24] chapter 19 is easy to read) includes an analysis of the "fully implicit" method

$$\frac{u_{j,l+1} - u_{j,l}}{\Delta t} = D\frac{u_{j+1,l+1} - 2u_{j,l+1} + u_{j-1,l+1}}{\Delta x^2}$$

for the diffusion equation

$$u_t = (Du_x)_x$$

where $D > 0$ *is assumed constant.* One learns that the fully implicit method is unconditionally stable, that is, that time steps Δt of any size may be taken, and will affect accuracy but not cause explosion of spurious high–frequency modes. A related analysis concludes that the fully implicit method for the nonconstant–diffusion equation

$$u_t = (D(x,t)u_x)_x$$

is unconditionally stable (exercise 2.8 of [17]; see also appendix B). A *nonlinear* diffusion equation of the form

$$u_t = (D(u,u_x)u_x)_x$$

which is not directly handled by the previous linear theory.

In any case, figure 4 shows the stencil of method 3, which emphasizes that "h is computed implicitly but the diffusivity is computed explicitly".

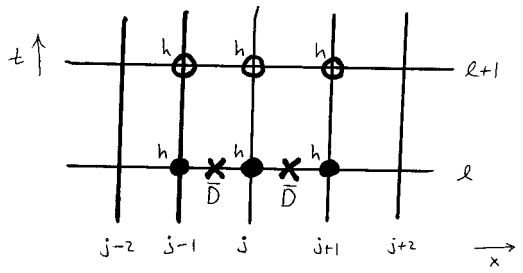

FIGURE 4. Method 3 for (14).

Method 4 (Semi–implicit, "type II")**.** One may also approximate (14) by

(21)
$$\frac{h_{j,l+1}-h_{j,l}}{\Delta t} = B + \frac{1}{\Delta x}\left(D_{j+1/2,l}\left(\frac{h_{j+1,l+1}-h_{j,l+1}}{\Delta x}\right) - D_{j-1/2,l}\left(\frac{h_{j,l+1}-h_{j-1,l+1}}{\Delta x}\right)\right)$$

where $D_{j+1/2,l}$, $D_{j-1/2,l}$ are computed by the "type II" method, equations (18), (19). Figure 5 shows the stencil.

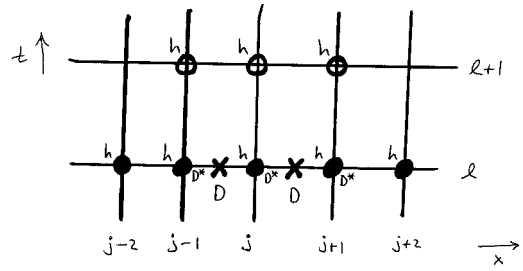

FIGURE 5. Method 4 for (14).

The following table compares the empirical stability of methods 1,2,3 and 4. These results are from `iceA.m`, shown in appendix M. In most cases the table gives the largest whole number value (in years) of the stepsize Δt for which the respective method gives empirical stability. If that stepsize is less than one year, the best tenth is given. Empirical stability is typically determined by noting diffusivity values in excess of 1.5 times the maximum steady state value at any time during a 25000 year run. In some cases, "instability" was determined by dramatically–in–error diffusivity profile. Grids of $N_x = 30, 100, 200$ values of x are compared. The runs use the analytical steady state solution (see appendix A) to the continuity equation as the

initial condition. Thus this experiment is for *stability around the steady state* only (see appendix B). The computation time in seconds on my machine (Pentium III at 800 Mhz; *Matlab* version 5.3) is given in brackets next to the critical value of Δt. These times should be taken as very general indications only.

	$N_x = 30$	$N_x = 100$	$N_x = 200$
Method 1	10 [0.77]	.8 [13.4]	(no run)
Method 2	30 [0.27]	2 [5.88]	.5 [37.3]
Method 3	34 [0.93]	2 [27.9]	.6 [160]
Method 4	320 [0.11]	26 [2.19]	5 [18.8]

In producing this table I noted that, and [14] also mentions that, the type I methods give a slight over–estimate and the type II methods give a somewhat greater under–estimate of the steady state profile $h(x)$. An ad hoc average of the type I and type II is possible. For instance, by taking (0.8 type I + 0.2 type II) one gets greater accuracy than either method, but the stability is essentially the same as the type I method. It is also possible to use type I differencing near the boundary and type II in the interior to get more accuracy and still keep the stability of the type II method. (These schemes are options in `iceA.m`.)

Methods 5 and 6 which follow have not yet been tested. They involve Newton iteration and this is a serious–but–not–insurmountable complication (see [1]).

Method 5 (Fully–implicit, type I, "Crank–Nicolson")**.** This is the method first used (I believe) for ice flow by Mahaffy [16]. It is $O(\Delta x^2, \Delta t^2)$ as $\Delta x, \Delta t \to 0$ according to [16]. One approximates (14) by

$$\begin{aligned}(22)\qquad \frac{h_{j,l+1} - h_{j,l}}{\Delta t} &= B + \\ &\frac{1}{2\Delta x}\left(\bar{D}_{j+1/2,l}\left(\frac{h_{j+1,l} - h_{j,l}}{\Delta x}\right) - \bar{D}_{j-1/2,l}\left(\frac{h_{j,l} - h_{j-1,l}}{\Delta x}\right)\right) + \\ &\frac{1}{2\Delta x}\left(\bar{D}_{j+1/2,l+1}\left(\frac{h_{j+1,l+1} - h_{j,l+1}}{\Delta x}\right) - \bar{D}_{j-1/2,l+1}\left(\frac{h_{j,l+1} - h_{j-1,l+1}}{\Delta x}\right)\right)\end{aligned}$$

where $\bar{D}_{j+1/2,l}$, etc., are computed by (16). The presumed advantage of this method is its higher accuracy and the fact that for the corresponding linear diffusion equation it is unconditionally stable.

Figure 6 shows the stencil. Exactly because D is computed at the new time t_{l+1}, nonlinear equations must be solved. This is typically done by Newton's method [24].

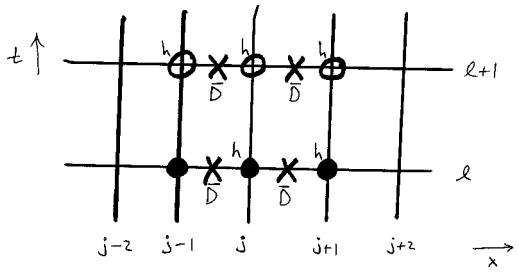

FIGURE 6. Method 5 for (14).

Method 6 (Fully–implicit, type II, "Crank–Nicolson")**.** Approximate (14) by

$$\frac{h_{j,l+1}-h_{j,l}}{\Delta t} = B+ \tag{23}$$
$$\frac{1}{2\Delta x}\left(D_{j+1/2,l}\left(\frac{h_{j+1,l}-h_{j,l}}{\Delta x}\right)-D_{j-1/2,l}\left(\frac{h_{j,l}-h_{j-1,l}}{\Delta x}\right)\right)+$$
$$\frac{1}{2\Delta x}\left(D_{j+1/2,l+1}\left(\frac{h_{j+1,l+1}-h_{j,l+1}}{\Delta x}\right)-D_{j-1/2,l+1}\left(\frac{h_{j,l+1}-h_{j-1,l+1}}{\Delta x}\right)\right)$$

where $D_{j+1/2,l}$, etc., are computed by (18), (19).

Figure 7 shows the stencil. One sees that significantly greater complication follows from the type II diffusivity approximation in this case. That is, a nonlinear "pentadiagonal" system must be solved in the two–dimensional case. In the three dimensional case using ADI techniques this would be even more complicated (though still conceivable). This method might give stability benefits over method 5. That would be its only justification.

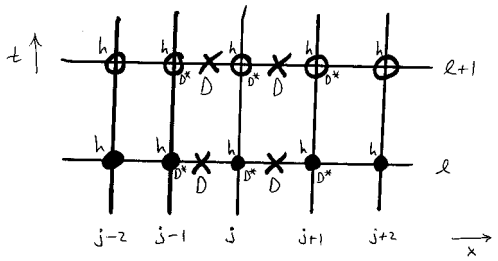

FIGURE 7. Method 6 for (14).

Method 7 (Semi–implicit, extrapolated diffusivity, "type II")**.** This method is best explained by starting with a related model equation. Suppose $D(t)$ is a given positive and differentiable function and suppose one wishes to solve

$$u_t = (D(t)u_x)_x. \tag{24}$$

For this linear equation the previous six methods reduce to three, that is, explicit, fully implicit and Crank–Nicolson. The last of these methods is the most accurate, of course. The fully implicit and Crank–Nicolson methods are easy to use because $D(t)$ can be evaluated at $t = t_{l+1}$ and $t = t_{l+1/2} = t_l + \frac{1}{2}\Delta t$ respectively. Recall that Crank–Nicolson has local truncation error $O(\Delta x^2, \Delta t^2)$ because it is symmetrical around the point $(x_k, t_{l+1/2})$.

Suppose, however, that one is only able to use the values of $D(t)$ at $t = t_l$ and possibly previous times—an artificial restriction in the current context, but more–or–less the case for the ice continuity equation. From Taylor series one sees that

$$D(t_{l+1/2}) = \frac{3}{2}D(t_l) - \frac{1}{2}D(t_{l-1}) + O(\Delta t^2).$$

That is, one can extrapolate to find $D(t_{l+1/2})$ using the previous two grid values $D(t_l), D(t_{l-1})$. This extrapolated value can be used in Crank–Nicolson in place of the exact value $D(t_{l+1/2})$.

In the nonlinear ice flow case where $D = D(h, h_x)$, the exact value is not known but must be approximated. However, there is a big difference in difficulty between *solving* for D at the future time (as in methods 5 and 6) to approximate D and using an equally accurate linear combination of the older values (as in the current method).

If one combines this extrapolation idea with Crank–Nicolson evaluation of the second spatial derivative of h in the ice continuity equation, one gets a new method, as follows. It is as accurate as full Crank–Nicolson but does not have the difficulty of solving nonlinear equations.[3] On the other hand it can be expected to be less stable.

Written out in detail one has:

$$\begin{aligned}\frac{h_{j,l+1} - h_{j,l}}{\Delta t} = B+ \\ \frac{1}{2\Delta x}\left(D_{j+1/2,\star}\left(\frac{h_{j+1,l} - h_{j,l}}{\Delta x}\right) - D_{j-1/2,\star}\left(\frac{h_{j,l} - h_{j-1,l}}{\Delta x}\right)\right) + \\ \frac{1}{2\Delta x}\left(D_{j+1/2,\star}\left(\frac{h_{j+1,l+1} - h_{j,l+1}}{\Delta x}\right) - D_{j-1/2,\star}\left(\frac{h_{j,l+1} - h_{j-1,l+1}}{\Delta x}\right)\right)\end{aligned} \tag{25}$$

where "$\star$" represents extrapolation to $t_{l+1/2}$ and thus

$$D_{j+1/2,\star} = \frac{3}{2}D_{j+1/2,l} - \frac{1}{2}D_{j+1/2,l-1}. \tag{26}$$

Of course, $D_{j+1/2,l}$, etc., are calculated by (18) for this type II scheme but could equally well be done by type I.

I emphasize that the extrapolation is done only for the "diffusivity" D. Equation (25) is solved in the usual tridiagonal manner. The three dimensional version of this method is compatible with an ADI scheme. Clearly two previous grids of D values

[3]The local truncation error needs to be tediously calculated, or rather with *Mathematica*. It is $O(\Delta x^2, \Delta t^2)$, I believe.

must be kept, but this is a small price to avoid Newton–Raphson, if that turns out to be possible from a stability point of view.

See the stencil in figure 8. Again, a related type I method is also possible.

The following table continues the comparison of stability. These results are from `iceB.m` in appendix M. Compare to the table on page 10.

	$N_x = 30$	$N_x = 100$	$N_x = 200$
Method 7	33 [1.10]	2 [32.9]	.6 [173]

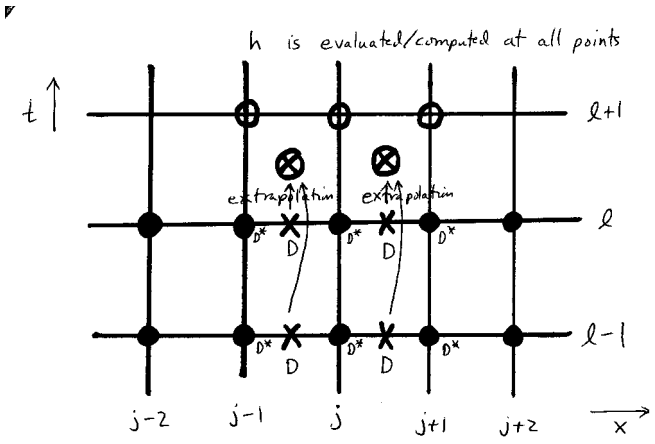

FIGURE 8. Method 7 for (14).

3. The free boundary value problem for T: A first experiment

It is clear that equations (1) through (11) can be collected together in a somewhat nicer package, and I will do so in the next section.

For the moment there is a boundary issue to settle. Note that the ice surface boundary condition (10) is not along a fixed surface in x, z space, but rather along the graph of $z = h(x, t)$ which is determined by the continuity equation (1) which is coupled to the temperature equation (3) through the Arrhenius relation (5). In other words, T solves a *free boundary problem*.[4]

This fact has been widely, but not universally, recognized, I think. D. Jenssen [15] introduced a change of vertical coordinate which, in essence, exchanges the free

[4]In fact h *also* solves a free boundary problem in general, but in this "fixed margin" experiment I take the boundary at fixed location.

boundary nature of the problem for a *more singular* temperature equation. In particular, if H is the thickness of the sheet and if

$$s = \frac{h-z}{H}$$

then (3) becomes

$$\frac{\partial T}{\partial t} + (\text{advection}) = \frac{k}{\rho C_p}\frac{1}{H^2}\frac{\partial^2 T}{\partial s^2} + (\text{heat source}),$$

on the fixed domain $\{-L \le x \le L, 0 \le s \le 1\}$.

Note that the vertical diffusivity coefficient is then unbounded as $H \to 0$ at the boundary of the ice sheet. I suspect this is *a* (possibly *the*?) source of "instability in solving the temperature equation." Nonetheless, apparently usable schemes based on the Jenssen change do exist, for instance [6, 15, 22, 27]. K. Hutter, at least, is well–aware of the free boundary nature of the model. See [12] and the references therein.[5]

I have an idea on how to deal, in practice, with this free boundary problem in a rather simple manner. To explain this in a convenient context, and for a first experiment, I make an additional assumption for *this section only*:

Temporary Assumption 2 (Section 3 only)**.** The horizontal advection term $u\frac{\partial T}{\partial x}$ in (3) is replaced by zero.

This is not a good physical approximation, but rather allows the consideration of temperature in a single column of the ice in isolation from the others. See section 7.2 of [27]. I will not make this assumption in section 4.

Recall that assumption 1 of the previous section means h, u, and w can be found independently of T. I will use the time–dependent values of h, u, w in solving for T in the column.

From these assumptions, $T = T(x,z,t)$ satisfies

$$\frac{\partial T}{\partial t} + w\frac{\partial T}{\partial z} = \frac{k}{\rho C_p}\frac{\partial^2 T}{\partial z^2} + \frac{g(h-z)}{C_p}\left|\frac{\partial h}{\partial x}\frac{\partial u}{\partial z}\right|.$$

Let $\bar{x}$ be a fixed horizontal position. The above equation corresponds to a column at $\bar{x}$ which is thermally–isolated because there is no horizontal conduction or convection (advection). Only vertical advection, vertical conduction, and a dissipation–driven heat source are included.

Let $T(z,t) = T(\bar{x},z,t)$, $h(t) = h(\bar{x},t)$, $\alpha(t) = \left|\frac{\partial h}{\partial x}(\bar{x},t)\right|$, $u(z,t) = u(\bar{x},z,t)$, and $w(z,t) = w(\bar{x},z,t)$ for the rest of this section. Let $\Sigma(z,t) = \frac{g(h-z)}{C_p}\alpha\left|\frac{\partial u}{\partial z}\right|$, $K_T = \frac{k}{\rho C_p}$, and $T_s = 239 + 8\times 10^{-8}|\bar{x}|$.

[5] I do not know what numerical methods have been used by his group, though it seems that R. Greve's program SICOPOLIS [6] does use the Jenssen change of coordinate.

I consider the problem:

$$\frac{\partial T}{\partial t} + w(z,t)\frac{\partial T}{\partial z} = K_T \frac{\partial^2 T}{\partial z^2} + \Sigma(z,t), \tag{27}$$

$$T(h(t),t) = T_s, \tag{28}$$

$$\frac{\partial T}{\partial z}(0,t) = -\frac{G}{k} \text{ if } T(0,t) \le \tilde{T}(o,t) \text{ and otherwise } T(0,t) = \tilde{T}(0,t) \tag{29}$$

$$\text{where } \tilde{T}(z,t) = T_r - \beta(h(t) - z).$$

The outstanding fact about this boundary value problem for temperature is that the Dirichlet boundary condition (28) is at a position which *moves*. That position is derived from the simultaneous solution of the continuity equation (14).

Note that (27) is otherwise a classical convection diffusion equation in one dimension, and much literature applies to its solution [17, 25].

Now for my idea. To avoid the Jenssen coordinate change I suppose $T(z,t)$ is defined for $t \ge 0$ and $0 \le z \le h_{max}$ where h_{max} is a hypothesized maximum ice thickness. I replace the boundary condition (28) by the fixed condition

$$T(h_{max},t) = T_s. \tag{30}$$

Replace the temperature equation (27) by

$$\frac{\partial T}{\partial t} + w(z,t)\frac{\partial T}{\partial z}(z < h(t)) = K_T \frac{\partial^2 T}{\partial z^2} + \Sigma(z,t)(z < h(t)) + \omega(T_s - T)(z \ge h(t)) \tag{31}$$

where "$(a < b)$" is one if true and zero if false (this is the *Matlab* style). That is, for $z < h(t)$, (27) holds, but for $z \ge h(t)$,

$$\frac{\partial T}{\partial t} = K_T \frac{\partial^2 T}{\partial z^2} + \omega(T_s - T). \tag{32}$$

I keep the vertical diffusion in (32) in order to smooth the transition of T between the $z < h$ and $z \ge h$ regions. (The inclusion of this term is likely *not* essential.)

Now, $\omega > 0$ is a new parameter, and is supposed large enough to rapidly decay T to the value T_s. How large should ω be? I choose the time step Δt as a half–life for $T_s - y$ in the "Newton's law of cooling" equation $\frac{dy}{dt} = \omega(T_s - y)$. That is, $e^{-\omega \Delta t} = 1/2$ or $\omega = \ln 2/\Delta t$. Clearly (32) is a diffusive version of Newton's law of cooling.

I discretize the z direction by $z_k = 0 + (k-1)\Delta z$ for $k = 1, \ldots, N_z + 1$ where $\Delta z \cdot N_z = h_{max}$. Vertical spacing is uniform for simplicity but this is generalized in the next section. Let $T_{k,l}$ be our approximation of $T(z_k, t_l)$. Equation (31) is approximated semi–implicitly by

$$\begin{aligned} \frac{T_{k,l+1} - T_{k,l}}{\Delta t} + w(z_k,t_l)\frac{T_{k+1,l} - T_{k-1,l}}{2\Delta z}(z_k < h(t_l)) \\ = K_T \frac{T_{k+1,l+1} - 2T_{k,l+1} + T_{k-1,l+1}}{\Delta z^2} + \Sigma(z_k,t_l)(z_k < h(t_l)) \\ + \omega(T_s - T_{k,l+1})(z_k \ge h(t_l)). \end{aligned} \tag{33}$$

Note that the diffusion and Newton's law terms are treated implicitly. Clearly a higher order implicit scheme is reasonable and will not involve a great increase in work because (31) is linear in T.

The basal boundary condition on T is taken as the rule: If $T_{1,l} < \tilde{T}(0,t_l)$ or $\frac{T_{2,l}-T_{1,l}}{\Delta z} < -\frac{G}{k}$ then $\frac{T_{2,l}-T_{1,l}}{\Delta z} = -\frac{G}{k}$ and otherwise $T_{1,l} = \tilde{T}(0,t_l)$. Also, at the end of the step if any $T_{k,l+1}$ is found to exceed $\tilde{T}(z_k,t_l)$ it is set to that value. This last accommodation to the nature of water is *ad hoc* and should be replaced by a "Stefan boundary condition at the free cold–temperate interface." See [6, 11].

A *Matlab* code `iceC.m` based this method appears in appendix M with representative output. It solves the continuity equation (14) by the semi–implicit type II method 4. That is, on the right side of $\frac{\partial h}{\partial t} = B + \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x}\right)$, D is type II evaluated at the old time step but h is evaluated at the new time step. Qualitatively reasonable and stable results are found. It is possible to choose Δt up to approximately 300 years in a 25000 year run with $\Delta x = 50$ km, and still get apparent stability. I believe, in other words, that the temperature model is included at *no stability cost at all*. It is the continuity equation which has an intrinsic stability limit on the time step [8].

4. Numerical solution method for the coupled equations

I now return to equations (1) through (11) and seek a numerical solution to the coupled system. The decoupling and simplifying assumptions of the previous section are dropped. Also, the parameter n will be an adjustable parameter.

A reorganization of the equations is appropriate. In particular, it is useful to identify a "diffusivity" analogous to D in section 2.

Let $\Gamma_0 = 2(\rho g)^n$. Let $\alpha(x,t) = \left|\frac{\partial h}{\partial x}(x,t)\right|$ as before. For $0 \le z \le h(x,t)$, let

$$\delta = f\Gamma_0 A(T)\alpha^{n-1}(h-z)^n.$$

Note $\delta \ge 0$ and $\delta > 0$ when $\frac{\partial h}{\partial x} \neq 0$ and $z < h(x,t)$. I introduce f as a flow enhancement factor which may be useful for adjusting the model to better agreement with experimentally observed overall dimensions for ice sheets.

I claim δ can be treated somewhat analogously to D as in section 2. It may be called a "local diffusivity rate", which is justified by the following rewriting of the ice continuity equation:

$$\frac{\partial h}{\partial x} = B + \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x} - u_b h\right) \quad \text{where} \quad D(x,t) = \int_0^{h(x,t)} \delta(x,\zeta,t)(h(x,t)-\zeta)\,d\zeta.$$

The velocities u, w and the dissipation heat source $\Sigma = \frac{g}{C_p}(h-z)\alpha\left|\frac{\partial u}{\partial z}\right|$ can all be written in terms of δ or integrals of δ, as follows. See appendix D for the calculations which produce these formulas. Note that only the form of δ is affected by choice of flow law (e.g. choice of n) and by the form of the Arrhenius relation, which is computed by the function `arr.m` in appendix M.

I claim equations (34) through (44) are equivalent to (1) through (11):

$$\delta(x,z,t) = f\Gamma_0 A(T(x,z,t))\alpha(x,t)^{n-1}(h(x,t)-z)^n, \tag{34}$$

$$I(x,z,t) = \int_0^z \delta(x,\zeta,t)\,d\zeta, \qquad J(x,z,t) = \int_0^z \delta(x,\zeta,t)(z-\zeta)\,d\zeta, \tag{35}$$

$$u(x,z,t) = -\frac{\partial h}{\partial x}(x,t)I(x,z,t) + u_b(x,t); \quad u_b = 0, \tag{36}$$

$$w(x,z,t) = \frac{\partial}{\partial x}\left(J(x,z,t)\frac{\partial h}{\partial x}(x,t)\right) - z\frac{\partial u_b}{\partial x}(x,t) + w_b(x,t); \quad w_b = 0, \tag{37}$$

$$\Sigma(x,z,t) = \frac{g}{C_p}\alpha(x,t)^2(h(x,t)-z)\delta(x,z,t), \tag{38}$$

$$D(x,t) = J(x,h(t),t), \tag{39}$$

$$\frac{\partial h}{\partial t} = B + \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x} - u_b h\right), \tag{40}$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + w\frac{\partial T}{\partial z} = K_T\frac{\partial^2 T}{\partial z^2} + \Sigma, \tag{41}$$

$$h(-L,t) = h(L,t) = 0; \quad L = 750 \text{ km}, \tag{42}$$

$$T(x,h,t) = T_s(x) = 239 + (8\times 10^{-8})|x| \tag{43}$$

$$\frac{\partial T}{\partial z}(x,0,t) = -\frac{G}{k} \text{ if } T(x,0,t) \le \tilde{T}(x,0,t), \text{ and otherwise} \tag{44}$$
$$T(x,0,t) = \tilde{T}(x,0,t) \text{ where } \tilde{T}(x,z,t) = T_r - \beta(h-z).$$

Clearly (34) through (39) are computations done at each step to solve the main equations (40), (41) subject to the boundary conditions (42), (43), (44).

For the continuity equation (40) I choose the same simple and stable method as was chosen in section 3. That is, method 4 which is semi–implicit and type II. Again, because this is type II, I need to augment the system above with the boundary conditions $D(-L,t) = D(L,t) = 0$. Other methods are obviously possible and appropriate.

Let $x_j = -L+(j-1)\Delta x$, $j = 1,\ldots,N_x+1$, with $\Delta x\cdot N_x = 2L$. Let $t_l = 0+(l-1)\Delta t$, $l = 1,\ldots,M_t+1$ with $\Delta t\cdot M_t = t_{end}$. Let $h_{j,l}$ be the approximation to $h(x_j,t_l)$. Let $\delta_{j,l}(z)$ denote the approximation to $\delta(x_j,z,t_l)$ and similarly for I, J, u, w and Σ.

Equations (34) through (40) then are easily discretized in the horizontal and time dimensions: for $j = 2,\ldots,N_x$

$$\delta_{j,l}(z) = f\Gamma_0 A(T_{j,l}(z))\left|\frac{h_{j+1,l}-h_{j-1,l}}{2\Delta x}\right|^{n-1}(h_{j,l}-z)^n, \tag{45}$$

$$I_{j,l}(z) = \int_0^z \delta_{j,l}(\zeta)\,d\zeta, \qquad J_{j,l}(z) = zI_{j,l}(z) - \int_0^z \delta_{j,l}(\zeta)\zeta\,d\zeta, \tag{46}$$

$$u_{j,l}(z) = -\frac{h_{j+1,l} - h_{j-1,l}}{2\Delta x} I_{j,l}(z), \tag{47}$$

$$w_{j,l}(z) = \frac{1}{2\Delta x^2}\Big[(J_{j+1,l}(z) + J_{j,l}(z))(h_{j+1,l} - h_{j,l}) - (J_{j,l}(z) + J_{j-1,l}(z))(h_{j,l} - h_{j-1,l})\Big], \tag{48}$$

$$\Sigma_{j,l}(z) = \frac{g}{C_p}\left|\frac{h_{j+1,l} - h_{j-1,l}}{2\Delta x}\right|^2 (h_{j,l} - z)\delta_{j,l}(z), \tag{49}$$

$$D^*_{j,l} = J_{j,l}(h_{j,l}); \quad D_{j+1/2,l} = \frac{1}{2}\left(D^*_{j,l} + D^*_{j+1,l}\right), \tag{50}$$

$$\frac{h_{j+1,l} - h_{j,l}}{\Delta t} = B + \frac{1}{\Delta x^2}\left[D_{j+1/2,l}\left(h_{j+1,l+1} - h_{j,l+1}\right) - D_{j-1/2,l}\left(h_{j,l+1} - h_{j-1,l+1}\right)\right]. \tag{51}$$

For $j = 1$ and $j = N_x + 1$, D^* (but not δ!) may be taken as zero because the thickness h is zero.

For the vertical discretization, it is worthwhile to add a level of sophistication to that of the previous section. That is, to allow a not equally–spaced ("general") grid. In this I follow Payne & Dongelmans [22] except that I do not make the Jenssen change of variables. See appendix C of the current paper for the derivation of finite difference schemes for general grids. In particular, suppose $h_{max} > 0$ is a fixed maximum thickness of the ice sheet. Suppose $\{z_k\}_{k=1}^{N_z+1}$ are given with $0 = z_1 < z_2 < \cdots < z_{N_z+1} = h_{max}$. For instance, [22] use the eleven levels $\{z_k\} = \{0.00, 0.02, 0.05, 0.10, 0.17, 0.25, 0.40, 0.55, 0.70, 0.85, 1.00\}$ (in the context of the Jenssen change of variables).

I need to suppose that integrals can be approximately computed in the vertical direction. Let $I_{j,k,l}$ be an approximation to $I_{j,l}(z_k)$ and similarly for $J_{j,k,l}$. See `vintlist.m` in appendix M for the details of the integration. For now, a trapezoid scheme is used. More sophisticated methods are possible. Note $J_{j,l}(h_{j,l})$ can be found by linear interpolation of $J_{j,k,l}$ and $J_{j,k+1,1}$ where $z_k \le h_{j,l} < z_{k+1}$.

In discretizing the temperature equation (41), let $T_{j,k,l}$ be the approximation to $T(x_j, z_k, t_l)$. The z derivatives in (41) are approximated on the not–equally–spaced grid. Also, the first derivatives $\frac{\partial T}{\partial x}$, $\frac{\partial T}{\partial z}$ are going to be approximated by "second–order upwinding" as in [22]. In fact, for the horizontal direction let

$$\mathrm{Up}(T_{k,l}\big|j,\lambda) = \begin{cases} \lambda \cdot \frac{(1/2)T_{j-2,k,l} - 2T_{j-1,k,l} + (3/2)T_{j,k,l}}{\Delta x} & \text{if } \lambda \ge 0, \\ \lambda \cdot \frac{-(3/2)T_{j,k,l} + 2T_{j+1,k,l} - (1/2)T_{j+2,k,l}}{\Delta x} & \text{if } \lambda < 0. \end{cases}$$

For the vertical direction let

$$\mathrm{Up}(T_{j,l}\big|k,\lambda) = \begin{cases} \lambda \cdot (\text{approximation (C4)}) & \text{if } \lambda \ge 0, \\ \lambda \cdot (\text{approximation (C5)}) & \text{if } \lambda < 0. \end{cases}$$

The various three–point approximations are given in appendix C. "Upwinding" is computed by `upwindhor.m`, `upwindvert.m` in appendix M.

Now let $\delta_{j,k,l}$ approximate $\delta_{j,l}(z_k)$ and similarly for $u_{j,k,l}$, $w_{j,k,l}$, $\Sigma_{j,k,l}$, $T_{j,k,l}$. In these terms I now discretize the temperature equation (41):

$$\begin{aligned}(52)\qquad &\frac{T_{j,k,l+1}-T_{j,k,l}}{\Delta t} + \mathrm{Up}(T_{k,l}\big|j,u_{j,k,l})\cdot(z_k<h_{j,l}) + \mathrm{Up}(T_{j,l}\big|k,w_{j,k,l})\cdot(z_k<h_{j,l})\\ &\quad = K_T\begin{pmatrix}\text{approximation (C6) or (C7)}\\ \text{at } (x_j,t_{l+1})\end{pmatrix} + \Sigma_{j,k,l}\cdot(z_k<h_{j,l})\\ &\qquad + \omega(T_s(x_j)-T_{j,k,l+1})\cdot(z_k\geq h_{j,l}).\end{aligned}$$

Compare to (33).

Of course this represents a linear system to be solved. For the interior points $k=3,\dots,N_z-1$, each equation can be written

$$(53)\qquad [1+\Delta t\omega(z_k\geq h_{j,l})]\,T_{j,k,l+1} - \Delta t K_T\,[(\text{C6 at } (x_j,t_{l+1})] = b_{j,k,l}$$

where

$$\begin{aligned}(54)\qquad b_{j,k,l} &= T_{j,k,l} - \Delta t(z_k<h_{j,l})\left[\mathrm{Up}(T_{k,l}\big|j,u_{j,k,l}) + \mathrm{Up}(T_{j,l}\big|k,w_{j,k,l}) - \Sigma_{j,k,l}\right]\\ &\quad + \Delta t\omega T_s(x_j)(z_j\geq h_{j,l})\end{aligned}$$

involves only quantities known at $t=t_l$. The $k=1,2$ and $k=N_z$ cases differ because they involve boundary conditions. Note that the equation at the base ($k=1$) is trivial in the warm–ice case (i.e. if $T_{j,1,l}=\tilde{T}(x_j,0,t_l)$). In the cold ice case (if $T_{j,1,l}<\tilde{T}(x_j,0,t_l)$) the geothermal heat flux (Neumann) condition is imposed using (C5).

In any case, for each $j=2,\dots,N_x$ this represents a pentadiagonal system of N_z equations in N_z unknowns $\{T_{j,k,l}\}_{k=1}^{N_z}$. See `pentaT.m` and `iceD.m` for the details of implementation.

The choice of the parameter ω is as described in section 3: $\Delta t\omega=\ln(2)$.

Consider the main time loop:

- Suppose the quantities $h_{j,l}$, $T_{j,k,l}$ are known at time $t=t_l$;
- compute $\delta_{j,k,l}$—here is where Glen flow law and Arrhenius relation appear;
- compute $I_{j,k,l}$ and $J_{j,k,l}$ by integration (`vintlist.m`);
- compute $D_{j+1/2,l}$ by interpolating J;
- compute $u_{j,k,l}$, $w_{j,k,l}$, $\Sigma_{j,k,l}$;
- solve the continuity equation (51), incorporating boundary conditions (42), by a single (N_x-1) by (N_x-1) tridiagonal matrix solve (w. no iteration);
- solve the temperature equation (52), incorporating boundary conditions (43),(44), by pentadiagonal means; requires N_x-1 solves of N_z by N_z pentadiagonal matrices;
- loop.

Several points are worth making. Note the systems of equations are linear and band–limited (i.e. tri– or penta–diagonal), which follows from two properties. First, the continuity equation is approximated by a semi–implicit scheme. Methods 3 and 4 from section 2 are of this type, but so is method 7, though it is multistep and its stability properties need to be explored. Second, the coefficients u, w, Σ of the lower order terms in the temperature equation, which also couple the system, are computed at the previous time step. Thus the temperature equation is computed semi–implicitly. A "Crank–Nicolson scheme" could *easily* be used on the temperature equation, as long as these coefficients are computed at previous time steps. In order to get further accuracy benefit either a nonlinear system has to be solved or the coefficients have to be extrapolated by a multi–step scheme.

In any case, see `iceD.m` in appendix M. The following are outputs are interesting enough to include in the text.

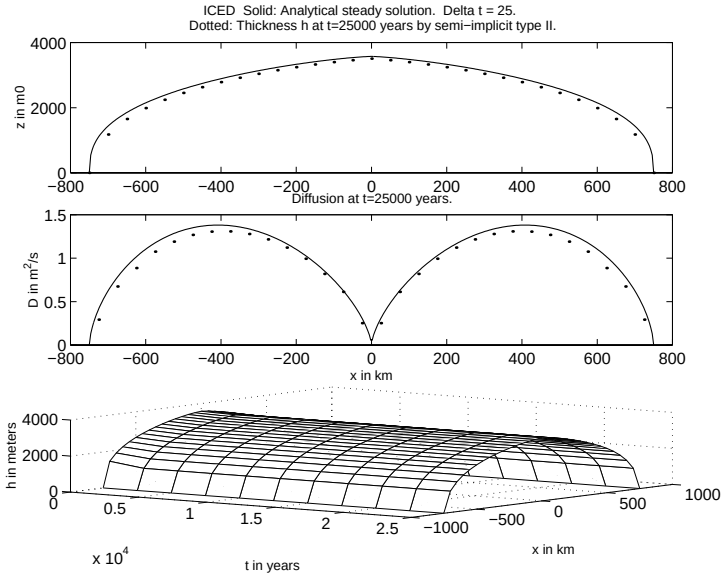

FIGURE 9. For comparison, as the Arrhenius relation $A(T)$ is replaced by a constant (thus this is *decoupled*). Note close agreement with analytical h and D profile at last time. Note $n = 3$ in Glen law. This is output figure 1 from `iceD.m`.

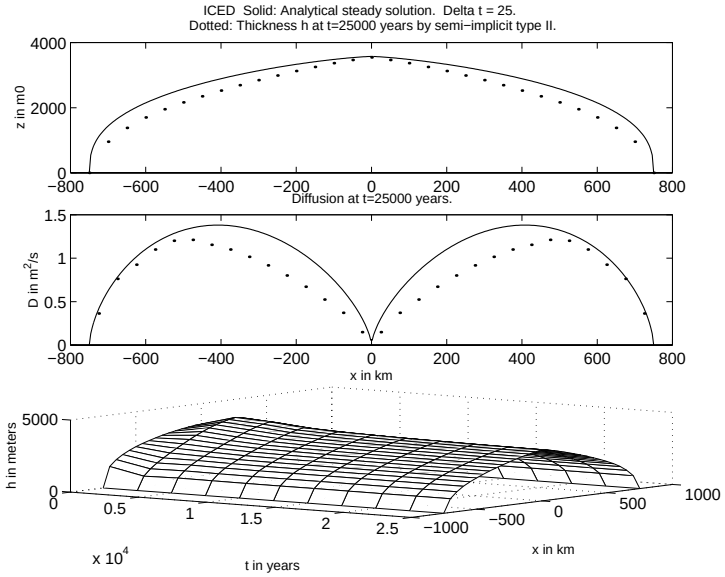

FIGURE 10. Same as above but *coupled* through the Arrhenius relation. A flow enhancement factor of $f = 5$ is used. Note the more pronounced central ridge.

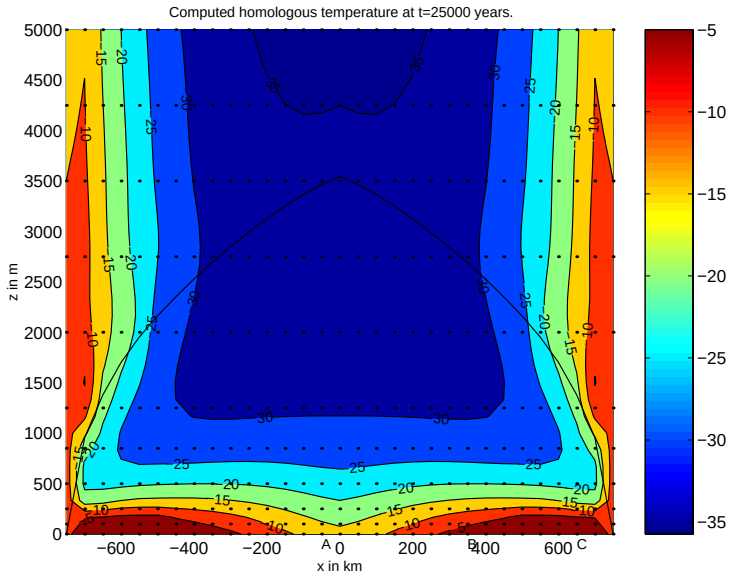

FIGURE 11. The computed temperature field for the same case as above. Ignor the computed temperatures above the computed h profile, shown. Note positions A,B,C for the next figure. This is output figure 2 from the same run as above.

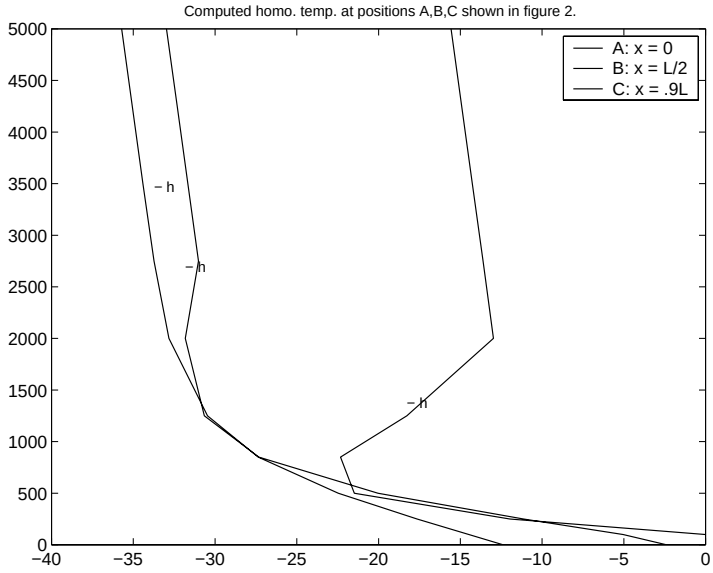

FIGURE 12. Computed temperature profiles at positions A,B,C in the previous figure. Valid only up to the level of the surface h, shown. This is output figure 3 from the same run as above.

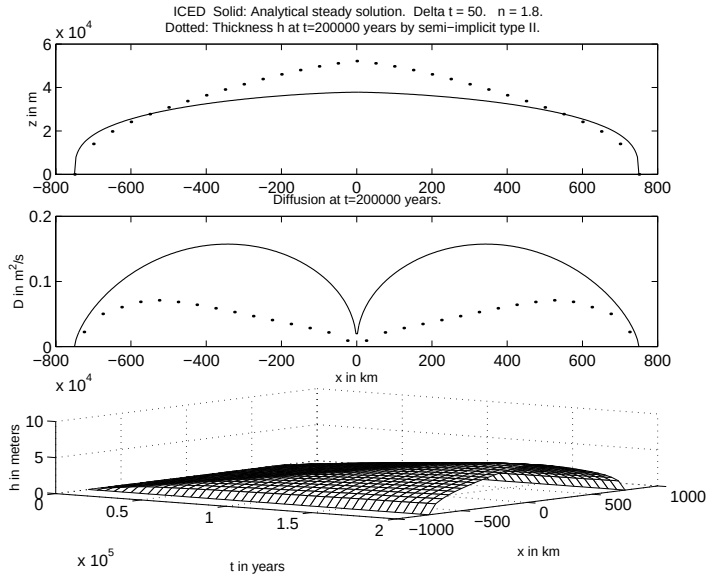

FIGURE 13. This shows two variations on the previous experiment, namely that $n = 1.8$ in the Glen flow law and that $h \equiv 0$ is the initial condition. Again $f = 5$. Note that equilibrium may or may not have been achieved by this time ($t = 200,000$ years).

References

[1] L. Bowman, *UAF Master's Project* (2002).

[2] R. L. Burden and J. D. Faires, *Numerical Analysis* 7th Ed., Brooks/Cole 2001.

[3] N. Calvo, J. Durany, and C. Vazquez, Numerical approach of temperature distribution in a free boundary model for polythermal ice sheets, *Numerische Mathematik* (1999) **83**, 557–580.

[4] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge 1996.

[5] J. W. Glen, The creep of polycrystalline ice, *Proc. Roy. Soc. London* (1955) Ser. A **228**, 519–538.

[6] R. Greve, A continuum–mechanical formulation for shallow polythermal ice sheets, *Phil. Trans. Royal Soc. London A* (1997) **355**, 921–974.

[7] R. C. A. Hindmarsh, On the numerical computation of temperature in an ice sheet, *J. Glaciology* (1999) **45**(151), 568–574.

[8] R. C. A. Hindmarsh and A. J. Payne, Time–step limits for stable solutions of the ice–sheet equation, *Ann. Glaciology* (1996) **23**, 74–85.

[9] R. Hooke, Flow law for polycrystalline ice in glaciers: comparison of theoretical predictions, laboratory data, and field measurements, *Rev. Geophys. Space. Phys.* (1981) **19**(4), 664–672.

[10] K. Hutter, *Theoretical Glaciology*, D. Reidel 1983.

[11] K. Hutter, Thermomechanically coupled ice–sheet response: cold, polythermal,temperate, *J. Glaciology* (1993) **39**(131), 65–86.

[12] K. Hutter, Mathematical foundation of ice sheet and ice shelf dynamics, a physicist's view, in *Free Boundary Problems: Theory and Applications*, I. Athanasopoulos, et. al., (editors) Chapman & Hall 1999.

[13] P. Huybrechts, A 3–D model for the Antarctic ice sheet: a sensitivity study on the glacial–interglacial contrast, *Climate Dynamics* (1990) **5**, 79–92.

[14] P. Huybrechts, et. al., The EISMINT benchmarks for testing ice–sheet models, *Ann. Glaciology* (1996) **23**, 1–12.

[15] D. Jenssen, A three–dimensional polar ice–sheet model, *J. Glaciology* (1977) **18**, 373–389.

[16] M. W. Mahaffy, A three–dimensional numerical model of ice sheets: tests on the Barnes ice cap, Northwest Territories, *J. Geophys. Res.* (1976) **81**(6), 1059–1066.

[17] K. W. Morton and D. F. Mayers, *Numerical Solution of Partial Differential Equations*, Cambridge 1994.

[18] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere 1980.

[19] W. S. B. Paterson, *The Physics of Glaciers* 3rd Ed., Pergamon 1994.

[20] A. J. Payne, A thermomechanical model of ice flow in West Antarctica, *Climate Dynamics* (1999) **15**, 115–125.

[21] A. J. Payne and D.J. Baldwin, Analysis of ice–flow instabilities identified in the EISMINT intercomparison exercise, *Ann. Glaciology* (2000) **30**, 204–210.

[22] A. J. Payne and P. W. Dongelmans, Self–organization in the thermomechanical flow of ice sheets, *J. Geophys. Res.* (1997) **102**(B6), 12219–12233.

[23] A. J. Payne et al., Results from the EISMINT model intercomparison: the effects of thermomechanical coupling, *J. Glaciology* (2000) **153**, 227–238.

[24] W. H. Press, et. al., *Numerical Recipes in C: The Art of Scientific Computing* 2nd Ed., Cambridge 1992.

[25] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial–Value Problems* 2nd Ed., Wiley 1967.

[26] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth 1989.

[27] C. J. van der Veen, *Fundamentals of glacier dynamics*, Balkeema 1999.

Appendix A. Analytical steady solution to the continuity equation

Consider the steady–state case of (14) which, in the $n = 3$ case, is

$$0 = B + \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x}\right), \qquad D = \frac{\Gamma}{5}h^5\left|\frac{\partial h}{\partial x}\right|^2. \tag{A1}$$

I determine $h(x)$ and $D(x)$ for the dual purpose of checking accuracy and building intuition.

It appears[6] that the boundary conditions which give a physically reasonable solution are $h'(0) = 0$ and $h(L) = 0$. Clearly symmetry considerations show that if h solves (A1) with $h(-L) = 0$ and $h(L) = 0$ and if $h'(0)$ exists then $h'(0) = 0$.

Integrating (A1) from 0 to x gives

$$0 = Bx + \frac{\Gamma}{5}h(x)^5h'(x)^3.$$

Integrating this from x to L gives

$$h(x) = C\left(L^{4/3} - x^{4/3}\right)^{3/8}, \qquad C = 2^{3/8}\left(\frac{5B}{\Gamma}\right)^{1/8}.$$

Note $h(0) = CL^{1/2} \approx 3575.06$ m. This value can be compared to the computed value `H0` from `iceA.m`, `iceB.m`, etc. See figure 9.

One gets the following analytical expression for the steady–state diffusivity:

$$D(x) = \frac{\Gamma C^7}{20}x^{2/3}\left(L^{4/3} - x^{4/3}\right)^{5/8}.$$

Note D is even and $D(0) = 0$ but $D'(0)$ does not exist. Note $D(-L) = D(L) = 0$ as well. The maximum of D occurs roughly halfway between $x = 0$ and $x = L$. The analytical value of $D(L/2) \approx 1.37396\ \frac{\text{m}^2}{\text{s}}$ can be compared to numerical results. See figure 9.

At risk of repeating myself, note $D'(0)$ does not exist. However, $Q(x) = D(x)h'(x)$ is differentiable for $-L < x < L$. In fact,

$$Q(x) = -Bx$$

as is expected from constant accumulation.

I suppose this is an explanation for the following quandary: Methods based on discretizing $h_t = B + \Gamma h^4h_x^4 + \frac{3\Gamma}{5}h^5h_x^2h_{xx}$ all seem to be unstable. That is, I discretize $h_t = B + D_xh_x + Dh_{xx}$ but then have a factor of D_x, which goes to infinity as $h_x \to 0$, multiplying h_x. The empirical instability indeed "comes from near $x = 0$" in running such a scheme. By contrast, the successful methods of section 2 are based

[6] A mystery is why $h(-L) = 0$, $h(L) = 0$ as boundary conditions do not seem to give this solution.

on discretizing $h_t = (Dh_x)_x$ in that form.[7] A more convincing explanation of this quandary is given in appendix B.

For the general n case, the above formulas become:

$$0 = B + \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x}\right), \quad D = \frac{\Gamma}{n+2}h^{n+2}\left|\frac{\partial h}{\partial x}\right|^{n-1}, \quad \Gamma = 2(\rho g)^n A,$$

$$h(x) = C_1\left(L^{\frac{n+1}{n}} - |x|^{\frac{n+1}{n}}\right)^{\frac{n}{2n+2}}, \quad C_1 = 2^{\frac{n}{2n+2}}\left(\frac{(n+2)B}{\Gamma}\right)^{\frac{1}{2n+2}},$$

$$D(x) = C_2|x|^{\frac{n-1}{n}}\left(L^{\frac{n+1}{n}} - |x|^{\frac{n+1}{n}}\right)^{\frac{n+2}{2n+2}}, \quad C_2 = \frac{\Gamma C_1^{2n+1}}{(n+2)2^{n-1}}.$$

Thus $h(0) = C_1 L^{1/2}$ and $D(L/2) = C_2 2^{-3/2}\left(2^{\frac{n+1}{n}} - 1\right)^{\frac{n+2}{2n+2}} L^{3/2}$.

The above formulas appear in `iceconstants.m`. Again, they are useful for checking the decoupled steady state results.

Appendix B. Maximum principles for finite difference approximations to diffusion equations.

The real title of this appendix is:

Why one has to discretize the ice flow equation in "self–adjoint" form.

In fact, by following section 2.15 of [17], I now answer a question I raised in appendix A. Namely, if one discretizes the right side of (14), that is,

$$\frac{\partial h}{\partial t} = B + \frac{\partial}{\partial x}\left(D\frac{\partial h}{\partial x}\right), \tag{B1}$$

as

$$B + \frac{1}{\Delta x^2}\left[D_{j+1/2}\left(h_{j+1} - h_j\right) - D_{j-1/2}\left(h_j - h_{j-1}\right)\right]$$

then things seem to work well—that is, seem to be stable—even with explicit methods, and for both type I or type II diffusivity approximation, and so on. In the ice case, where $D = D(h, h_x)$, there are indeed stability limits (see section 2 of these notes and [8]), but they are tolerable in practice.

On the other hand, if one rewrites (B1) in the equivalent form

$$\frac{\partial h}{\partial t} = B + \frac{\partial D}{\partial x}\frac{\partial h}{\partial x} + D\frac{\partial^2 h}{\partial x^2} \tag{B2}$$

[7] In practice, that is, computing numerically, Q is a relatively well-behaved function but it seems to be the values of D which are first victims of instability. One *computes* D either way, since one computes Q as $-D\frac{\partial h}{\partial x}$. It is therefore worth seeing the less stable quantity and determining stability limits from it.

and discretizes the right side as

$$B + \left(\frac{D_{j+1} - D_{j-1}}{2\Delta x}\right)\left(\frac{h_{j+1} - h_{j-1}}{2\Delta x}\right) + \frac{D_j}{\Delta x^2}\left(h_{j+1} - 2h_j + h_{j-1}\right)$$

then there seem to be unacceptable stability problems *even if* implicit methods are used, *even if* the new first derivative term is dealt with by upwinding, *regardless* of the type I/II choice, and so on. (For very small time steps one may be able to use this method on the ice flow equation (B1) [1], at least for initial conditions near the analytical steady state.)

I claim that the distinction between these two seemingly equivalent schemes can be seen through a *maximum principle*. If one shows that a scheme for evolution equations has the property that the maximum value on a grid is always achieved at the boundary then the scheme will be stable [17]. That is, a maximum principle of the type below is sufficient for stability.

For an example before addressing the *nonlinear* diffusion equation (B1), consider the explicit method for the heat equation $u_t = u_{xx}$:

$$\text{(B3)}\quad \frac{u_{j,l+1} - u_{j,l}}{\Delta t} = \frac{u_{j+1,l} - 2u_{j,l} + u_{j-1,l}}{\Delta x^2} \quad \text{or} \quad u_{j,l+1} = a_1 u_{j+1,l} + a_2 u_{j,l} + a_3 u_{j-1,l}$$

where $a_1 = R, a_2 = 1 - 2R, a_3 = R$ and $R = \frac{\Delta t}{\Delta x^2}$. Note $a_1 + a_2 + a_3 = 1$. *If* $a_2 \geq 0$, i.e. if

$$\text{(B4)}\qquad \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2},$$

then

$$\text{(B5)}\qquad u_{j,l+1} \leq \max\{u_{j+1,l}, u_{j,l}, u_{j-1,l}\}.$$

This follows both because all of the a_i are positive *and* because they sum to one. One can now easily show that the maximum value of the $\{u_{j,l}\}$ is achieved at the boundary or as an initial value, which is the maximum principle part. What does this have to do with stability? If (B5) holds then the classic instability which appears on the right of figure 14 cannot occur because values at future time ($u_{j,l+1}$) exceed the past values ($u_{j-1,l}, u_{j,l}, u_{j+1,l}$). The graph on the left has shorter time steps satisfying (B4) and "can not" show this instability because a maximum principle holds true.

For the nonhomogeneous equation $u_t = B + u_{xx}$ where B is constant, note $v = u - Bt$ solves $v_t = v_{xx}$. This justifies considering only the $B = 0$ case. Returning to the ice continuity equation (B1), which is nonlinear, one addresses the stability around a given solution—for instance the steady state solution—by looking at the equation satisfied by deviation from that solution. In fact, if h solves (B1) and if $\bar{h}$ solves the steady state version ($\bar{h}_t = 0 = B + (D(\bar{h}, \bar{h}_x)\bar{h}_x)_x$) then the deviation $\eta = h - \bar{h}$ solves

$$\frac{\partial \eta}{\partial t} = \frac{\partial}{\partial x}\left[(D(h, h_x) - D(\bar{h}, \bar{h}_x))\frac{\partial \bar{h}}{\partial x}\right] + \frac{\partial}{\partial x}\left[D(h, h_x)\frac{\partial \eta}{\partial x}\right].$$

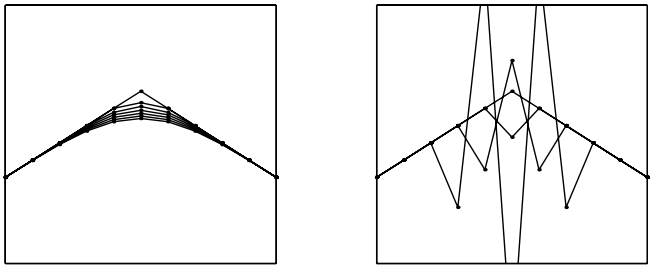

FIGURE 14. A maximum–principle–failing instability.

Assuming a solution h is "close enough" to the steady state in this nonlinear equation is exactly to assume that $D(h, h_x) - D(\bar{h}, \bar{h}_x) \approx 0$. The equation for the deviation then has $B = 0$. In any case, we assume $B = 0$ from now on.

For the rest of this appendix the diffusivity D is supposed to be a nonnegative function of x and t which is bounded by some constant $\bar{D}$. It does not matter if D is known explicitly or if it depends on x, t through a functional dependence on h, h_x as in the ice equation.

I prove a maximum principle for explicit and implicit and semi–implicit cases. For this purpose, let $0 \le \theta \le 1$[8] and discretize (B1) as

$$\begin{aligned}(\text{B6}) \qquad \frac{h_{j,l+1} - h_{j,l}}{\Delta t} &= \frac{1-\theta}{\Delta x^2}\left[D_{j+1/2,l}(h_{j+1,l} - h_{j,l}) - D_{j-1/2,l}(h_{j,l} - h_{j-1,l})\right] \\ &+ \frac{\theta}{\Delta x^2}\left[D_{j+1/2,l+1}(h_{j+1,l+1} - h_{j,l+1}) - D_{j-1/2,l+1}(h_{j,l+1} - h_{j-1,l+1})\right].\end{aligned}$$

The case $\theta = 0$ is the explicit method, $\theta = 1$ is first–order fully implicit, and $\theta = 1/2$ is Crank–Nicolson. Because the method for determining D is not addressed in the current analysis, semi–implicit methods like methods 3, 4 and 7 of section 2 are handled as are implicit methods 5 and 6.

Again it is useful to write out the method as

$$a_0 h_{j,l+1} = a_1 h_{j-1,l+1} + a_2 h_{j+1,l+1} + a_3 h_{j-1,l} + a_4 h_{j,l} + a_5 h_{j+1,l}.$$

Assuming the hypothesis for D, all of the coefficients a_0, a_1, a_2, a_3 and a_5 are nonnegative without any condition on $\Delta t, \Delta x$ or θ. However, $a_4 = 1-(1-\theta)RD_{j+1/2,l}-(1-\theta)RD_{j-1/2,l}$ where $R = \frac{\Delta t}{\Delta x^2}$ as before. Recalling that $\bar{D}$ is a bound on the function D, if

$$(\text{B7}) \qquad 1 - 2(1-\theta)R\bar{D} \ge 0 \quad \text{or} \quad \frac{\Delta t}{\Delta x^2} \le \frac{1}{2(1-\theta)\bar{D}} \quad (0 < \theta \le 0)$$

[8]The "θ–method" is well–known in numerical PDEs. See [17].

then (B6) implies $h_{j,l+1} \le \frac{a_1+a_2+a_3+a_4+a_5}{a_0} \max\{h_{j-1,l+1}, h_{j+1,l+1}, h_{j-1,l}, h_{j,l}, h_{j+1,l}\}$. The reader can check that $a_3 + a_4 + a_5 = 1$ and $a_0 = 1 + a_1 + a_2$. Thus

$$h_{j,l+1} \le \max\{h_{j-1,l+1}, h_{j+1,l+1}, h_{j-1,l}, h_{j,l}, h_{j+1,l}\}.$$

This is a *maximum principle* for the θ–method applied to the ice equation (B1) under condition (B7) and the hypothesis that D is bounded and nonnegative.

This analysis shows (see [17]) that if D is bounded and if condition (B7) holds, relating θ, Δt, Δx and $\bar{D} = \max D_{j,l}$, then (B6) is a stable method. In particular, the first–order fully implicit method, and indeed the linearized versions of methods 3 and 4, are "unconditionally stable" around the steady state. (However, there is a nonlinear effect which appears and results in empirical instability. See [8].) On the other hand, this analysis suggests (B7) as a quantitative stability criterion for the explicit and Crank–Nicolson methods.

The Crank–Nicolson method is unconditionally stable for constant D (this can be shown by Fourier analysis), but does not satisfy an *unconditional* maximum principle. This shows the limitations of maximum principles as tools for predicting instability.

Finally, I return to consider the discretization of (B2), again with $B = 0$ and the θ–method. This reduces to

$$\begin{aligned}(1 + 2R\theta D_{j,l+1})h_{j,l+1} &= R\theta(-\frac{1}{4}D_{j+1,l+1} + \frac{1}{4}D_{j-1,l+1} + D_{j,l+1})h_{j-1,l+1} \\ &+ R\theta(\frac{1}{4}D_{j+1,l+1} - \frac{1}{4}D_{j-1,l+1} + D_{j,l+1})h_{j+1,l+1} \\ &+ R(1-\theta)(-\frac{1}{4}D_{j+1,l} + \frac{1}{4}D_{j-1,l} + D_{j,l})h_{j-1,l} \\ &+ (1 - 2R(1-\theta)D_{j,l})h_{j,l} \\ &+ R(1-\theta)(\frac{1}{4}D_{j+1,l} - \frac{1}{4}D_{j-1,l} + D_{j,l})h_{j+1,l}\end{aligned}$$

or $b_0 h_{j,l+1} = b_1 h_{j-1,l+1} + b_2 h_{j+1,l+1} + b_3 h_{j-1,l} + b_4 h_{j,l} + b_5 h_{j+1,l}$ if written more compactly.

Here $b_0 > 0$ and $b_4 \ge 0$ under the same condition as before, namely (B7). The coefficients b_1, b_2, b_3, b_5 are only nonnegative under a new condition on the variability of D, however. In particular,

(B8) $\qquad (b_3 \ge 0 \text{ and } b_5 \ge 0) \quad$ if and only if $\quad |D_{j+1,l} - D_{j-1,l}| \le 4|D_{j,l}|$

and with a similar condition for the nonnegativity of b_1, b_2.

Such a condition will not hold true at locations where D is small and varies rapidly. But such a situation definitely applies to the ice equation (B1) even in the steady state, as can be seen in appendix A. It can happen either when $h \to 0$ at the edges of the icesheet or at ridges where $h_x \to 0$. Thus this discretization of (B2), and also several other choices, is inappropriate for the ice equation.

Appendix C. Finite differences on not–equally–spaced grids

See [2] section 3.1 for the basics of polynomial interpolation.

Suppose $\{z_k\}$ are grid points with $a \le z_1 < z_2 < \cdots < z_{N+1} \le b$. Suppose approximate derivatives of the function $T : [a, b] \to \mathbb{R}$ are sought at the points z_k using only the values $T_k = T(z_k)$.

Now, the unique mth degree polynomial $P(z)$ which goes through the points $(\zeta_1, T(\zeta_1))$, $(\zeta_2, T(\zeta_2)), \ldots, (\zeta_{m+1}, T(\zeta_{m+1}))$, where $a \le \zeta_1 < \zeta_2 < \cdots < \zeta_{m+1} \le b$, is

$$P(z) = \sum_{i=1}^{m+1} T(\zeta_i) L_i(z).$$

The special polynomials $L_i(z)$ associated to the $\{\zeta_j\}$ were first written down by Lagrange:

$$L_i(z) = \prod_{\substack{j=1 \\ j \ne i}}^{m+1} \frac{z - \zeta_j}{\zeta_i - \zeta_j}.$$

A little thought about the functions $L_i(z)$ is worthwhile because the idea that they are "delta–functions" will come through. That is, $L_i(z_j) = \delta_{ij}$.[9] If T has $m + 1$ derivatives then the error made in using P as an approximation to T is exactly

$$T(z) - P(z) = \frac{T^{(m+1)}(\tilde{\zeta})}{(m+1)!} \prod_{i=1}^{m+1} (z - \zeta_i), \tag{C1}$$

where $\tilde{\zeta}$ is between z and the most distant ζ_i. Since $\tilde{\zeta}$ depends on z in an uncontrolled manner, this formula is only useful in the context of a bound on $T^{(m+1)}$. Furthermore, this error formula can not be usefully differentiated.

The *total* number of grid point is $N + 1$. Second order finite difference approximations to first derivatives require $m = 2$, as one is interested in differentiating quadratic interpolating polynomials at a grid point z_k. In fact, let

$$P_C(z) = T_{k-1} \frac{(z - z_k)(z - z_{k+1})}{(z_{k-1} - z_k)(z_{k-1} - z_{k+1})} + T_k \frac{(z - z_{k-1})(z - z_{k+1})}{(z_k - z_{k-1})(z_k - z_{k+1})} + T_{k+1} \frac{(z - z_{k-1})(z - z_k)}{(z_{k+1} - z_{k-1})(z_{k+1} - z_k)}$$

$$P_L(z) = T_{k-2} \frac{(z - z_{k-1})(z - z_k)}{(z_{k-2} - z_{k-1})(z_{k-2} - z_k)} + T_{k-1} \frac{(z - z_{k-2})(z - z_k)}{(z_{k-1} - z_{k-2})(z_{k-1} - z_k)} + T_k \frac{(z - z_{k-2})(z - z_{k-1})}{(z_k - z_{k-2})(z_k - z_{k-1})}$$

$$P_R(z) = T_k \frac{(z - z_{k+1})(z - z_{k+2})}{(z_k - z_{k+1})(z_k - z_{k+2})} + T_{k+1} \frac{(z - z_k)(z - z_{k+2})}{(z_{k+1} - z_k)(z_{k+1} - z_{k+2})} + T_{k+2} \frac{(z - z_k)(z - z_{k+1})}{(z_{k+2} - z_k)(z_{k+2} - z_{k+1})}.$$

These three quadratic polynomials are *centered*, *left*, and *right* approximations to T around z_k, respectively. See figure 15.

[9] Thus "delta–functions" were discovered by Lagrange and not Kronecker or Dirac!

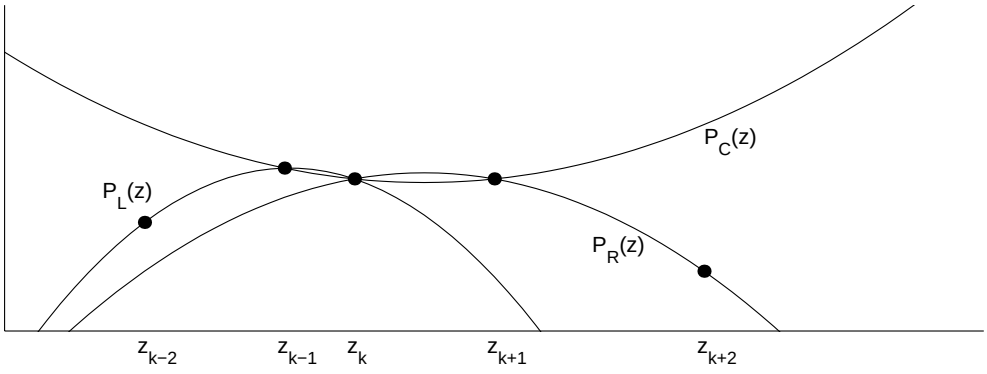

FIGURE 15. Three neighboring quadratic interpolants around z_k.

These polynomials generate approximations of derivatives of $T(z)$ at $z = z_k$. In particular, P_C generates the centered approximation:

(C2) $T'(z_k) \approx P_C'(z_k)$

$$= T_{k-1}\frac{z_k - z_{k+1}}{(z_{k-1}-z_k)(z_{k-1}-z_{k+1})} + T_k\frac{2z_k - z_{k-1} - z_{k+1}}{(z_k - z_{k-1})(z_k - z_{k+1})} + T_{k+1}\frac{z_k - z_{k-1}}{(z_{k+1}-z_{k-1})(z_{k+1}-z_k)}.$$

There is also a "centered" second derivative approximation

(C3) $T''(z_k) \approx P_C''(z_k)$

$$= T_{k-1}\frac{2}{(z_{k-1}-z_k)(z_{k-1}-z_{k+1})} + T_k\frac{2}{(z_k - z_{k-1})(z_k - z_{k+1})} + T_{k+1}\frac{2}{(z_{k+1}-z_{k-1})(z_{k+1}-z_k)}.$$

Note that if the $\{z_k\}$ are equally–spaced with spacing Δz, the above two formulae really are centered, and they reduce to the familiar approximations:

$$T'(z_k) \approx \frac{T_{k+1} - T_{k-1}}{2\Delta z}$$
$$T''(z_k) \approx \frac{T_{k+1} - 2T_k + T_{k-1}}{\Delta z^2}.$$

The polynomials P_L, P_R generate one–sided approximations of $T'(z_k)$ which are useful in upwinding:

(C4) $T'(z_k) \approx P_L'(z_k)$

$$= T_{k-2}\frac{z_k - z_{k-1}}{(z_{k-2}-z_{k-1})(z_{k-2}-z_k)} + T_{k-1}\frac{z_k - z_{k-2}}{(z_{k-1}-z_{k-2})(z_{k-1}-z_k)} + T_k\frac{2z_k - z_{k-2} - z_{k-1}}{(z_k - z_{k-2})(z_k - z_{k-1})}$$

(C5) $T'(z_k) \approx P_R'(z_k)$

$$= T_k\frac{2z_k - z_{k+1} - z_{k+2}}{(z_k - z_{k+1})(z_k - z_{k+2})} + T_{k+1}\frac{z_k - z_{k+2}}{(z_{k+1}-z_k)(z_{k+1}-z_{k+2})} + T_{k+2}\frac{z_k - z_{k+1}}{(z_{k+2}-z_k)(z_{k+2}-z_{k+1})}.$$

Compare to [22] equation (A2). In the equally–spaced case, the formulas reduce to:

$$T'(z_k) \approx \frac{1}{\Delta z}\left(\frac{1}{2}T_{k-2} - 2T_{k-1} + \frac{3}{2}T_k\right)$$
$$T'(z_k) \approx \frac{1}{\Delta z}\left(-\frac{3}{2}T_k + 2T_{k+1} - \frac{1}{2}T_{k+2}\right),$$

which may–or–may–not be familiar but appear in tables [4].

If T has a bounded third derivative then the first derivative approximations above based on P_C, P_L, P_R are second order ($O(\Delta z^2)$ as $\Delta z \to 0$) in the equally–spaced case. This follows because the error formula has a degree three polynomial, in essence, but requires a Taylor series argument because (C1) can not be differentiated. Furthermore, in the equally–spaced case the approximation $T''(z_k) \approx P_C''(z_k)$ is also $O(\Delta x^2)$ if T has a bounded fourth derivative—this follows from another Taylor series argument [2].

If the $\{z_k\}$ are not equally–spaced, let Δ be a bound for the differences $\{z_{k+1} - z_k\}$ in the sense that there is $0 < \lambda < 1$ such that $\lambda\Delta < z_{k+1} - z_k < \Delta$. If T has a bounded third derivative then the first derivative approximations (C2), (C4), (C5) are $O(\Delta^2)$ as $\Delta \to 0$. However, approximation (C3) of the second derivative $T''(z_k)$ is only $O(\Delta)$ and does not benefit from symmetry as in the equally spaced case.

Therefore consider the *cubic* polynomial through the points (z_{k-2}, T_{k-2}), (z_{k-1}, T_{k-1}), (z_k, T_k), (z_{k+1}, T_{k+1}):

$$\tilde{P}(z) = \frac{T_{k-2}(z - z_{k-1})(z - z_k)(z - z_{k+1})}{(z_{k-2} - z_{k-1})(z_{k-2} - z_k)(z_{k-2} - z_{k+1})} + \frac{T_{k-1}(z - z_{k-2})(z - z_k)(z - z_{k+1})}{(z_{k-1} - z_{k-2})(z_{k-1} - z_k)(z_{k-1} - z_{k+1})}$$
$$+ \frac{T_k(z - z_{k-2})(z - z_{k-1})(z - z_{k+1})}{(z_k - z_{k-2})(z_k - z_{k-1})(z_k - z_{k+1})} + \frac{T_{k+1}(z - z_{k-2})(z - z_{k-1})(z - z_k)}{(z_{k+1} - z_{k-2})(z_{k+1} - z_{k-1})(z_{k+1} - z_k)}.$$

It generates an $O(\Delta^2)$ estimate of the second derivative:

(C6)
$$T''(z_k) \approx \tilde{P}''(z_k) = \frac{T_{k-2}(4z_k - 2z_{k-1} - 2z_{k+1})}{(z_{k-2} - z_{k-1})(z_{k-2} - z_k)(z_{k-2} - z_{k+1})} + \frac{T_{k-1}(4z_k - 2z_{k-2} - 2z_{k+1})}{(z_{k-1} - z_{k-2})(z_{k-1} - z_k)(z_{k-1} - z_{k+1})}$$
$$+ \frac{T_k(6z_k - 2z_{k-2} - 2z_{k-1} - 2z_{k+1})}{(z_k - z_{k-2})(z_k - z_{k-1})(z_k - z_{k+1})} + \frac{T_{k+1}(4z_k - 2z_{k-2} - 2z_{k-1})}{(z_{k+1} - z_{k-2})(z_{k+1} - z_{k-1})(z_{k+1} - z_k)}.$$

This approximation is "lop–sided", of course, but it has the accuracy as claimed. It takes advantage of the fact that $\Delta z_k = z_{k+1} - z_k$ increases as k increases in the situation of interest. For the $k = 2$ case it is useful to have the form where the lop–sided–ness is on the right:

(C7)
$$T''(z_k) \approx \frac{T_{k-1}(4z_k - 2z_{k+1} - 2z_{k+2})}{(z_{k-1} - z_k)(z_{k-1} - z_{k+1})(z_{k-1} - z_{k+2})} + \frac{T_k(6z_k - 2z_{k-1} - 2z_{k+1} - 2z_{k+2})}{(z_k - z_{k-1})(z_k - z_{k+1})(z_k - z_{k+2})}$$
$$+ \frac{T_{k+1}(4z_k - 2z_{k-1} - 2z_{k+2})}{(z_{k+1} - z_{k-1})(z_{k+1} - z_k)(z_{k+1} - z_{k+2})} + \frac{T_{k+2}(4z_k - 2z_{k-1} - 2z_{k+1})}{(z_{k+2} - z_{k-1})(z_{k+2} - z_k)(z_{k+2} - z_{k+1})}.$$

Formulae (C4), (C5), (C6), (C7) are used in section 4.

Appendix D. Computations involving the "local diffusivity rate" δ

Let $\delta = 2(\rho g)^n A(T)\alpha^{n-1}(h - z)^n$ as in section 4 (take $f = 1$ here). Let $I = \int_0^z \delta\, d\zeta$ for convenience. Formula (36) for u follows immediately from (2).

Since D is defined (see (1)) by the relation

$$Q = \int_0^h u\, dz = -D\frac{\partial h}{\partial x} + u_b h$$

it follows from (36) that

$$D(x,t) = \int_0^{h(x,t)} \int_0^z \delta(x,\zeta,t)\,d\zeta dz = \int_0^{h(x,t)} \delta(x,z,t)(h(x,t)-z)\,dz.$$

(Change variables using a picture of $\{0 \le z \le h, 0 \le \zeta \le z\}$, for instance.) Letting $J = \int_0^z \delta(z-\zeta)\,d\zeta$, we have equation (39).

The heat source Σ involves $\frac{\partial u}{\partial z} = -\delta\frac{\partial h}{\partial x}$. The fundamental theorem of calculus shows $\Sigma = \frac{g(h-z)}{C_p}\left|\frac{\partial h}{\partial x}\frac{\partial u}{\partial z}\right| = \frac{g(}{C_p}(h-z)\left|\frac{\partial h}{\partial x}\right|^2\delta$, which is (38).

Finally, w needs to be computed from incompressibility (4) and the boundary condition $w_b = 0$ (7). Note this represents a choice. It is also possible to use the "surface kinematical condition"

$$\frac{\partial h}{\partial t} + \frac{\partial h}{\partial x}\cdot u\Big|_{z=h} - w\Big|_{z=h} = B \tag{D1}$$

(see [12]) to provide an upper boundary condition for w. On the other hand, it is natural to check (D1) as a diagnostic if $w_b = 0$ is used as a boundary condition. In any case,

$$w(x,z,t) - 0 = \int_0^z \frac{\partial w}{\partial z}\,d\zeta = \int_0^z -\frac{\partial u}{\partial x}\,d\zeta$$

from (4). Now, $-\frac{\partial u}{\partial x} = \frac{\partial}{\partial x}\left(I\frac{\partial h}{\partial x}\right)$ if $u_b = 0$ so

$$w = \int_0^z \frac{\partial}{\partial x}\left(I\frac{\partial h}{\partial x}\right)\,d\zeta = \frac{\partial}{\partial x}\left(\left[\int_0^z I\,d\zeta\right]\frac{\partial h}{\partial x}\right)$$

and $J = \int_0^z I\,d\zeta$, thus (37).

Appendix M. *Matlab* codes and outputs

`iceconstants.m` code.

```
% ICECONSTANTS  Values for global constants for iceA.m, iceC.m, etc.
%    Also computes analytical solution for steady state.
%    Note n, L, x, xplot must be defined to run this script.

% copy these global declarations into client function
global SperA A B rho g kk Cp G Gam C gCp KT
global L x xplot hexact hexactplot Dexactplot
global A0 Q Rgas cc kappa Tr
global a1 a2 Q1 Q2 RgasPB

SperA=31556926; % year is this many seconds (i.e. 365.2422 days)

% adjustable parameters
A=1e-16/SperA;  %=3.17e-24  1/(Pa^3 s); (EISMINT value) flow law parameter
B=0.3/SperA; %=9.51e-9 m/s; ice accumulation rate

% fixed physical constants
```

```
rho=910; % kg/m^3; density of ice
g=9.81; % m/s^2; accel of gravity
kk=2.10; % J/m K s; thermal conductivity of ice
Cp=2009; % J/kg K; specific heat capacity of ice
G=.042; % J/m^2 s; geothermal heat flux
betaCC=8.7e-4; % K/m; change of melting point with depth (Clausius-Clap.)

% for Hooke81 rule
A0=2.948e-9; % Pa^-3 s^-1;
Q=7.88e4; % J/mol; activation energy for creep
Rgas=8.321; % J/(mol K); gas constant
kappa=1.17;
cc=0.16612; % K^kappa
Tr=273.39; % K; triple point of water

% for Paterson&Budd82 rule
a1=3.615e-13; % Pa^-3 s^-1; T<263
a2=1.733e3; % Pa^-3 s^-1; T>=263
Q1=6.0e4; % J/mol; T<263
Q2=13.9e4; % J/mol; T>=263
RgasPB=8.314; % J/(mol K);

% derived constants
Gam=2*(rho*g)^n*A; % overall constant in continuity eqn
C=B*(n+2)/Gam;
gCp=g/Cp;
KT=kk/(rho*Cp); % temperature eqn diffusion constant

% compute analytical steady state h and D
pow1=1+1/n;  pow2=2+2/n;  twoCpow=2*C^(1/n);
Lxpow=L^pow1-abs(x).^pow1;
hexact=(twoCpow*Lxpow).^(1/pow2);

Lxpowplot=L^pow1-abs(xplot).^pow1;
hexactplot=(twoCpow*Lxpowplot).^(1/pow2);

C1=2^(n/(2*n+2))*C^(1/(2*n+2));
C2=Gam*C1^(2*n+1)/((n+2)*2^(n-1));
pow3=(n+2)/(2*n+2);
Dexactplot=C2*abs(xplot).^(1-1/n).*Lxpowplot.^pow3;
%for n=3 only:  c1=2^(3/8)*(5*B/Gam)^(1/8);
%Dexactplot=(Gam*c1^7/20)*abs(xplot).^(2/3).*(Lxpowplot).^(5/8);
```

`iceA.m` code.

```
function [H0, DD, Rh]=iceA(dtyear, Nx, Mt, type);
% ICEA  [H0, DD, Rh]=iceA(dtyear, Nx, Mt, type)
%    Solves continuity equation decoupled from heat model:
%       h_t = B + (D h_x)_x,    D = Gam |h_x|^(n-1) h^(n+2)
%    Computes five O(dx^2,dt) finite difference methods:
```

```
%       type == 1: explicit with type I diffusion
%       type == 2: explicit with type II diffusion
%       type == 3: semi-implicit with type I diffusion
%       type == 4: semi-implicit with type II diffusion
%       type == 5: s.-impl. with type (.2 II + .8 I)
%       type == 6: s.-impl. with II at interior pts; smear to I at bdry
%    PARAMETERS:
%      dtyear = time step in years
%      Nx = number of horizontal steps (dividing [-L,L])
%      Mt = number of time steps
%             (so dtyear*Mt = T_end; must be multiple of 100)
%      H0 = computed center height at final time
%      DD = computed x=L/2 "diffusion" at final time
%      Rh = coefficient of flow equation difference scheme
%
% Try "dty=10; [H0 DD Rh]=iceA(dty,30,ceil(25000/dty),type)"
%   for comparison to Eismint.
% (ELB 6/27/02)

global SperA A B rho g kk Cp G Gam C gCp KT
global L x xplot hexact hexactplot
n=3;
L=750000; % meters
dx=(2*L)/Nx; x=-L:dx:L; % x grid
xmid=-L+dx/2:dx:L-dx/2; % midpt grid
xplot=linspace(-L,L,400); % for plotting analytical soln
iceconstants;

% constants related to grid
dt=dtyear*SperA;
R0=dt/(dx*dx);  % presumed related to stability for continuity eqn
Tend=dt*Mt; % final time
t=0:dt:Tend;

% use either steady state analytical soln or zero as initial condition
ic=hexact;
%ic=zeros(1,Nx+1);

% allocate space for solutions
hh=zeros(Nx+1,Mt+1); % hh(j,l) with j for x and l for t
D=zeros(Nx+1,1); %column vector
hh(:,1)=ic'; %insert initial condition
%enforce boundary conditions at start
hh(1,:)=0; hh(Nx+1,:)=0;
D(1)=0; D(Nx+1)=0; % see steady bdry

% time-stepping loop
tic
switch type
case 1
   for l=1:Mt
      delh=(hh(2:Nx+1,l)-hh(1:Nx,l))/dx; % Nx by 1 column vector
      hav=(hh(2:Nx+1,l)+hh(1:Nx,l))/2; % Nx by 1 col vect
```

```
      Dmid=(Gam/(n+2))*hav.^(n+2).*abs(delh).^(n-1); % Nx by 1 col vect
      F=Dmid.*(hh(2:Nx+1,l)-hh(1:Nx,l));
      hh(2:Nx,l+1)=hh(2:Nx,l)+B*dt+R0*(F(2:Nx)-F(1:Nx-1));
   end;
case 2
   for l=1:Mt
      delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 col vect
      D(2:Nx)=(Gam/(n+2))*hh(2:Nx,l).^(n+2).*abs(delh).^(n-1);
      Dmid=.5*(D(2:Nx+1)+D(1:Nx)); % D is Nx+1 by 1; Dmid is Nx by 1
      F=Dmid.*(hh(2:Nx+1,l)-hh(1:Nx,l));
      hh(2:Nx,l+1)=hh(2:Nx,l)+B*dt+R0*(F(2:Nx)-F(1:Nx-1));
   end;
case 3
   for l=1:Mt
      delh=(hh(2:Nx+1,l)-hh(1:Nx,l))/dx; % Nx by 1 column vector
      hav=(hh(2:Nx+1,l)+hh(1:Nx,l))/2; % Nx by 1 col vect
      Dmid=(Gam/(n+2))*hav.^(n+2).*abs(delh).^(n-1); % Nx by 1 col vect
      MM=spdiags([-R0*Dmid(2:Nx) (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
            -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
      bb=B*dt+hh(2:Nx,l);
      hh(2:Nx,l+1)=MM bb; % note backslash: tridiagonal solve
   end;
case 4
   for l=1:Mt
      delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 col vect
      D(2:Nx)=(Gam/(n+2))*hh(2:Nx,l).^(n+2).*abs(delh).^(n-1);
      Dmid=.5*(D(2:Nx+1)+D(1:Nx)); % D is Nx+1 by 1; Dmid is Nx by 1
      MM=spdiags([-R0*Dmid(2:Nx) (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
            -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
      bb=B*dt+hh(2:Nx,l);
      hh(2:Nx,l+1)=MM bb; % note backslash: tridiagonal solve
   end;
case 5
   mu=.8;
   for l=1:Mt
      % average: mu * type I + (1-mu) * type II
      delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 col vect
      D(2:Nx)=(Gam/(n+2))*hh(2:Nx,l).^(n+2).*abs(delh).^(n-1);
      DmidII=.5*(D(2:Nx+1)+D(1:Nx)); % D is Nx+1 by 1; Dmid is Nx by 1
      delh=(hh(2:Nx+1,l)-hh(1:Nx,l))/dx; % Nx by 1 column vector
      hav=(hh(2:Nx+1,l)+hh(1:Nx,l))/2; % Nx by 1 col vect
      DmidI=(Gam/(n+2))*hav.^(n+2).*abs(delh).^(n-1);
      Dmid=mu*DmidI+(1-mu)*DmidII; % Nx by 1 col vect
      MM=spdiags([-R0*Dmid(2:Nx) (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
            -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
      bb=B*dt+hh(2:Nx,l);
      hh(2:Nx,l+1)=MM bb; % note backslash: tridiagonal solve
   end;
case 6
   for l=1:Mt
      delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 col vect
      D(2:Nx)=(Gam/(n+2))*hh(2:Nx,l).^(n+2).*abs(delh).^(n-1);
      Dmid=.5*(D(2:Nx+1)+D(1:Nx)); % D is Nx+1 by 1; Dmid is Nx by 1
```

```
      % redo Dmid at ends by type I; never actually uses D(-L)=0 or D(L)=0
      DmidIleft=(Gam/(n+2))*(hh(1:3,l)/2+hh(2:4,l)/2).^(n+2).*...
         abs(hh(2:4,l)/dx-hh(1:3,l)/dx).^(n-1);
      DmidIright=(Gam/(n+2))*(hh(Nx-2:Nx,l)/2+hh(Nx-1:Nx+1,l)/2).^(n+2).*...
         abs(hh(Nx-1:Nx+1,l)/dx-hh(Nx-2:Nx,l)/dx).^(n-1);
      Dmid(1:3)=[1 2/3 1/3]'.*DmidIleft+[0 Dmid(2)/3 2*Dmid(3)/3]';
      Dmid(Nx-2:Nx)=[1/3 2/3 1]'.*DmidIright+[2*Dmid(Nx-2)/3 Dmid(Nx-1)/3 0]';
      MM=spdiags([-R0*Dmid(2:Nx) (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
            -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
      bb=B*dt+hh(2:Nx,l);
      hh(2:Nx,l+1)=MM bb; % note backslash: tridiagonal solve
   end;
case 7
   for l=1:Mt
      delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 col vect
      D(2:Nx)=(Gam/(n+2))*hh(2:Nx,l).^(n+2).*abs(delh).^(n-1);
      Dmid=.5*(D(2:Nx+1)+D(1:Nx)); % D is Nx+1 by 1; Dmid is Nx by 1
      % redo Dmid at ends by type I; never actually uses D(-L)=0 or D(L)=0
      DmidIleft=(Gam/(n+2))*(hh(1:5,l)/2+hh(2:6,l)/2).^(n+2).*...
         abs(hh(2:6,l)/dx-hh(1:5,l)/dx).^(n-1);
      DmidIright=(Gam/(n+2))*(hh(Nx-4:Nx,l)/2+hh(Nx-3:Nx+1,l)/2).^(n+2).*...
         abs(hh(Nx-3:Nx+1,l)/dx-hh(Nx-4:Nx,l)/dx).^(n-1);
      Dmid(1:5)=[1 4/5 3/5 2/5 1/5]'.*DmidIleft+...
         [0 Dmid(2)/5 2*Dmid(3)/5 3*Dmid(4)/5 4*Dmid(5)/5]';
      Dmid(Nx-4:Nx)=[1/5 2/5 3/5 4/5 1]'.*DmidIright+...
         [4*Dmid(Nx-4)/5 3*Dmid(Nx-3)/5 2*Dmid(Nx-2)/5 Dmid(Nx-1)/5 0]';
      MM=spdiags([-R0*Dmid(2:Nx) (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
            -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
      bb=B*dt+hh(2:Nx,l);
      hh(2:Nx,l+1)=MM bb; % note backslash: tridiagonal solve
   end;
otherwise
   error('Unknown type (must be in 1,2,3,4,5,6).');
end;
toc

H0=hh(floor(Nx/2)+1,Mt+1); % center height
DD=Dmid(floor(Nx*.75)+1); % compare to D(L/2)
Rh=max(abs(Dmid))*R0;
if Mt<100, hplot=hh; tplot=t;
else, tt=1:100:Mt+1; tplot=t(tt); hplot=hh(:,tt); end;

% plot of thickness
pos=get(gcf,'Position'); set(gcf,'Position',[pos(1) 100 600 600]);
subplot(3,1,1), plot(x/1000,hh(:,Mt+1),'.',xplot/1000,hexactplot,'r');
ylabel('meters');
typename=strvcat('explicit type I', 'explicit type II',...
   'semi--implicit type I', 'semi--implicit type II',...
   'mu*I+(1-mu)*II','mostly II; smear to I bdry','type 7');
title(strvcat(['ICEA  Solid: Analytical steady solution.  Delta t = '...
      num2str(dtyear) '.'],['Dotted: Thickness h at t=' num2str(Tend/SperA) ...
      ' years computed by ' typename(type,:) ' method.']));
subplot(3,1,2), plot(xplot/1000,Dexactplot,'r'); hold on
```

```
plot(xmid/1000,Dmid,'.');
hold off
ylabel('diffusion D in m^2/s'); xlabel('x in km');
% 3D plot
subplot(3,1,3), mesh(tplot/SperA,x/1000,hplot), view(37.5,30);
ylabel('x in km'); xlabel('t in years'); zlabel('h in meters');
```

iceB.m code.

```
function [H0, DD, Rh]=iceB(dtyear, Nx, Mt)
% ICEB   [H0, DD, Rh]=iceB(dtyear, Nx, Mt)
%
%    Solves flow equation with a semi-implicit
%    finite differences method of order ?.
%    In particular,
%        h_t = (D(t) h_x)_x
%    is approximated by Crank-Nicolson type method
%    but D(t*) is approximated by
%    3/2 D(t_l) - 1/2 D(t_l-1).
% Try " dty=25; [H0 DD Rh]=iceB(dty,30,ceil(25000/dty)) ".
% (Stability limit around  dty=37  for this Nx=30.)
% (ELB 7/1/02)

global SperA A B rho g kk Cp G Gam C gCp KT
global L x xplot hexact hexactplot
n=3;
L=750000; % meters
dx=(2*L)/Nx; x=-L:dx:L; % x grid
xmid=-L+dx/2:dx:L-dx/2; % midpt grid
xplot=linspace(-L,L,400); % for plotting analytical soln
iceconstants;

% constants related to grid
dt=dtyear*SperA;
R0=dt/(dx*dx);  % presumed related to stability for continuity eqn
Tend=dt*Mt; % final time
t=0:dt:Tend;

% use either analytical steady state or zero as initial condition
ic=hexact;
%ic=zeros(1,Nx+1);

% allocate space for solutions
hh=zeros(Nx+1,Mt+1); % hh(j,l) with j for x and l for t
D=zeros(Nx+1,1); %column vector
hh(:,1)=ic'; %insert initial condition
%enforce boundary conditions at start
hh(1,:)=0; hh(Nx+1,:)=0;
D(1)=0; D(Nx+1)=0; % see steady bdry

% first step of lower order
hh(1,2)=0; hh(Nx+1,2)=0;
```

```
delh=(ic(3:Nx+1)-ic(1:Nx-1))'/(2*dx);
Dold(2:Nx,1)=(Gam/(n+2))*hh(2:Nx,1).^(n+2).*abs(delh).^(n-1);
Dold(1)=0; Dold(Nx+1)=0;
Dmid=.5*(Dold(2:Nx+1)+Dold(1:Nx));
MM=spdiags([-R0*Dmid(2:Nx) ...
      (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
      -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
bb=B*dt+ic(2:Nx)';
hh(2:Nx,2)=MM bb; %note backslash: tridiagonal solve
% time-stepping loop
R=.5*R0;
tic
for l=2:Mt
   hh(1,l+1)=0; hh(Nx+1,l+1)=0; % boundary conditions
   delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 column vector
   % D is Nx+1 by 1;
   D(2:Nx,1)=(Gam/(n+2))*hh(2:Nx,l).^(n+2).*abs(delh).^(n-1);
   D(1)=0; D(Nx+1)=0; % see steady bdry
   % Dmid is Nx by 1; extrapolate to t*
   Dmid=.75*(D(2:Nx+1)+D(1:Nx))-.25*(Dold(2:Nx+1)+Dold(1:Nx));
   Dm2=Dmid(1:Nx-1)+Dmid(2:Nx);
   MM=spdiags([-R*Dmid(2:Nx) ...
         (ones(Nx-1,1)+R*Dm2) ...
         -R*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
   bb=B*dt+(ones(Nx-1,1)-R*Dm2).*hh(2:Nx,l) + ...
      R*Dmid(2:Nx).*hh(3:Nx+1,l)+R*Dmid(1:Nx-1).*hh(1:Nx-1,l);
   hh(2:Nx,l+1)=MM bb; %note backslash: tridiagonal solve
   Dold=D;
end;
toc

H0=hh(floor(Nx/2)+1,Mt+1); % center height
Rh=max(abs(D))*R0;
DD=D(floor(Nx*.75)+2);
if Mt<100, hplot=hh; tplot=t;
else, tt=1:100:Mt+1; tplot=t(tt); hplot=hh(:,tt); end;

% plot of thickness
pos=get(gcf,'Position'); set(gcf,'Position',[pos(1) 100 600 600]);
subplot(3,1,1), plot(x/1000,hh(:,Mt+1),'.',xplot/1000,hexactplot,'r'); % profile plot
ylabel('h in meters');
title(strvcat(['ICEB  Solid: Analytical steady solution.  Delta t = '...
      num2str(dtyear) '.'],['Dotted: Thickness h at t=' num2str(Tend/SperA) ...
      ' years by semi-implicit type II extrapolated diffusion.']));
subplot(3,1,2), plot(xplot/1000,Dexactplot,'r'); hold on
plot(x/1000,D,'.'); hold off % diffusion plot
ylabel('D in m^2/s'); xlabel('x in km');
title(['Diffusion at t=' num2str(Tend/SperA) ' years.']);
% 3D plot
subplot(3,1,3), mesh(tplot/SperA,x/1000,hplot), view(37.5,30);
ylabel('x in km'); xlabel('t in years'); zlabel('h in meters');
```

`iceC.m` code (and some outputs).

```
function [H0, Rh, RT]=iceC(dtyear, Nx, Mt, Nz, hmax, plugpos);
% ICEC:   [H0, Rh, RT]=iceC(dtyear, Nx, Mt, Nz, hmax, plugpos)
%
%    Solves continuity equation
%       h_t = B + (D h_x)_x,    D = Gam |h_x|^(n-1) h^(n+2)
%    with a semi-implicit, first order finite differences method.
%    Solves simultaneously for the depth-dependent velocities
%    and temperature in a column at fixed x.  That is, solves
%       T_t + w(z) T_z = KT T_zz + S(z)
%    where w is the vertical velocity *calculated based on
%    a constant value of A*, KT is a diffusion coefficient, and
%       S(z) = (g/Cp) (h-z) |dh/dx du/dz|
%    is a dissipation heat source.
%    PARAMETERS:
%      dtyear = time step in years
%      Nx = number of horizontal steps (dividing [-L,L])
%      Mt = number of time steps
%             (so dtyear*Mt = T_end; must be multiple of 100)
%      Nz = number of vertical steps (in the column)
%      hmax = maximum expected thickness
%      plugpos = position of column as fraction of width
%      H0 = computed center height at final time
%      Rh = coefficient of flow equation difference scheme
%      RT = coefficient of temperature eqn difference scheme
%
% Try "[H0 Rh RT]=iceC(25,30,1000,36,3600,.65)".
% (ELB 7/8/02)

global SperA A B rho g kk Cp G Gam C gCp KT
global L x xplot hexact hexactplot Dexactplot
n=3;
L=750000; dx=(2*L)/Nx; x=-L:dx:L; %x grid based on Nx
xplot=linspace(-L,L,400);
iceconstants;

% constants related to grid
dt=dtyear*SperA;
dz=hmax/Nz;
R0=dt/(dx*dx);  % presumed related to stability for continuity eqn
Tend=dt*Mt; % final time
RT=KT*dt/(dz*dz); % presumed related to stability for temperature eqn
geoheat=-G*dz/kk; % bdry conds: T_z(0) = -G/k when cold
z=0:dz:hmax;
t=0:dt:Tend;
dtomega=log(2);

% use either steady state analytical soln or zero as initial condition
ic=hexact;
%ic=zeros(1,Nx+1);

% mostly build matrix for temperature--in--column model
```

```
J=floor(plugpos*Nx)+1; % index of column
xJ=-L+(J-1)*dx; % = bar x in notes
Ts=239+(8e-8)*abs(xJ);
MT=sparse(Nz,Nz);
MT(2,1)=-RT;
MT(2:Nz,2:Nz)=spdiags([-RT*ones(Nz-1,1) (1+2*RT)*ones(Nz-1,1) -RT*ones(Nz-1,1)],...
   -1:1,Nz-1,Nz-1);

% allocate space for solutions
hh=zeros(Nx+1,Mt+1); % hh(j,l) with j for x and l for t
hh(:,1)=ic'; %insert initial condition
hh(1,1)=0; hh(Nx+1,1)=0; %enforce boundary conditions at start
T=zeros(Nz+1,Mt+1); % space for temperatures
T(:,1)=Ts*ones(Nz+1,1); %initial condition for temperature

% time-stepping loop
tic, for l=1:Mt
   % continuity eqn: semi--implicit type II step
   hh(1,l+1)=0; hh(Nx+1,l+1)=0; % boundary conditions
   delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 column vector
   D(2:Nx,1)=(Gam/(n+2))*hh(2:Nx,l).^(n+2).*abs(delh).^(n-1);
   D(1)=0; D(Nx+1)=0; % see steady bdry
   Dmid=.5*(D(2:Nx+1)+D(1:Nx)); % D is Nx+1 by 1; Dmid is Nx by 1

   MM=spdiags([-R0*Dmid(2:Nx) (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
         -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
   bb=B*dt+hh(2:Nx,l);
   hh(2:Nx,l+1)=MM bb; %note backslash: tridiagonal solve

   % build velocities u (hor) and w (vert) in column; also u_z and Sigma
   h=hh(J,l); alf=delh(J-1); hxx=(1/(dx*dx))*(hh(J-1,l)-2*h+hh(J+1,l));
   h2=h*h; h3=h2*h; h4=h3*h; h5=h4*h; alf2=alf*alf;
   w=-Gam*alf2*(.75*hxx*(h5/5-(h-z).^5/5-h4*z)+alf2*(h4/4-(h-z).^4/4-h3*z));
   dudz=-Gam*alf2*alf*(h-z).^3; source=gCp*(h-z).*abs(alf*dudz);

   % temperature: semi--implicit step
   hoT=Tr-betaCC*h; % homologous temp at base
   if (T(1,l) < hoT) | (T(2,l)-T(1,l) < geoheat)
      MT(1,1:2)=[-1 1]; bT(1,1)=geoheat; % cold case: Neumann cond
   else, MT(1,1:2)=[1 0]; bT(1,1)=hoT; end % hot case: Dirichlet cond
   bT(2:Nz,1)=T(2:Nz,l)+( -(dt/(2*dz))*w(2:Nz)'.*(T(3:Nz+1,l)-T(1:Nz-1,l))+...
      dt*source(2:Nz)' ).*(z(2:Nz)<h)'+dtomega*Ts*(z(2:Nz)>=h)';
   bT(Nz,1)=bT(Nz,1)+RT*Ts;
   T(1:Nz,l+1)=(MT+spdiags(dtomega*(z(1:Nz)>=h)',0,Nz,Nz)) bT;
   %note backslash: tridiagonal solve
   T(Nz+1,l+1)=Ts;
   for k=1:Nz+1
      hoT=Tr-betaCC*(h-z(k)); % homologous temp at depth z
      if T(k,l+1)>hoT, T(k,l+1)=hoT; end % if too hot, set to hoT
   end;
end, toc

H0=hh(floor(Nx/2)+1,Mt+1); % center height
```

```
Rh=max(abs(D))*R0;
if Mt<100, hplot=hh; tplot=t; Tplot=T;
else, tt=1:100:Mt+1; tplot=t(tt); hplot=hh(:,tt); Tplot=T(:,tt); end;

% plot of thickness
figure(1);
pos=get(gcf,'Position'); set(gcf,'Position',[pos(1) 100 600 600]);
subplot(3,1,1), plot(x/1000,hh(:,Mt+1),'.',xplot/1000,hexactplot,'r');
hold on; plot([xJ/1000 xJ/1000],[0 hh(J,Mt+1)]); hold off;
ylabel('meters');
title(['ICEC  Dotted: computed thickness h at t=' num2str(Tend/SperA) ...
      ' years.  Solid: analytical steady solution.']);
subplot(3,1,2), xmid=-L+.5*dx:dx:L-.5*dx;
plot(xmid/1000,Dmid,'.',xplot/1000,Dexactplot,'r'); % diffusion plot
ylabel('m^2/s'); xlabel('x in km');
title(['Diffusion D at t=' num2str(Tend/SperA) ' years.']);
% 3D plot
subplot(3,1,3), mesh(tplot/SperA,x/1000,hplot), view(37.5,30);
ylabel('x in km'); xlabel('t in years'); zlabel('h in meters');

% plot velocities at last time
figure(2);
hindex=ceil(h/dz); zh=0:dz:hindex*dz;
uh=(1/4)*Gam*alf2*alf*((h-zh).^4-h4);
pos=get(gcf,'Position'); set(gcf,'Position',[pos(1) 100 600 600]);
subplot(2,1,1), plot(uh,zh,'r.');
xlabel('m/s'); ylabel('z in meters');
title(['ICEC  Profile of horizontal velocity u at x=' num2str(xJ/1000)...
      ' km and at t=' num2str(Tend/SperA) ' years.']);
subplot(2,1,2), plot(w(1:hindex+1)*100,zh,'b.');
xlabel('m/s'); ylabel('z in meters');
title(['Vertical velocity w at x=' num2str(xJ/1000)...
      ' km and at t=' num2str(Tend/SperA) ' years.']);

% plot heat source and temp at last time
figure(3);
Th=T(1:hindex+1,Mt+1);
pos=get(gcf,'Position'); set(gcf,'Position',[pos(1) 100 600 600]);
subplot(2,1,1), plot(source(1:hindex+1),zh,'.');
xlabel('(deg K)/s'); ylabel('z in meters');
title(['ICEC  Heat source from dissipation Sigma at x=' num2str(xJ/1000)...
      ' km and t=' num2str(Tend/SperA) ' years.']);
subplot(2,1,2), plot(Th'-(Tr-betaCC*(h-zh)),zh,'r.');
xlabel('deg K below melting-point-at-depth'); ylabel('z in meters');
title(['Temperature T at x=' num2str(xJ/1000)...
      ' km and t=' num2str(Tend/SperA) ' years.']);

% 3D plot of T versus t and z
figure(4);
mesh(tplot/SperA,zh,Tplot(1:hindex+1,:)); view(37.5,30);
ylabel('z in m'); xlabel('t in years'); zlabel('deg K');
title(['ICEC  Temperature T at x=' num2str(xJ/1000)...
      ' km.']);
```

The following are output figures of "`iceC(25,30,1000,36,3600,.65)`":

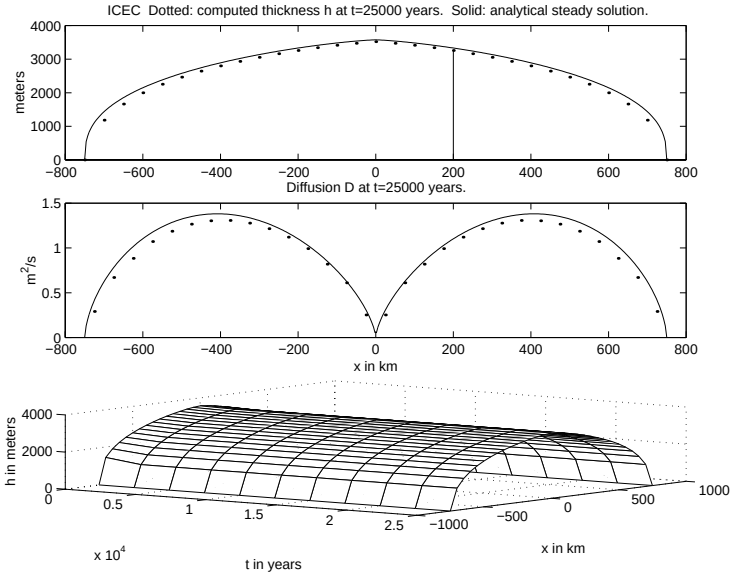

FIGURE 16. `figure(1)` of `iceC.m`

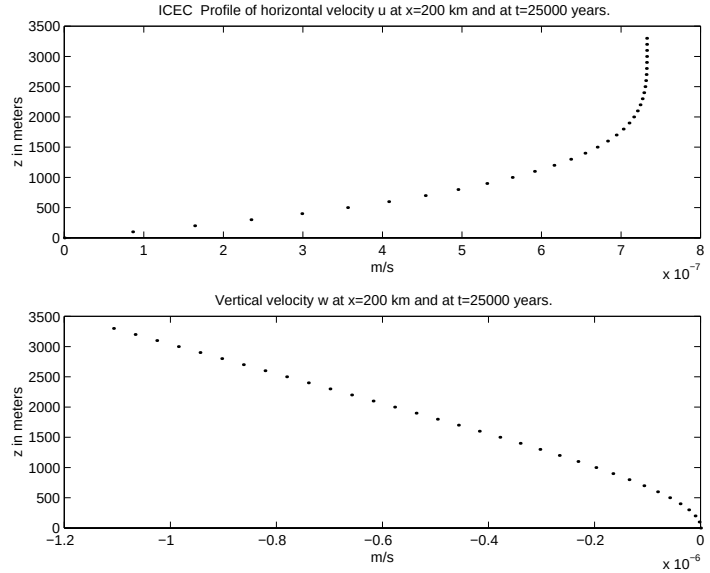

FIGURE 17. `figure(2)` of `iceC.m`

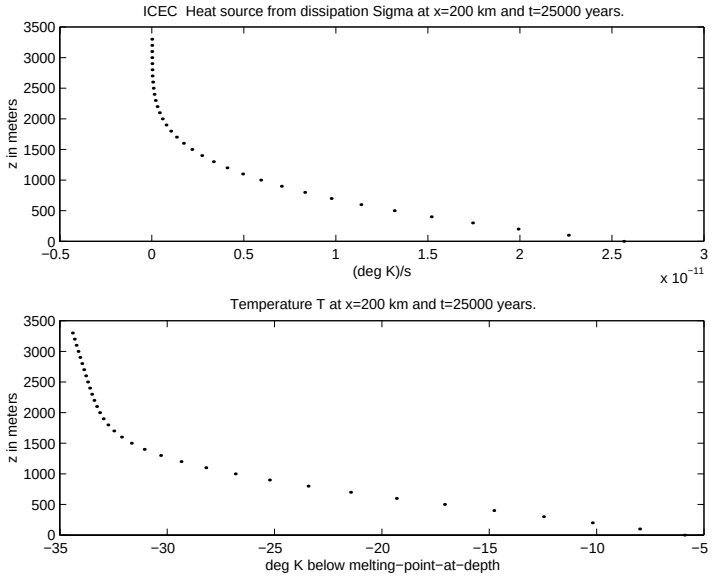

FIGURE 18. `figure(3)` of `iceC.m`

`iceD.m` code.

```
function iceD(dtyear, myNx, Mt, myzk, coupled, flow, n, arrtype)
% ICED   iceD(dtyear, myNx, Mt, myzk, coupled, flow, n, arrtype)
%
%    Solves coupled thermomechanical problem in 2D.
%    Solves continuity equation
%    with a semi-implicit type II method (method 4).
%    Solves simultaneously for the depth-dependent velocities.
%    Solves simultaneously for the depth-dependent temperature,
%    using a ``free-boundary'' model.
%    INPUTS:
%      dtyear = time step in years
%      myNx = number of horizontal steps (dividing [-L,L])
%      Mt = number of time steps
%             (so dtyear*Mt = T_end; must be multiple of 100)
%      myzk = row vector of vertical positions
%             (zk(1)=0; zk(last)=h_max; zk increasing)
%      coupled = 1 or 0
%      flow = flow enhancement factor
%      n = power in Glen flow law (typically 3 or 1.8)
%      arrtype = 'Hooke' or 'PatBud' (see arr.m)
%
% Try "iceD(25,30,10,100*[0 4 8 12 16 20 24 28 32 36 40],0,1,3,'Hooke')"
%    equal spaced;
% Try "zk=4000*[0 .02 .05 .1 .17 .25 .4 .55 .7 .85 1]
%      iceD(25,30,1000,zk,0,1,3,'Hooke')"
%    for Payne & Dongelmans spacing DECOUPLED. [123 sec]
% Try "zk=5000*[0 .02 .05 .1 .17 .25 .4 .55 .7 .85 1]
%      iceD(25,30,1000,zk,1,5,3,'Hooke')"
%    for Payne & Dongelmans spacing COUPLED. [122 sec]
% Try "zk(1:12)=5000*[0 .01 .02 .03 .05 .08 .11 .14 .17 .21 .25 .30];
%      zk(13:23)=5000*[.35 .40 .45 .51 .58 .65 .72 .79 .86 .93 1]
%      iceD(10,60,2500,zk,1,5,3,'Hooke')"
%    for more accuracy.  [19 min]
% (ELB 7/29/02)

global SperA A B rho g kk Cp G C gCp KT
global L dx Nx x xplot hexact hexactplot
global zk Nz dzk hdzk d2zk d3zk dtKT dtomega

% x and t grid info
L=750000; % meters
Nx=myNx; dx=(2*L)/Nx; x=-L:dx:L; % x grid
xmid=-L+dx/2:dx:L-dx/2; % midpt grid
xplot=linspace(-L,L,400); % for plotting analytical soln
iceconstants; % get values of constants
dt=dtyear*SperA;
R0=dt/(dx*dx);  % presumed related to stability for continuity eqn
Tend=dt*Mt; % final time
t=0:dt:Tend;
Gam0=2*(rho*g)^n;
dtomega=log(2);
```

```
dtKT=dt*KT;
geoheat=-G/kk;

% extract z grid info from zk input
zk=myzk; Nz=size(zk,2)-1; hmax=zk(Nz+1);
dzk=zk(2:Nz+1)-zk(1:Nz); hdzk=.5*dzk;
d2zk=zk(3:Nz+1)-zk(1:Nz-1); d3zk=zk(4:Nz+1)-zk(1:Nz-2);
dzkmax=max(dzk); dzkmin=min(dzk);
[zzk xx]=meshgrid(zk,x);
dTdzvect=[(2*zk(1)-zk(2)-zk(3))/(dzk(1)*d2zk(1)), ...
      d2zk(1)/(dzk(1)*dzk(2)), ...
      -dzk(1)/(d2zk(1)*dzk(2))]; % from (C5); at k=1

% initial conditions for h and T
hh=zeros(Nx+1,Mt+1); % hh(j,l) with j for x and l for t
% use either analytical steady state or zero as initial condition
%hh(:,1)=hexact';
hh(:,1)=repmat(0,Nx+1,1);
hh(1,:)=0; hh(Nx+1,:)=0; %enforce boundary conditions at start
T=zeros(Nx+1,Nz+1,Mt+1); % T(j,k,l)
% initial condition for temperature
Ts=239+(8e-8)*(abs(x)/1000).^3; % Eismint version
T(:,:,1)=repmat(Ts',1,Nz+1);

% allocate for other quantities
ddf=zeros(Nx+1,Nz+1); % local diffusivity delta
II=zeros(Nx+1,Nz+1); % integral of ddf
JJ=zeros(Nx+1,Nz+1); % other integral of ddf
u=zeros(Nx+1,Nz+1); % hor vel
w=zeros(Nx+1,Nz+1); % vert vel
Sig=zeros(Nx+1,Nz+1); % dissipation source
alfhnm1=zeros(Nx-1,Nz+1);
hmz=zeros(Nx-1,Nz+1);
JJavdh=zeros(Nx,Nz+1);
Dstar=zeros(Nx+1,1);
Dmid=zeros(Nx,1);

% create temperature matrices [sparse, pentadiagonal, Nz by Nz]
MTN=pentaT; % Neumann case
MTD=MTN; MTD(1,1:3)=[1 0 0]; % in Dirichlet case, T(j,1,l) is known

% time loop
tic
for l=1:Mt
   % h, T given; compute ddf
   delh=(hh(3:Nx+1,l)-hh(1:Nx-1,l))/(2*dx); % Nx-1 by 1 col vect
   alfnm1=repmat(abs(delh).^(n-1),1,Nz+1);
   hmz=repmat(hh(2:Nx,l),1,Nz+1)-zzk(2:Nx,:);
   hmz=hmz.*(hmz>0);
   if coupled, AR=arr(T(2:Nx,:,l),arrtype);
   else, AR=repmat(A,Nx-1,Nz+1); flow=1; end
   ddf(2:Nx,:)=flow*Gam0*AR.*alfnm1.*abs(hmz).^n;
```

```
% compute I, J, D using vintlist
for j=2:Nx
   II(j,:)=vintlist(ddf(j,:));
   JJ(j,:)=zk.*II(j,:)-vintlist(ddf(j,:).*zk);
   Dstar(j)=interp1(zk,JJ(j,:),hh(j,l)); % linearly interpolate
end
Dmid=.5*(Dstar(1:Nx)+Dstar(2:Nx+1)); % 1 by Nx (row) vector

% solve continuity equation for hh(:,l+1)
Mh=spdiags([-R0*Dmid(2:Nx) (ones(Nx-1,1)+R0*(Dmid(1:Nx-1)+Dmid(2:Nx))) ...
      -R0*Dmid(1:Nx-1)], -1:1, Nx-1, Nx-1);
bh=B*dt+hh(2:Nx,l);
hh(2:Nx,l+1)=Mh bh; % note backslash: tridiagonal solve

% stop if heights exceed hmax
if max(hh(:,l+1))>hmax,
   error(['ICED: max height for given zk grid exceeded at l = ' ...
         num2str(l)]), end

% compute u, w, Sig for temperature equation
u(2:Nx,:)=-repmat(delh,1,Nz+1).*II(2:Nx,:);
dh=hh(2:Nx+1,l)-hh(1:Nx,l);
JJavdh=.5*(JJ(1:Nx,:)+JJ(2:Nx+1,:)).*repmat(dh,1,Nz+1);
w(2:Nx,:)=(1/(dx*dx))*(JJavdh(2:Nx,:) - JJavdh(1:Nx-1,:));
Sig(2:Nx,:)=(g/Cp)*repmat(delh.*delh,1,Nz+1).*hmz.*ddf(2:Nx,:);

% solve temp eqn by semi-implicit in each vertical column
for j=2:Nx
   % Build right sides.  Note: T(:,k,l) is an Nx+1 by 1 column vector;
   %    T(j,:,l) is a 1 by Nz+1 row vector.
   h=hh(j,l); ins = (zk<h); outs = 1-ins; % ins=1 if inside ice
   switch j % decide on whether to actually upwind horizontally
   case 2, uhd='R'; case Nx, uhd='L'; otherwise, uhd='u'; end
   % use right side first deriv at k=2 (not upwind)
   bT(2,1)=-dt*ins(2)*(upwindhor(T(:,2,l)',j,u(j,2),uhd)...
      + upwindvert(T(j,:,l),2,w(j,2),'R'));
   for k=3:Nz-1
      bT(k,1)=-dt*ins(k)*(upwindhor(T(:,k,l)',j,u(j,k),uhd)...
         + upwindvert(T(j,:,l),k,w(j,k),'u')); end
   % use left side first deriv at k=Nz (not upwind)
   bT(Nz,1)=-dt*ins(Nz)*(upwindhor(T(:,Nz,l)',j,u(j,Nz),uhd)...
      + upwindvert(T(j,:,l),Nz,w(j,Nz),'L'));
   % add other terms to right side
   bT(2:Nz,1)=bT(2:Nz,1) + T(j,2:Nz,l)' + dt*Sig(j,2:Nz)'.*ins(2:Nz)'+...
      dtomega*Ts(j)*outs(2:Nz)';
   % to REMOVE ADVECTION: bT(2:Nz,1)=T(j,2:Nz,l)'+dtomega*Ts(j)*outs(2:Nz)';
   bT(Nz,1)=bT(Nz,1) ... % add bdry term
      +dtKT*2*(2*zk(Nz)-zk(Nz-2)-zk(Nz-1))/(d3zk(Nz-2)*d2zk(Nz-1)*dzk(Nz))*T(j,Nz+1,l);

   % solve pentadiagonal system for temperatures in column
   dTdz=dTdzvect'*T(j,1:3,l); % vertical derivative of T at base
   hoT=Tr-betaCC*(h-zk); % homologous temp at depth
   if (T(j,1,l) < hoT(1)) | (dTdz < geoheat)
```

```
         bT(1,1)=geoheat; % cold case: Neumann cond
         T(j,1:Nz,l+1)=(MTN + spdiags([0; dtomega*outs(2:Nz)'],0,Nz,Nz))   bT;
         %full(MTN + spdiags([0; dtomega*outs(2:Nz)'],0,Nz,Nz))
      else
         bT(1,1)=hoT(1); % warm case: Dirichlet cond
         T(j,1:Nz,l+1)=(MTD + spdiags([0; dtomega*outs(2:Nz)'],0,Nz,Nz))   bT;
      end
      T(j,:,l+1)=min([hoT; T(j,:,l+1)]); % if too hot, set to hoT
      T(j,Nz+1,l+1)=Ts(j);
      T(1,:,l+1)=repmat(Ts(j),Nz+1,1); T(Nx+1,:,l+1)=T(1,:,l+1);
   end;
end;
toc

hoT=T(:,:,Mt+1)-Tr+betaCC*(repmat(hh(:,Mt+1),1,Nz+1)-zzk); % final homo. temp.
disp(['Central thickness         h(x=0)     = ' num2str(hh(floor(Nx/2)+1,Mt+1))]);
disp(['Diffusivity               D(x=L/2)   = ' num2str(Dmid(floor(Nx*.75)+1))]);
disp(['Homo. basal temp.         T(x=0,z=0) = ' num2str(hoT(floor(Nx*.5)+1,1))]);
disp(['Stability ratio for continuity eqn   Rh = ' num2str(max(abs(Dmid))*R0)]);
disp(['Stability ratio for temperature eqn  RT = ' num2str(dtKT/(dzkmin*dzkmin))]);

if Mt<100, hplot=hh; tplot=t;
else, tt=1:100:Mt+1; tplot=t(tt); hplot=hh(:,tt); end;
% plot of thickness
figure(1), clf, pos=get(gcf,'Position'); set(gcf,'Position',[pos(1) 100 600 600]);
subplot(3,1,1), plot(x/1000,hh(:,Mt+1),'.',xplot/1000,hexactplot,'r'); % profile plot
ylabel('z in m');
title(strvcat(['ICED  Solid: Analytical steady solution.  Delta t = ' ...
      num2str(dtyear) '.   n = ' num2str(n) '.'],...
      ['Dotted: Thickness h at t=' num2str(Tend/SperA)...
      ' years by semi-implicit type II.']));
subplot(3,1,2), plot(xplot/1000,Dexactplot,'r'); hold on
plot(xmid/1000,Dmid,'.'); hold off % diffusion plot
ylabel('D in m^2/s'); xlabel('x in km');
title(['Diffusion at t=' num2str(Tend/SperA) ' years.']);
% 3D plot
subplot(3,1,3), mesh(tplot/SperA,x/1000,hplot), view(37.5,30);
ylabel('x in km'); xlabel('t in years'); zlabel('h in meters');

% contour plot of temperature at last time
figure(2), clf, pos=get(gcf,'Position'); set(gcf,'Position',[pos(1) pos(2) 800 400]);
% interpolate T onto regular grid; display
[zi xi]=meshgrid(0:hmax/(3*Nz):hmax,x);
hoTi = interp2(zzk,xx,hoT,zi,xi,'cubic');
contourf(xi/1000,zi,hoTi), shading flat, hold on
[c hT]=contour(xi/1000,zi,hoTi,'k-'); clabel(c,hT), colorbar
plot(x/1000,hh(:,Mt+1)), plot(xx/1000,zzk,'g.')
jA=ceil(Nx/2); jB=ceil((3/4)*Nx); jC=ceil((19/20)*Nx);
text(x(jA)/1000,zk(1)-200,'A');
text(x(jB)/1000,zk(1)-200,'B');
text(x(jC)/1000,zk(1)-200,'C');
title(['Computed homologous temperature at t=' num2str(Tend/SperA) ' years.']);
xlabel('x in km'); ylabel('z in m');hold off
```

```
% profile plots of temp. at chosen positions
figure(3), clf
plot(hoT(jA,:),zk,hoT(jB,:),zk,hoT(jC,:),zk)
legend('A: x = 0','B: x = L/2','C: x = .9L')
title(['Computed homo. temp. at positions A,B,C shown in figure 2.']);
zA=max(find(zk<hh(jA,Mt+1))); text(hoT(jA,zA),hh(jA,Mt+1),'- h')
zB=max(find(zk<hh(jB,Mt+1))); text(hoT(jB,zB),hh(jB,Mt+1),'- h')
zC=max(find(zk<hh(jC,Mt+1))); text(hoT(jC,zC),hh(jC,Mt+1),'- h')
```

arr.m code.

```
function A=arr(T,type);
% ARR A=arr(T,type)
%    Arrhenius relation.
%     type='Hooke': Hooke81
%     type='PatBud': Paterson&Budd82

% test graph (compare to Payne&Baldwin2000 figure 5):
%    n=3; L=7.5e5; % for iceconstants
%    x=linspace(-L,L,21); xplot=x; iceconstants;
%    T=223:273;
%    AH=SperA*arr(T,'Hooke');
%    APB=SperA*arr(T,'PatBud');
%    semilogy(T,AH,'-+',T,APB,'-o')
%    ylabel('A (Pa^-3 a^-1)')
%    xlabel('Temperature (K)')
%    legend('Hooke','Paterson&Budd',2)

global A0 Q Rgas cc kappa Tr
global a1 a2 Q1 Q2 RgasPB

switch type
case 'Hooke'
   A = A0*exp(-Q./(Rgas*T)+ (3*cc)./(Tr-T).^kappa);
case 'PatBud'
   A = a1*exp(-Q1./(RgasPB*T)).*(T<263) ...
      + a2*exp(-Q2./(RgasPB*T)).*(T>=263);
end %switch
```

pentaT.m code.

```
function A=pentaT;
%  PENTAT  A=pentaT
%    Builds Nz by Nz pentadiagonal temp eqn matrix
%    based on second deriv formulas from cubic approx.
%    Incorporates Neumann cond at k=1.  See appendix C.
%    OUTPUT:
%       A = sparse Nz by Nz matrix
%    To test:
%       global zk Nz dzk d2zk d3zk dtKT
```

```
%       zk=4000*[0 .02 .05 .1 .17 .25 .4 .55 .7 .85 1];
%       Nz=size(zk,2)-1; dzk=zk(2:Nz+1)-zk(1:Nz); d2zk=zk(3:Nz+1)-zk(1:Nz-1);
%       d3zk=zk(4:Nz+1)-zk(1:Nz-2); dt=10; dtKT=dt*31556926*2.10/(910*2009);
%       P=pentaT; spy(P), full(P)

global zk Nz dzk d2zk d3zk dtKT
A=sparse(Nz,Nz);

% use (C5) for k=1
A(1,1:3)=[(2*zk(1)-zk(2)-zk(3))/(dzk(1)*d2zk(1)), ...
             d2zk(1)/(dzk(1)*dzk(2)), ...
             -dzk(1)/(d2zk(1)*dzk(2))];
k=2; % use (C7)
A(2,1:4)=[0 1 0 0] ...
   - dtKT*[-2*(2*zk(k)-zk(k+1)-zk(k+2))/(dzk(k-1)*d2zk(k-1)*d3zk(k-1)), ...
      2*(3*zk(k)-zk(k-1)-zk(k+1)-zk(k+2))/(dzk(k-1)*dzk(k)*d2zk(k)), ...
      -2*(2*zk(k)-zk(k-1)-zk(k+2))/(d2zk(k-1)*dzk(k)*dzk(k+1)), ...
      2*(2*zk(k)-zk(k-1)-zk(k+1))/(d3zk(k-1)*d2zk(k)*dzk(k+1))];
% use (C6)
for k=3:Nz-1
   A(k,k-2:k+1)= [0 0 1 0] ...
     - dtKT*[-2*(2*zk(k)-zk(k-1)-zk(k+1))/(dzk(k-2)*d2zk(k-2)*d3zk(k-2)), ...
             2*(2*zk(k)-zk(k-2)-zk(k+1))/(dzk(k-2)*dzk(k-1)*d2zk(k-1)), ...
             -2*(3*zk(k)-zk(k-2)-zk(k-1)-zk(k+1))/(d2zk(k-2)*dzk(k-1)*dzk(k)), ...
             2*(2*zk(k)-zk(k-2)-zk(k-1))/(d3zk(k-2)*d2zk(k-1)*dzk(k))];
end
k=Nz; % use (C6)
A(Nz,Nz-2:Nz)= [0 0 1] ...
     - dtKT*[-2*(2*zk(k)-zk(k-1)-zk(k+1))/(dzk(k-2)*d2zk(k-2)*d3zk(k-2)), ...
             2*(2*zk(k)-zk(k-2)-zk(k+1))/(dzk(k-2)*dzk(k-1)*d2zk(k-1)), ...
             -2*(3*zk(k)-zk(k-2)-zk(k-1)-zk(k+1))/(d2zk(k-2)*dzk(k-1)*dzk(k))];
```

upwindhor.m code.

```
function y = upwindhor(ff,j,lam,forceLR)
% UPWINDHOR  y = upwindhor(ff,j,lam,forceLR)
%    Finds first derivative of ff according to upwinding on the value lam:
%                 / (left-side equal-spaced 3 pt formula)    if  lam >= 0
%      y = lam * |
%                  (right-side equal-spaced 3 pt formula)   if  lam < 0
%    See appendix C.

% to test:
%    global Nx dx
%    Nx=10; dx=.3; xj=0:dx:3;
%    f=.5*xj.^2                % note  (.5 x^2)' = x
%    upwindhor(f,4,1,'u')      % = xj(4) = .9
%    upwindhor(f,1,1,'R')      % = xj(1) = 0

global Nx dx
```

```
switch forceLR
case 'u' % actually upwind
   if lam>=0
      if j>=3
         y=.5*ff(j-2)-2*ff(j-1)+1.5*ff(j);
      else, error('UPWINDHOR: index out of range for lambda nonnegative')
      end;
   else
      if j<=Nx-1
         y=-1.5*ff(j)+2*ff(j+1)-.5*ff(j+2);
      else, error('UPWINDHOR: index out of range for lambda negative')
      end;
   end
case 'L' % forced use of left-side approx
   y=.5*ff(j-2)-2*ff(j-1)+1.5*ff(j);
case 'R' % forced use of right-side approx
   y=-1.5*ff(j)+2*ff(j+1)-.5*ff(j+2);
otherwise
   error('UPWINDHOR: forcing option not allowed')
end
y=(lam/dx)*y;
```

upwindvert.m code.

```
function y = upwindvert(ff,k,lam,forceLR)
% UPWINDVERT  y = upwindvert(ff,k,lam,forceLR)
%    Finds first derivative of ff according to upwinding on the value lam:
%                / (left-side 3 pt formula)    if  lam >= 0
%      y = lam * |
%                 (right-side 3 pt formula)   if  lam < 0
%    See appendix C.

% to test:
%    global Nz zk dzk d2zk, Nz=15,
%    zk=[0:.05:.5 .6:.1:1]
%    dzk=zk(2:Nz+1)-zk(1:Nz), d2zk=zk(3:Nz+1)-zk(1:Nz-1)
%    f=.5*zk.^2                              % note  (.5 z^2)' = z
%    upwindvert(f,4,1,'u')                   % = zk(4) = .15

global Nz zk dzk d2zk

switch forceLR
case 'u'
   if lam>=0
      if k>=3 % use left-sided approx (C4)
         y=ff(k-2)*dzk(k-1)/(dzk(k-2)*d2zk(k-2)) - ...
            ff(k-1)*d2zk(k-2)/(dzk(k-2)*dzk(k-1)) + ...
            ff(k)*(2*zk(k)-zk(k-2)-zk(k-1))/(d2zk(k-2)*dzk(k-1));
      else, error('UPWINDVERT: index out of range for lambda nonnegative')
      end;
   else
      if k<=Nz-1 % use right-sided approx (C5)
```

```
            y=ff(k)*(2*zk(k)-zk(k+1)-zk(k+2))/(dzk(k)*d2zk(k)) + ...
               ff(k+1)*d2zk(k)/(dzk(k)*dzk(k+1)) - ...
               ff(k+2)*dzk(k)/(d2zk(k)*dzk(k+1));
         else, error('UPWINDVERT: index out of range for lambda negative')
         end
      end
   case 'L' % forced use of left-side approx
      y=ff(k-2)*dzk(k-1)/(dzk(k-2)*d2zk(k-2)) - ...
         ff(k-1)*d2zk(k-2)/(dzk(k-2)*dzk(k-1)) + ...
         ff(k)*(2*zk(k)-zk(k-2)-zk(k-1))/(d2zk(k-2)*dzk(k-1));
   case 'R' % forced use of right-side approx
      y=ff(k)*(2*zk(k)-zk(k+1)-zk(k+2))/(dzk(k)*d2zk(k)) + ...
         ff(k+1)*d2zk(k)/(dzk(k)*dzk(k+1)) - ...
         ff(k+2)*dzk(k)/(d2zk(k)*dzk(k+1));
   otherwise
      error('UPWINDVERT: forcing option not allowed')
   end
   y=lam*y;
```

vintlist.m code.

```
function Jlist = vintlist(ff)
% VINTLIST  Jlist = vintlist(ff)
%   Computes integral of  ff  at points  zk by trapezoid rule:
%      Jlist(k) = int_0^zk(k) ff(zeta) d zeta;   k = 1,...,Nz+1

% to test:
%    global Nz zk hdzk, Nz=10,
%    zk=[0 .02 .05 .10 .17 .25 .40 .55 .7 .85 1]
%    % compare: zk=[0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1]
%    hdzk=.5*(zk(2:Nz+1)-zk(1:Nz))
%    f=(3/2)*zk.^(1/2) % integral is zk.^(3/2)
%    plot(zk,f,'.',zk,vintlist(f),'o',zk,zk.^(3/2))
%    legend('f','vintlist(f)','exact integral of f'), grid on
%    title(['max. error ' num2str(max(abs(vintlist(f)-zk.^(3/2))))])

global Nz hdzk
% hdzk(k) = .5 (zk(k+1)-zk(k));  k = 1,...,Nz

Jlist = cumsum([0 hdzk.*(ff(1:Nz)+ff(2:Nz+1))]);
```