

Numerical modelling of glaciers, ice sheets, and ice shelves

Ed Bueler

University of Alaska, Fairbanks
elbueler@alaska.edu

McCarthy, Alaska, June 2016

slogans & scope

slogans: I will

- ▶ focus on approximating ice flow
- ▶ provide numerical codes that actually work
- ▶ always care about the continuum model

scope: I will cover these

- ▶ models
 - shallow ice approximation (SIA) in 2D
 - shallow shelf approximation (SSA) in 1D
 - mass continuity & surface kinematical equations
- ▶ numerical ideas
 - finite difference schemes
 - solving algebraic systems from stress balances
 - verification

notation

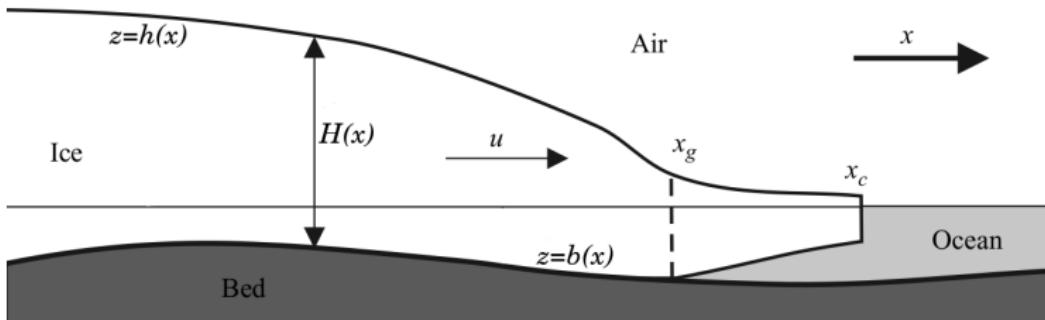


figure modified from Schoof (2007)

- ▶ coordinates t, x, y, z (with z vertical, positive upward)
- ▶ subscripts for partial derivatives $u_x = \partial u / \partial x$
- ▶ H = ice thickness
- ▶ h = ice surface elevation
- ▶ b = bedrock surface elevation
- ▶ T = temperature
- ▶ $\mathbf{u} = (u, v, w) =$ ice velocity
- ▶ ρ = density of ice
- ▶ ρ_w = density of ocean water
- ▶ g = acceleration of gravity
- ▶ $n = 3$ Glen flow law exponent = 3
- ▶ $A = A(T) =$ ice softness in Glen law ($\mathbf{D}_{ij} = A(T)\tau^{n-1}\tau_{ij}$)
- ▶ **please ask about notation!**

Matlab/Octave codes

- ▶ lectures and notes are structured around 18 ice flow codes
- ▶ several codes will appear in these lectures, but not all
- ▶ each is ~ 1/2 page of Matlab/Octave code
- ▶ please give them a try!
 - .zip and .tar.gz forms available from memory stick
 - and online:

<https://github.com/bueler/mccarthy>

Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

shelves and streams

ice in glaciers is treated a *fluid*

- ▶ what's a fluid?

ice in glaciers is treated a *fluid*

- ▶ what's a fluid?
- ▶ at minimum, we describe fluids by
 - a *density field* $\rho(t, x, y, z)$
 - a vector *velocity field* $\mathbf{u}(t, x, y, z)$

ice in glaciers is a *fluid* 2

- ▶ if the ice were
 - faster-moving than it actually is, and
 - linearly-viscous like liquid water
- then it would be a “typical” fluid

- ▶ for some typical fluids one uses the Navier-Stokes equations as the model:

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

$$\rho (\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau}_{ij} + \rho \mathbf{g} \qquad \textit{force balance}$$

$$2\nu \mathbf{D}_{ij} = \boldsymbol{\tau}_{ij} \qquad \qquad \qquad \textit{flow law}$$

- force balance equation is “ $ma = F$ ”

hmmm . . . does not sound like glaciology to me!

- ▶ **yes**, numerical ice sheet flow modelling is “computational fluid dynamics”
 - it’s large-scale like atmosphere and ocean
 - . . . but it is a weird one
- ▶ consider what makes atmosphere/ocean flow exciting:
 - turbulence
 - convection
 - coriolis force
 - density/salinity stratification
 - chemistry (methane, ozone, . . .)
- ▶ none of the above list is relevant to ice flow
- ▶ what could be interesting about the flow of slow, cold, stiff, laminar, inert old ice?
 - it’s *ice dynamics!*

ice is a slow, shear-thinning fluid

- ▶ ice fluid is

$$\text{slow: } \rho(\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u}) \approx 0$$

non-Newtonian: viscosity ν is not constant

- ▶ “slow”:

$$\rho(\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u}) \approx 0 \quad \iff \quad \begin{pmatrix} \text{forces of inertia} \\ \text{are neglected} \end{pmatrix}$$

- ▶ ice is non-Newtonian in a “shear-thinning” way
 - higher strain rates means lower viscosity
- ▶ so the standard “full” model is Glen-law ($n = 3$) **Stokes**:

$$\nabla \cdot \mathbf{u} = 0$$

incompressibility

$$0 = -\nabla p + \nabla \cdot \boldsymbol{\tau}_{ij} + \rho \mathbf{g}$$

force balance

$$\mathbf{D}_{ij} = A\tau^2 \boldsymbol{\tau}_{ij}$$

flow law

“slow” means no memory of velocity (i.e. momentum)

- ▶ note *no time derivatives* in Stokes model:

$$\nabla \cdot \mathbf{u} = 0$$

$$0 = -\nabla p + \nabla \cdot \boldsymbol{\tau}_{ij} + \rho \mathbf{g}$$

$$\mathbf{D}_{ij} = A\tau^2 \boldsymbol{\tau}_{ij}$$

- ▶ thus a time-stepping ice sheet code
 - recomputes the full velocity field at every time step, and
 - does not require velocity from the previous time step¹
- ▶ because there is no memory of previous velocity, velocity is a “diagnostic” output, not needed for (e.g.) restarting the model

¹to paraphrase (?) Bob Dylan: to be a weather forecaster you've got to know which way the wind blows ... but don't expect that from a glaciologist

plane flow Stokes

- ▶ recall the Stokes model:

$$\nabla \cdot \mathbf{u} = 0$$

incompressibility

$$0 = -\nabla p + \nabla \cdot \boldsymbol{\tau}_{ij} + \rho \mathbf{g}$$

force balance

$$\mathbf{D}_{ij} = A\tau^2 \boldsymbol{\tau}_{ij}$$

flow law

- ▶ suppose we work in a x, z plane, such as the centerline of a glacier, or in a cross-flow plane
- ▶ notation on next several slides:
 - x, z as subscripts denote partial derivatives
 - τ_{13} is the “vertical” shear stress
 - τ_{11} and $\tau_{33} = -\tau_{11}$ are (deviatoric) longitudinal stresses

plane flow Stokes 2

- ▶ in the x, z plane flow case the Stokes equations say

$$u_x + w_z = 0$$

incompressibility

$$p_x = \tau_{11,x} + \tau_{13,z}$$

stress balance (x)

$$p_z = \tau_{13,x} - \tau_{11,z} - \rho g$$

stress balance (z)

$$u_x = A\tau^2 \tau_{11}$$

flow law (diagonal)

$$u_z + w_x = 2A\tau^2 \tau_{13}$$

flow law (off-diagonal)

- ▶ we have five equations in five unknowns ($u, w, p, \tau_{11}, \tau_{13}$)
- ▶ this is complicated enough ... what about in a simplified situation?

slab-on-a-slope

- ▶ suppose: constant thickness, tilted bedrock
- ▶ rotated coordinates:

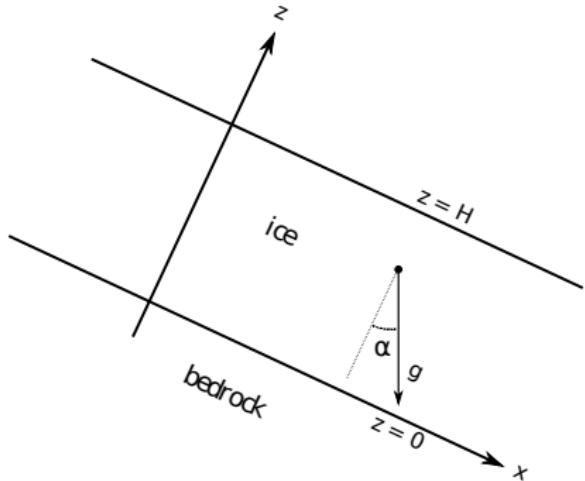
$$\mathbf{g} = g \sin \theta \hat{x} - g \cos \theta \hat{z}$$

- ▶ so p_x, p_z equations are now:

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

$$p_z = \tau_{13,x} - \tau_{11,z} - \rho g \cos \theta$$

- ▶ for this **slab-on-a-slope** there is *no variation in x*
- ▶ the equations simplify:



$$w_z = 0$$

$$0 = \tau_{11}$$

$$\tau_{13,z} = -\rho g \sin \theta$$

$$u_z = 2A\tau^2 \tau_{13}$$

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

slab-on-a-slope 2

- ▶ add some boundary conditions:

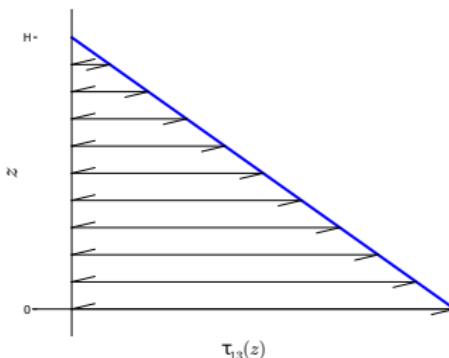
$$w(\text{base}) = 0, \quad p(\text{surface}) = 0, \quad u(\text{base}) = u_0$$

- ▶ by integrating vertically, get:

$$w = 0, \quad p = \rho g \cos \theta (H - z), \quad \tau_{13} = \rho g \sin \theta (H - z)$$

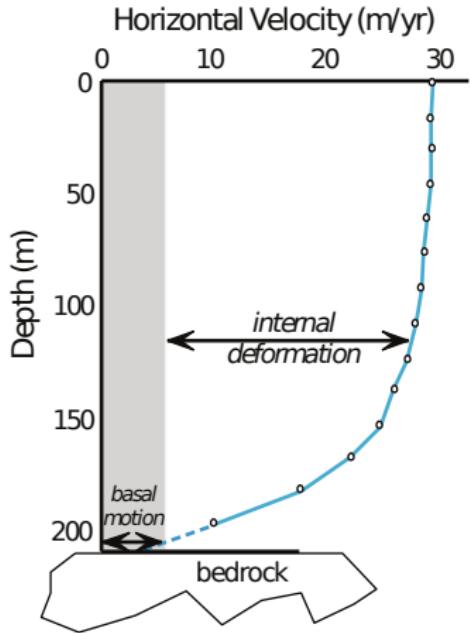
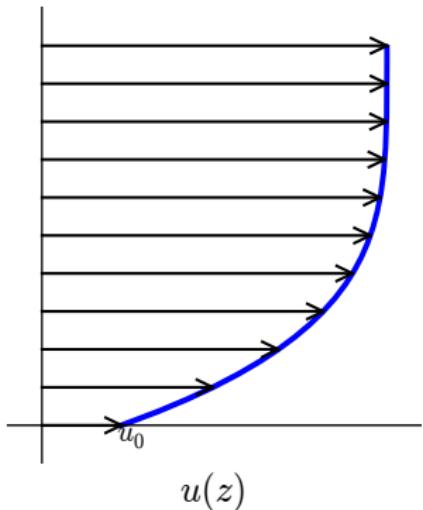
- ▶ and from " $u_z = 2A\tau^2\tau_{13}$ " get **velocity formula**

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



slab-on-a-slope 3

- ▶ do we believe these equations?
- ▶ velocity formula on last slide gives figure below
- ▶ compare to observations at right



Velocity profile of the Athabasca Glacier, Canada, derived from inclinometry (Savage and Paterson, 1963)

mass conservation

- ▶ Q: having computed the velocity $u = u(t, x, z) \dots$ so what?

mass conservation

- ▶ Q: having computed the velocity $u = u(t, x, z) \dots$ so what?
- ▶ A: velocity moves the ice around ... we want to predict where it goes

- ▶ here we start to take a “glacier-suitable” view of mass conservation
- ▶ suppose, instead of slab-on-a-slope, that our ice flow has variable thickness $H(t, x)$
- ▶ compute the vertical average of velocity:

$$\bar{u}(t, x) = \frac{1}{H} \int_0^H u(t, x, z) dz$$

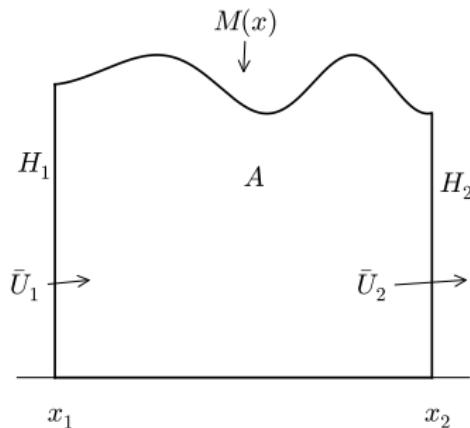
mass conservation 2

- ▶ $M(x)$ is climatic(-basal) mass balance at x
- ▶ consider change of area in the figure:

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

- ▶ assume width $dx = x_2 - x_1$ is small so $A \approx dx H$
- ▶ divide eqn * by dx and get

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



- ▶ this is a *mass continuity equation*
- ▶ area in 2D becomes volume in 3D

combine equations so far

- ▶ from slab-on-slope velocity formula in $u_0 = 0$ case,

$$\begin{aligned}\bar{u}H &= \int_0^H \frac{1}{2}A(\rho g \sin \theta)^3 (H^4 - (H-z)^4) \ dz \\ &= \frac{2}{5}A(\rho g \sin \theta)^3 H^5\end{aligned}$$

- ▶ note $\sin \theta \approx \tan \theta = -h_x$
- ▶ combine with mass continuity $H_t = M - (\bar{u}H)_x$ to get:

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

- ▶ this is a rough explanation of “shallow ice approximation” (SIA)
... next

Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

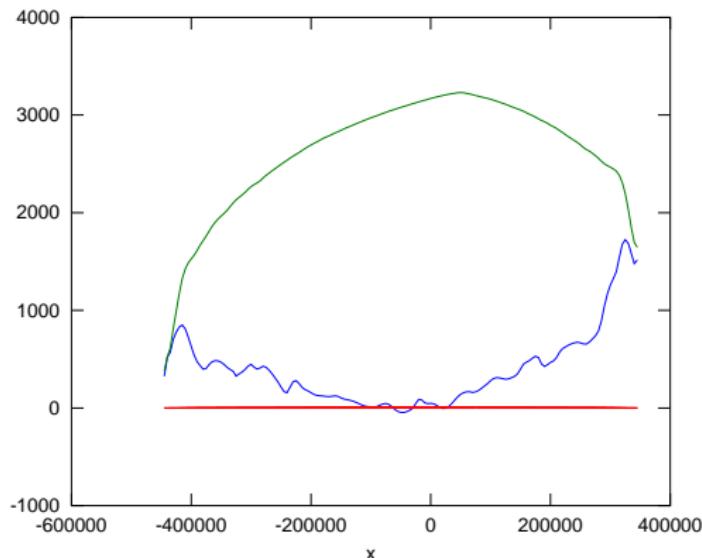
shelves and streams

slow, non-Newtonian, shallow, and sliding

- ▶ ice sheets have four outstanding properties *as fluids*:
 1. slow
 2. non-Newtonian
 3. shallow (usually)
 4. contact slip (sometimes)

regarding “shallow”

- ▶ below in red is a no-vertical-exaggeration cross section of Greenland at 71°
- ▶ green and blue: standard vertically-exaggerated cross section



flow model I: non-sliding, isothermal shallow ice approximation = (SIA)

a model which applies to

- ▶ small depth-to-width ratio (“shallow”) grounded ice sheets
- ▶ on not-too-rough bed topography,
- ▶ whose flow is *not* dominated by sliding and/or liquid water at the base or margin



“Polaris Glacier,” northwest Greenland, photo 122, Post & LaChapelle (2000)

SIA model equations

- ▶ though the best explanation of the SIA is to use shallowness to simplify the Stokes equations, here we take the simple slogan:

the SIA uses the formulas from slab-on-a-slope

- ▶ shear stress approximation:

$$(\tau_{13}, \tau_{23}) = -\rho g(h - z)\nabla h$$

- ▶ let $\mathbf{u} = (u, v)$ be the horizontal velocity
- ▶ use this approximation:

$$\begin{aligned}\mathbf{u}_z &= 2A|(\tau_{13}, \tau_{23})|^{n-1}(\tau_{13}, \tau_{23}) \\ &= -2A(\rho g)^n(h - z)^n|\nabla h|^{n-1}\nabla h\end{aligned}$$

- ▶ by integrating vertically, in the non-sliding case,

$$\mathbf{u} = -\frac{2A(\rho g)^n}{n+1} [H^{n+1} - (h - z)^{n+1}] |\nabla h|^{n-1} \nabla h$$

- ▶ but mass continuity remains, $H_t = M - (\bar{\mathbf{u}} H)_x$

SIA thickness equation

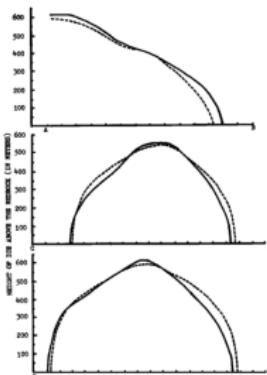
- ▶ from last slide, we get the non-sliding, isothermal shallow ice approximation for how thickness changes:

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

- where H is ice thickness, h is ice surface elevation, b is bed elevation ($h = H + b$)
 - M combines surface and basal mass (im)balance: accumulation if $M > 0$, ablation if $M < 0$
 - n is the exponent in the Glen flow law
 - $\Gamma = 2A(\rho g)^n / (n+2)$ is a positive constant
- ▶ numerically solve (1) and you've got a usable model for . . . *the Barnes ice cap* (Mahaffy, 1976)

good questions:

1. where does equation (1) come from?
2. how to solve it numerically?
3. how to *think* about it?



heat equation

- ▶ to understand SIA, see heat equation
- ▶ recall Newton's law of cooling

$$\frac{dT}{dt} = -K(T - T_{\text{ambient}})$$

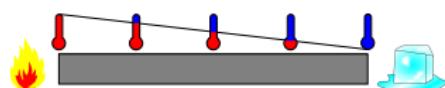
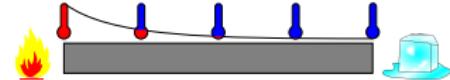
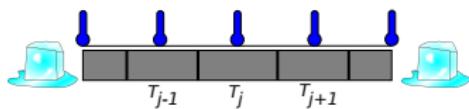
where T is object temperature and K relates to material and geometry of object (e.g. cup of coffee)

- ▶ Newton's law for segments of a rod:

$$\begin{aligned}\frac{dT_j}{dt} &= -K \left(T_j - \frac{1}{2}(T_{j-1} + T_{j+1}) \right) \\ &= \frac{K}{2} (T_{j-1} - 2T_j + T_{j+1})\end{aligned}$$

- ▶ this has limit as segments shrink:

$$T_t = DT_{xx}$$



analogy: SIA versus 2D heat equation

- ▶ general heat eqn: $T_t = F + \nabla \cdot (D \nabla T)$
- ▶ side-by-side comparison:

SIA for ice thickness $H(t, x, y)$

heat eqn for temperature $T(t, x, y)$

$$H_t = M + \nabla \cdot (\Gamma H^{n+2} |\nabla h|^{n-1} \nabla h) \quad T_t = F + \nabla \cdot (D \nabla T)$$

- ▶ identify the diffusivity in the SIA:

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

- ▶ non-sliding shallow ice flow *diffuses* the ice sheet
- ▶ “issues” with this analogy:
 - D depends on solution $H(t, x, y)$
 - $D \rightarrow 0$ at margin, where $H \rightarrow 0$
 - $D \rightarrow 0$ at divides/domes, where $|\nabla h| \rightarrow 0$

numerics for heat equation: basic ideas of finite differences

- ▶ numerical schemes for heat equation are good start for SIA
- ▶ for differentiable $f(x)$ and any h , *Taylor's theorem* says

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

- ▶ you can replace " h " by multiples of Δx , e.g.:

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

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

- ▶ *main idea*: combine expressions like these to give approximations of derivatives, from values on a grid

finite differences for partial derivatives

- ▶ we want partial derivative expressions, for example with any function $u = u(t, x)$:

$$u_t(t, x) = \frac{u(t + \Delta t, x) - u(t, x)}{\Delta t} + O(\Delta t),$$

$$u_t(t, x) = \frac{u(t + \Delta t, x) - u(t - \Delta t, x)}{2\Delta t} + O(\Delta t^2),$$

$$u_x(t, x) = \frac{u(t, x + \Delta x) - u(t, x - \Delta x)}{2\Delta x} + O(\Delta x^2),$$

$$u_{xx}(t, x) = \frac{u(t, x + \Delta x) - 2u(t, x) + u(t, x - \Delta x)}{\Delta x^2} + O(\Delta x^2)$$

and so on

- ▶ sometimes we want a derivative in-between grid points:

$$u_x(t, x + (\Delta x/2)) = \frac{u(t, x + \Delta x) - u(t, x)}{\Delta x} + O(\Delta x^2)$$

- ▶ “ $+O(h^2)$ ” is better than “ $+O(h)$ ” if h is a small number

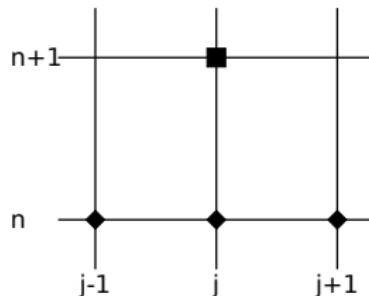
explicit scheme for heat equation

- ▶ consider 1D heat equation $T_t = DT_{xx}$
- ▶ approximately true:

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

- ▶ the difference between the equation $T_t = DT_{xx}$ and the scheme is $O(\Delta t, \Delta x^2)$ (Morton and Mayers, 2005)
- ▶ notation: (t_n, x_j) is a point in the time-space grid
- ▶ notation: $T_j^n \approx T(t_n, x_j)$
- ▶ let $\nu = D\Delta t / (\Delta x)^2$, so “explicit” scheme is

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



- ▶ “stencil” at right →

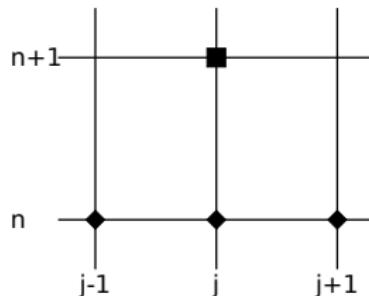
explicit scheme for heat equation

- ▶ consider 1D heat equation $T_t = DT_{xx}$
- ▶ approximately true:

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

- ▶ the difference between the equation $T_t = DT_{xx}$ and the scheme is $O(\Delta t, \Delta x^2)$ (Morton and Mayers, 2005)
- ▶ notation: (t_n, x_j) is a point in the time-space grid
- ▶ notation: $T_j^n \approx T(t_n, x_j)$ ← can you distinguish these?
- ▶ let $\nu = D\Delta t / (\Delta x)^2$, so “explicit” scheme is

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



- ▶ “stencil” at right →

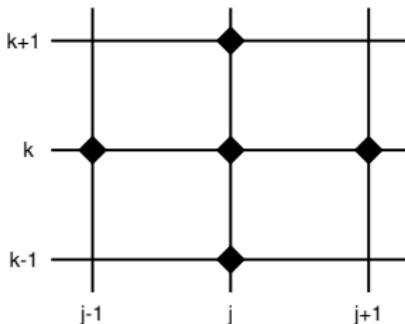
explicit scheme in 2D

- ▶ recall heat equation two spatial variables (2D):

$$T_t = D(T_{xx} + T_{yy})$$

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

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



implementation

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

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

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

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

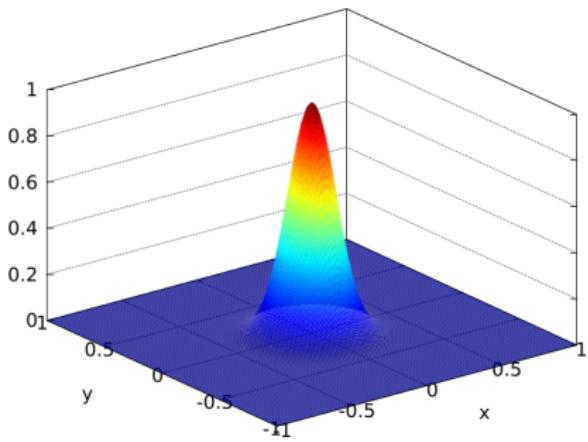
heat.m

- ▶ solves $T_t = D(T_{xx} + T_{yy})$ on square $-1 < x < 1, -1 < y < 1$
- ▶ example uses initial condition $T_0(x, y) = e^{-30r^2}$
- ▶ code uses “colon notation” to remove loops (over space)
- ▶ » `heat(1.0, 30, 30, 0.001, 20)`
approximates T on 30×30 spatial grid, with $D = 1$ and $N = 20$ steps of $\Delta t = 0.001$

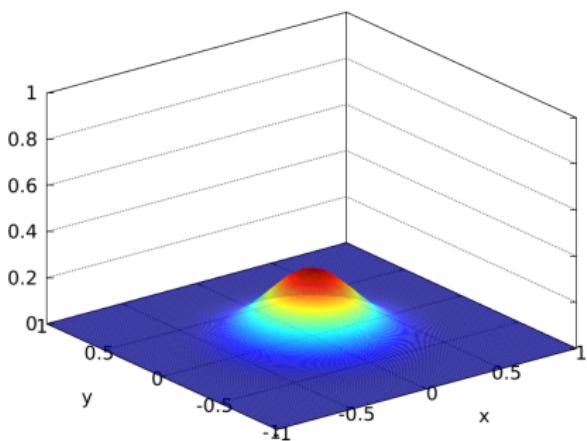
the look of success

- ▶ solving $T_t = D(T_{xx} + T_{yy})$ on 30×30 grid

initial condition $T(0, x, y)$

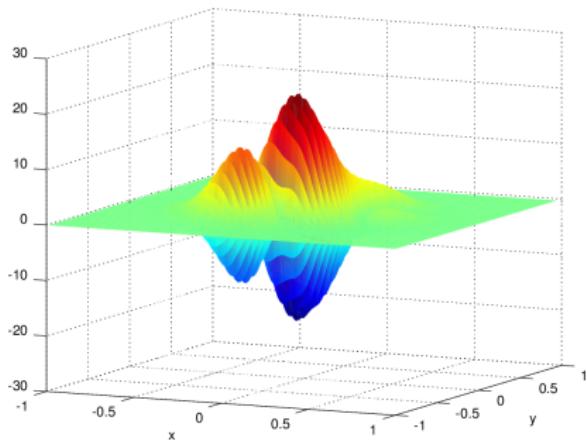
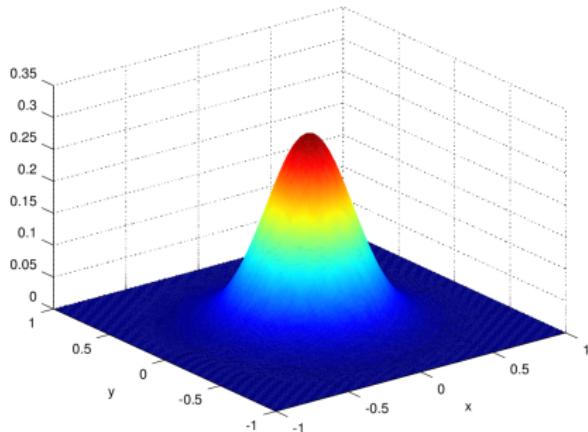


approximate solution $T(t, x, y)$ at
 $t = 0.02$ with $\Delta t = 0.001$



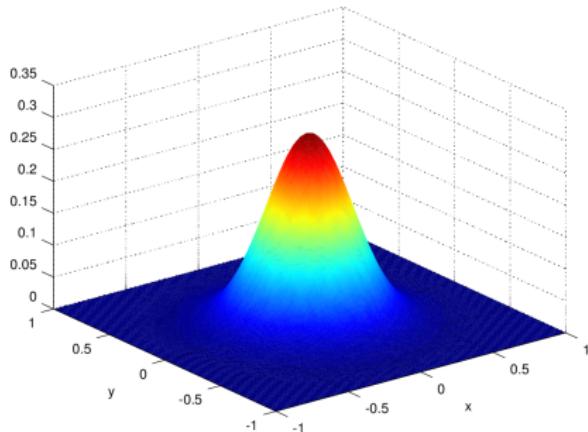
the look of instability

- ▶ both figures are from solving $T_t = D(T_{xx} + T_{yy})$ on the same space grid, but with slightly different time steps

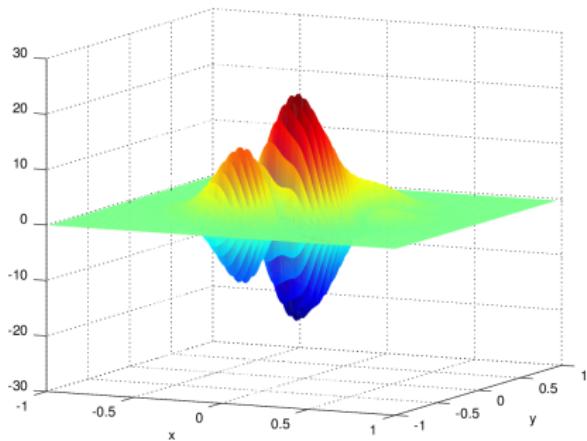


the look of instability

- ▶ both figures are from solving $T_t = D(T_{xx} + T_{yy})$ on the same space grid, but with slightly different time steps



$$\frac{D\Delta t}{\Delta x^2} = 0.2$$



$$\frac{D\Delta t}{\Delta x^2} = 0.4$$

avoid the instability

- ▶ recall 1D explicit scheme had the form

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

- ▶ thus the new value u_j^{n+1} is an average of the old values, *if the middle coefficient is positive*:

$$1 - 2\nu \geq 0 \iff \frac{D\Delta t}{\Delta x^2} \leq \frac{1}{2} \iff \Delta t \leq \frac{\Delta x^2}{2D}$$

- ▶ averaging is always stable because averaged wiggles are always smaller than the original wiggles
- ▶ this condition is a sufficient *stability criterion*
- ▶ so:

the result was unstable because the time step was too big

adaptive implementation: guaranteed stability

```
function T = heatadapt(D,L,J,K,tf,Tinitial)

dx = 2 * L / J;      dy = 2 * L / K;
[x,y] = ndgrid(-L:dx:L, -L:dy:L);
if nargin<6, Tinitial = exp(-30*(x.*x + y.*y)); end
T = Tinitial;

t = 0.0;    count = 0;
while t < tf
    dt0 = 0.25 * min(dx,dy)^2 / D;
    dt = min(dt0, tf - t);
    mu_x = dt * D / (dx*dx);    mu_y = dt * D / (dy*dy);
    T(2:J,2:K) = T(2:J,2:K) + ...
        mu_x * ( T(3:J+1,2:K) - 2 * T(2:J,2:K) + T(1:J-1,2:K) ) + ...
        mu_y * ( T(2:J,3:K+1) - 2 * T(2:J,2:K) + T(2:J,1:K-1) );
    t = t + dt;
    count = count + 1;
end

figure(1), surf(x,y,T), shading('interp'), xlabel x, ylabel y
```

heatadapt.m

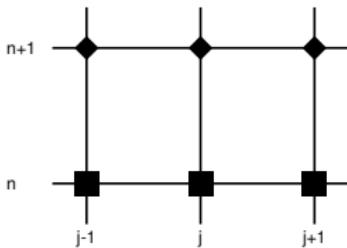
- ▶ same as heat.m except

choose time step from stability criterion

alternative instability fix: implicitness

- ▶ **implicit** methods can be designed to be stable for *any* positive time step Δt

- ▶ example is *Crank-Nicolson* scheme →
- ▶ has smaller error too: $O(\Delta t^2, \Delta x^2)$



- ▶ *but you must solve linear (or nonlinear) systems of equations to take each time step*
- ▶ Donald Knuth has advice for ice sheet modelers:

We should forget about small efficiencies . . . : premature optimization is the root of all evil.

variable diffusivity and time steps

- ▶ recall the analogy: (SIA) \leftrightarrow (heat eqn)
- ▶ the SIA has a diffusivity which varies in space, so consider a more general heat equation:

$$T_t = F + \nabla \cdot (D(x, y) \nabla T)$$

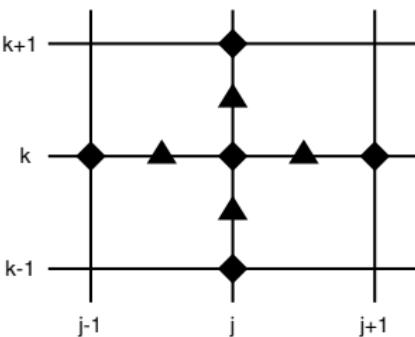
- ▶ the explicit method is conditionally stable with the same time step restriction if we evaluate diffusivity $D(x, y)$ at **staggered** grid points:

$$\begin{aligned}\nabla \cdot (D(x, y) \nabla u) &\approx \frac{D_{j+1/2,k}(T_{j+1,k} - T_{j,k}) - D_{j-1/2,k}(T_{j,k} - T_{j-1,k})}{\Delta x^2} \\ &+ \frac{D_{j,k+1/2}(T_{j,k+1} - T_{j,k}) - D_{j,k-1/2}(T_{j,k} - T_{j,k-1})}{\Delta y^2}\end{aligned}$$

in stencil at right:

diamonds: T

triangles: D



general diffusion equation code

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

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

t = 0.0;    count = 0;
while t < tf
    maxD = [max(max(Dup)) max(max(Ddown)) max(max(Dleft)) max(max(Dright))];
    maxD = max(maxD);
    if maxD <= 0.0
        dt = tf - t;
    else
        dt0 = 0.25 * min(dx,dy)^2 / maxD;
        dt = min(dt0, tf - t);
    end
    mu_x = dt / (dx*dx);    mu_y = dt / (dy*dy);
    Tb = T + b;
    T(2:J,2:K) = T(2:J,2:K) + ...
        mu_y * Dup .* ( Tb(2:J,3:K+1) - Tb(2:J,2:K) ) - ...
        mu_y * Ddown .* ( Tb(2:J,2:K) - Tb(2:J,1:K-1) ) + ...
        mu_x * Dright .* ( Tb(3:J+1,2:K) - Tb(2:J,2:K) ) - ...
        mu_x * Dleft .* ( Tb(2:J,2:K) - Tb(1:J-1,2:K) );
    T = T + F * dt;
    t = t + dt;    count = count + 1;
end
dtav = tf / count;
```

diffusion.m

- ▶ solves abstract diffusion equation $T_t = \nabla \cdot (D(x, y) \nabla T)$
- ▶ user supplies diffusivity on staggered grid

interruption: verification

- ▶ how do you make sure your *implemented* numerical ice flow code is correct?
 - *technique 1*: don't make any mistakes
 - *technique 2*: compare your model with others, and hope that the outliers are the ones with errors
 - *technique 3*: build-in a comparison to an exact solution, and actually measure the numerical error = **verification**
- ▶ where to get exact solutions for ice flow models?
 - textbooks: Greve and Blatter (2009), van der Veen (2013)
 - similarity solns to SIA (Halfar 1983; Bueler et al 2005)
 - manufactured solns to thermo-coupled SIA (Bueler et al 2007)
 - flowline and cross-flow SSA solns (Bodvarsson, 1955; van der Veen, 1985; Schoof, 2006; Bueler 2014)
 - flowline Blatter solns (Glowinski and Rappaz 2003)
 - constant viscosity flowline Stokes solns (Ladyzhenskaya 1963, Balise and Raymond 1985)
 - manufactured solns to Stokes equations (Sargent and Fastook 2010; Jouvet and Rappaz 2011; Leng et al 2013)

an exact solution of heat equation

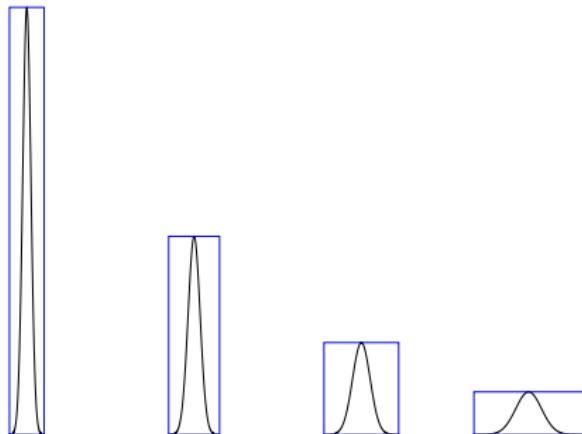
- ▶ the simple heat equation in 1D with constant diffusivity $D > 0$ is:

$$T_t = DT_{xx}$$

- ▶ many *exact* solutions to the heat equation are known
- ▶ I'll show the Green's function solution (a.k.a. "fundamental solution" or "heat kernel")
- ▶ it starts at time $t = 0$ with a "delta function" of heat at the origin $x = 0$ and then it spreads out over time
- ▶ we find it by a method which generalizes to the SIA

Green's function of heat equation

- ▶ the solution is “self-similar” over time
- ▶ as time goes it changes shape by
 - shrinking the output (vertical) axis and
 - lengthening the input (horizontal) axis
- ▶ ... but otherwise it is the same shape
- ▶ the integral over x is independent of time



increasing time →

similarity solutions

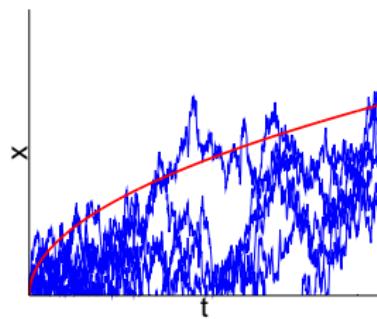
- ▶ Green's function of heat equation in 1D is

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

- ▶ “similarity” variables for 1D heat equation are

$$\begin{array}{ccc} \text{input scaling} & & \text{output scaling} \\ s & = & t^{-1/2}x, \quad T(t, x) & = & t^{-1/2}\phi(s) \end{array}$$

- ▶ 1905: Einstein discovers that the average distance traveled by particles in thermal motion scales like \sqrt{t} , so $s = t^{-1/2}x$ is an invariant



similarity solution to SIA

- ▶ 1981: P. Halfar discovers the similarity solution of the SIA in the case of flat bed and no surface mass balance
- ▶ Halfar's 2D solution (1983) for Glen flow law with $n = 3$ has scalings

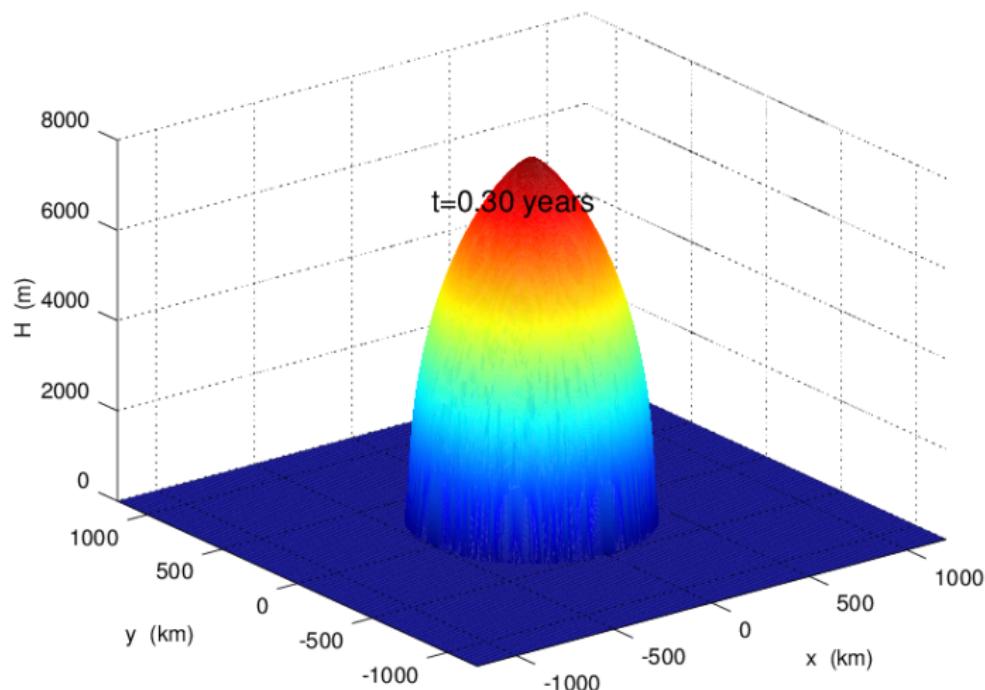
$$H(t, r) = t^{-1/9} \phi(s), \quad s = t^{-1/18} r$$

- ▶ ... so the diffusion of ice really slows down as the shape flattens out!

Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

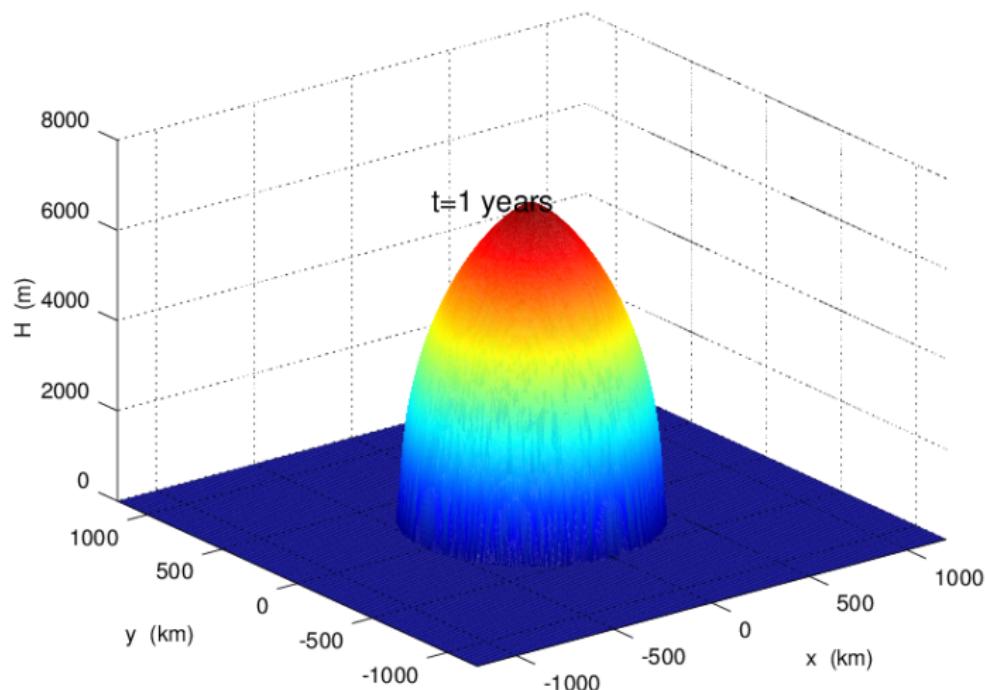
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

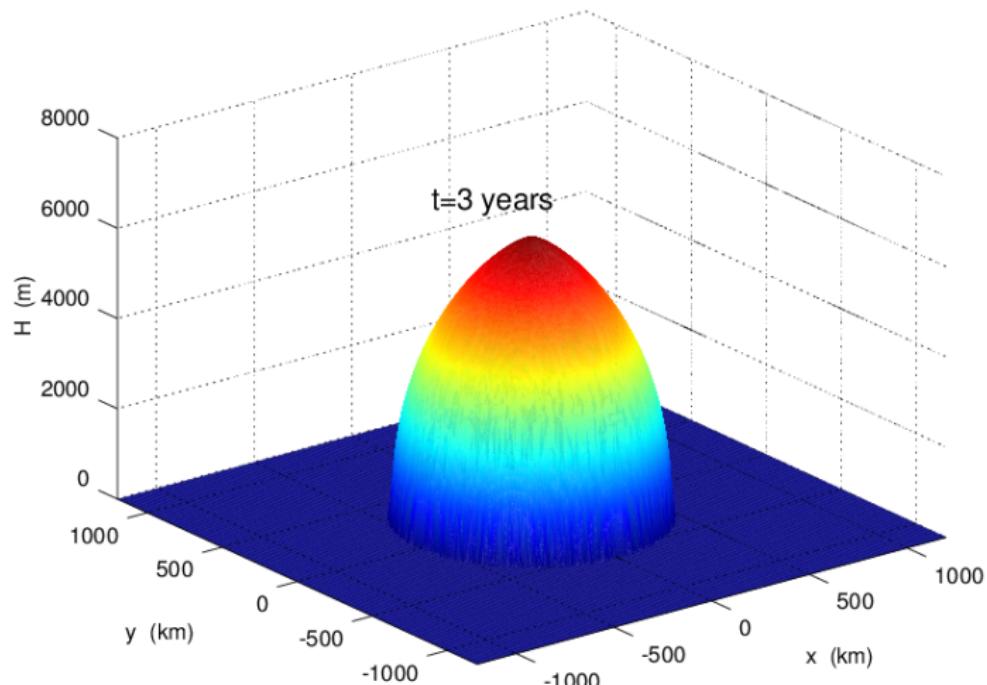
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

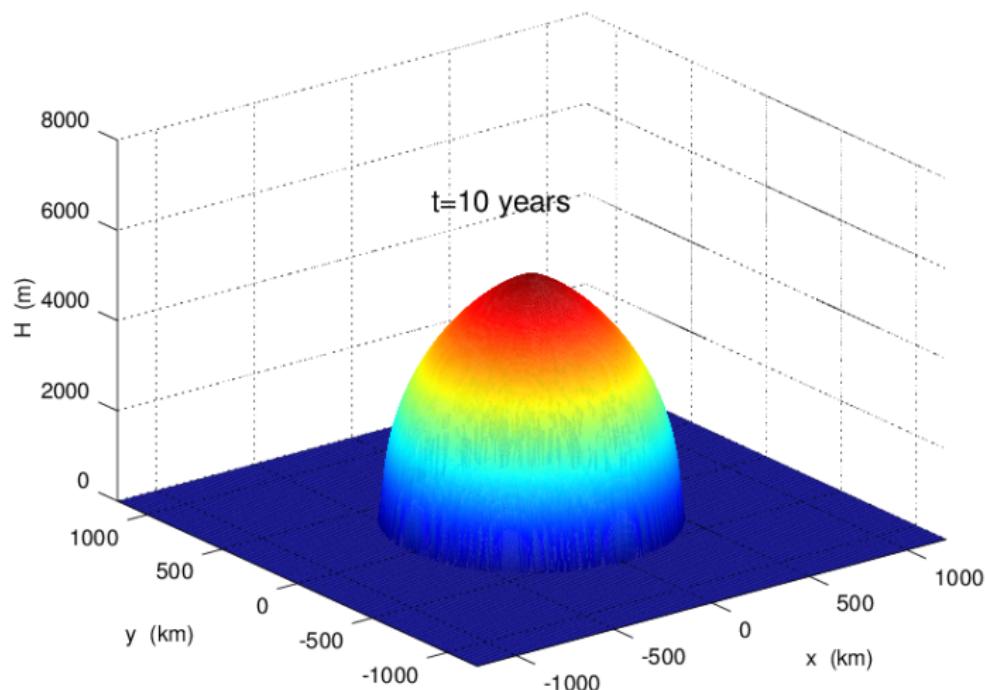
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

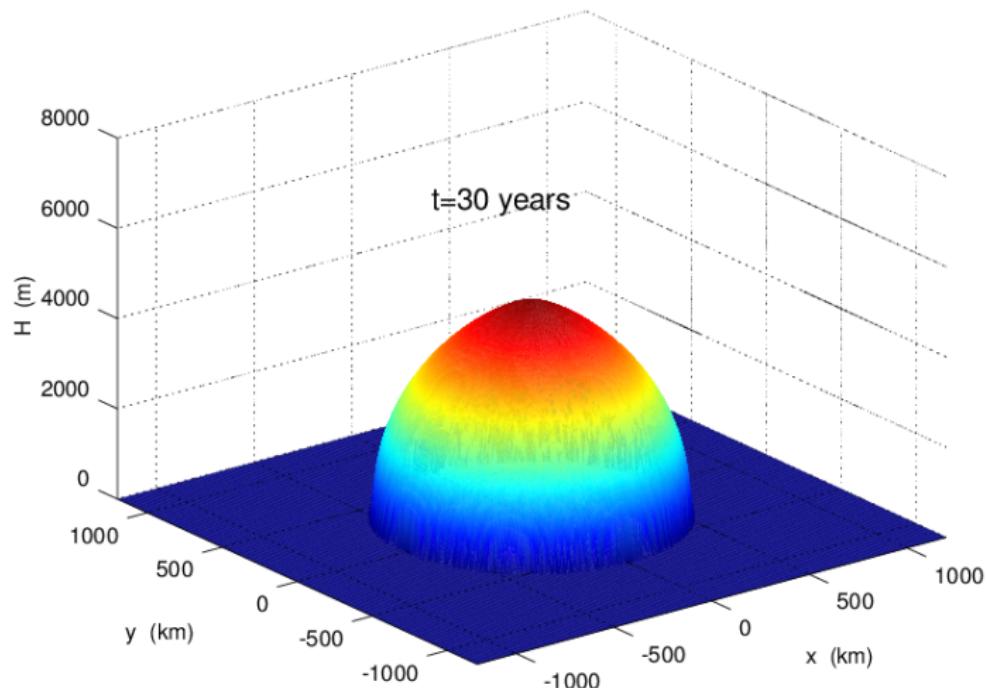
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

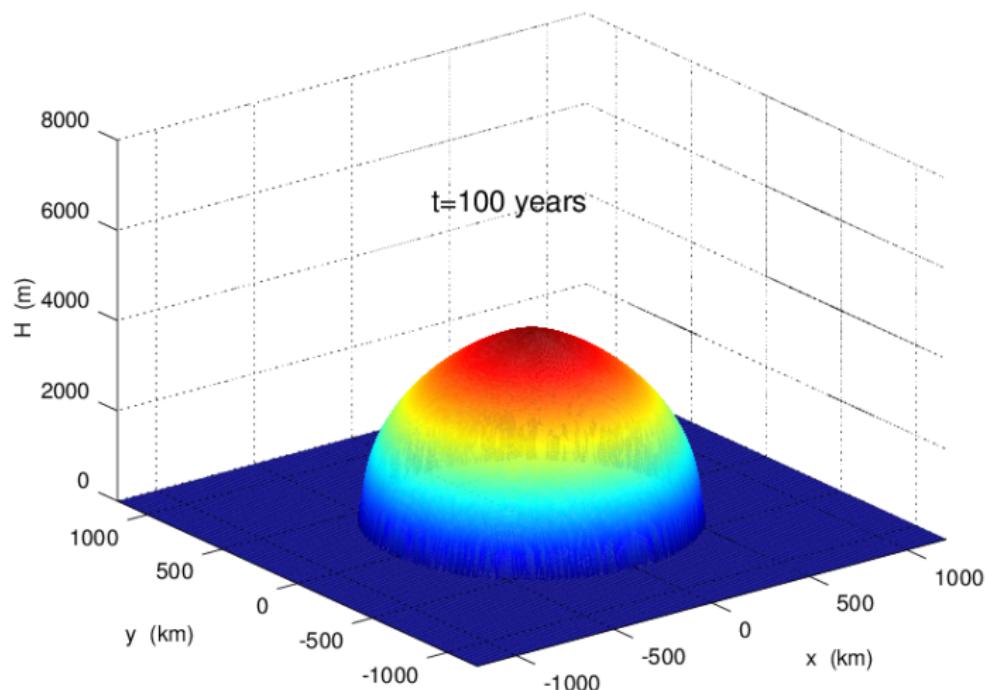
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

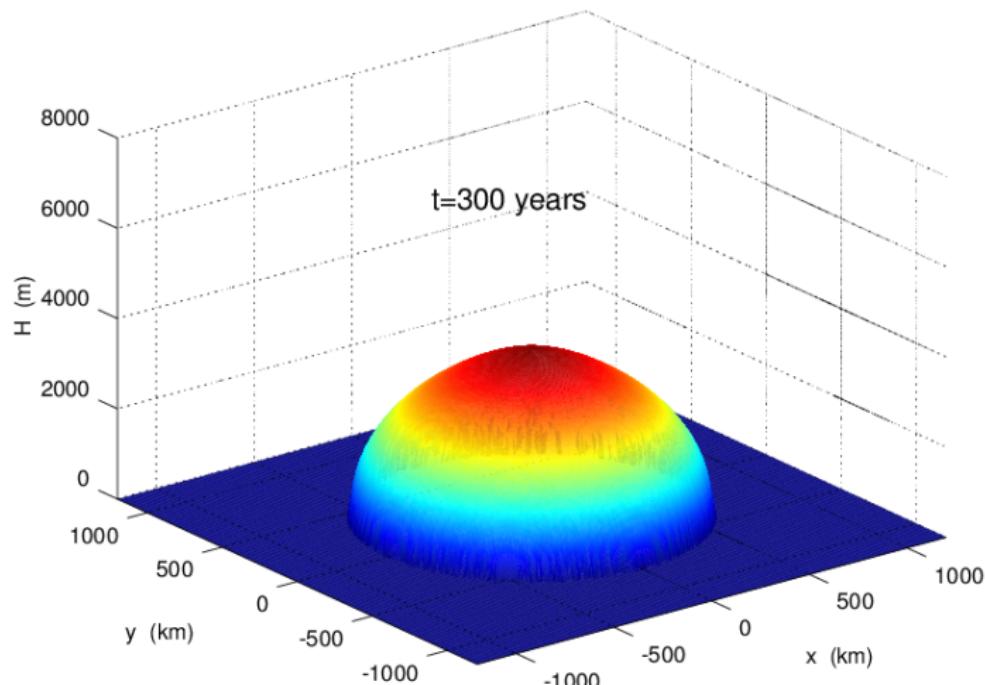
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

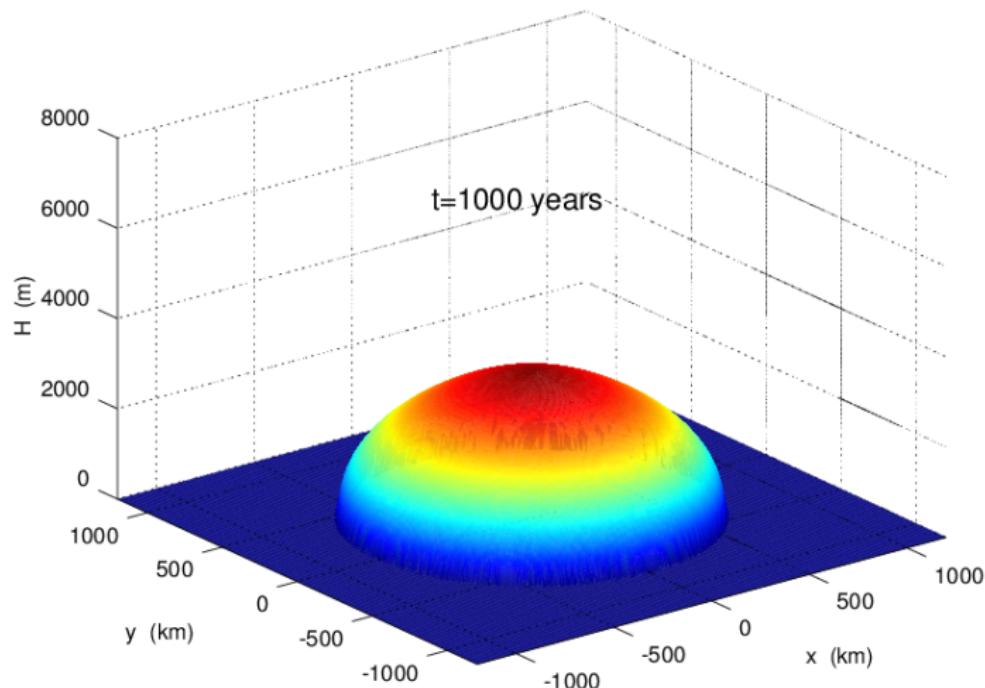
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

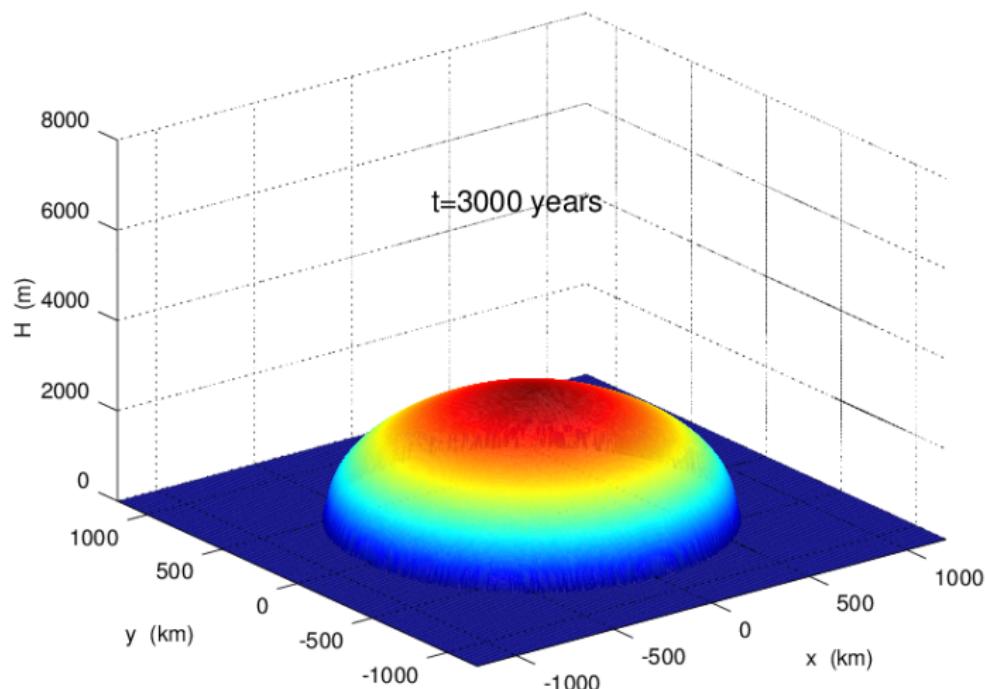
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

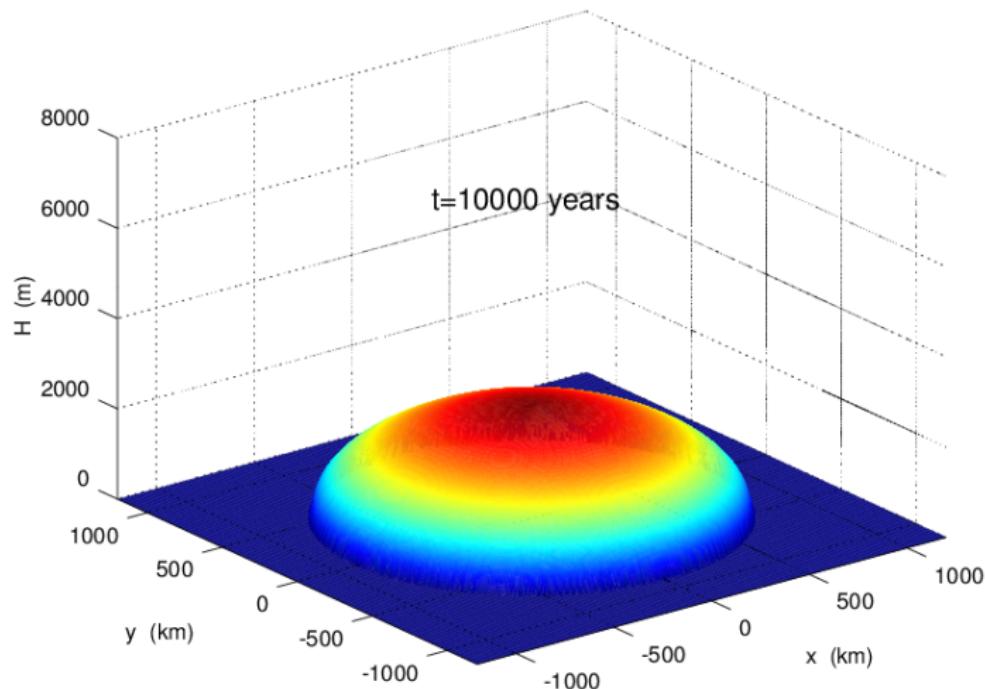
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

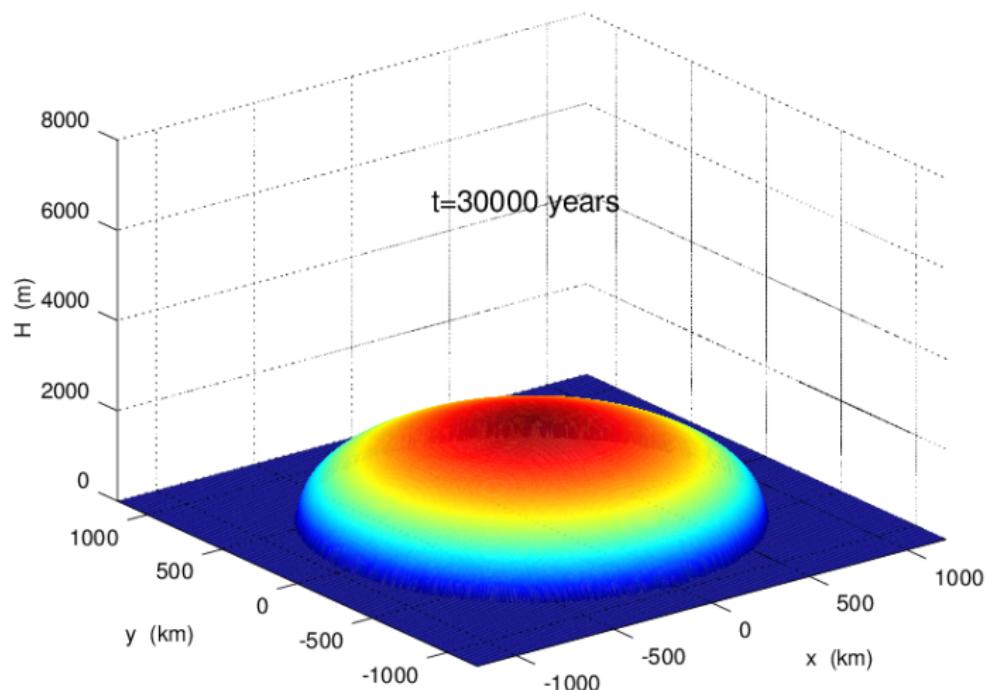
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

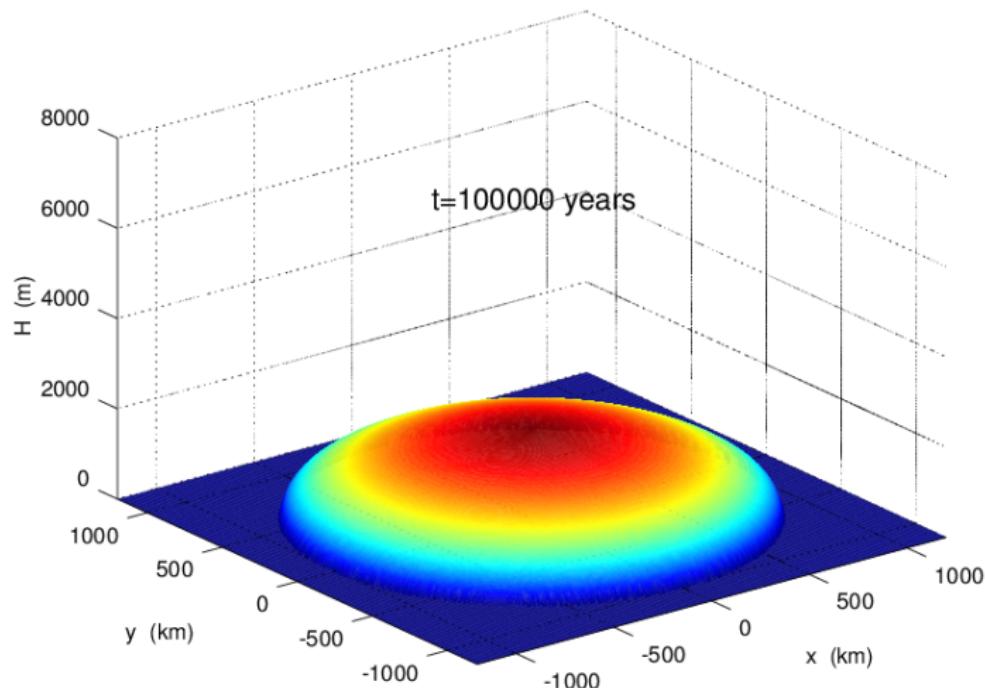
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

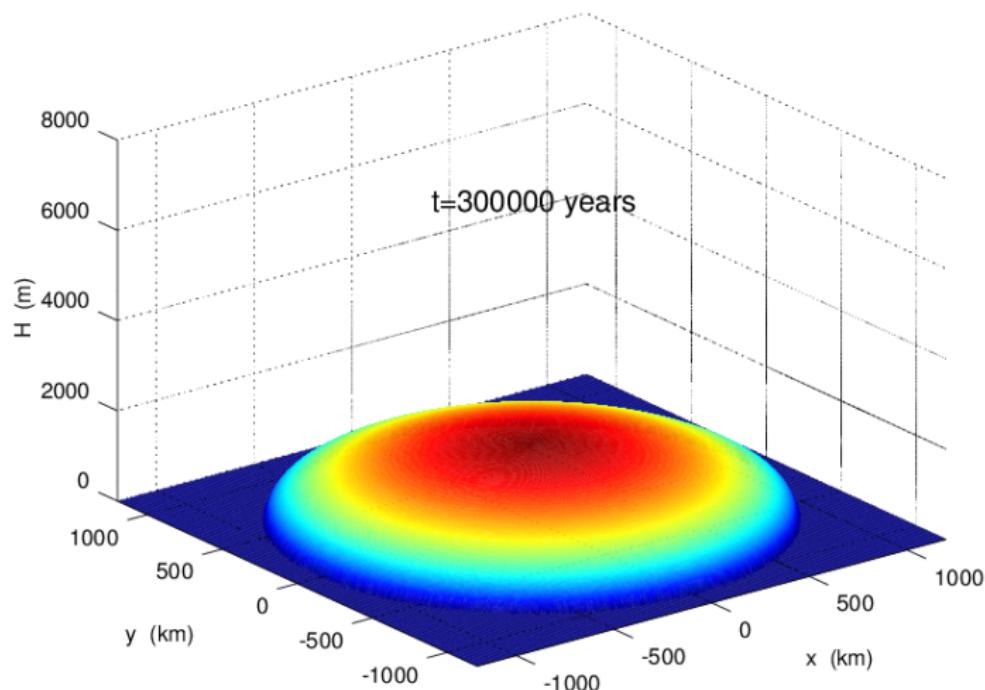
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

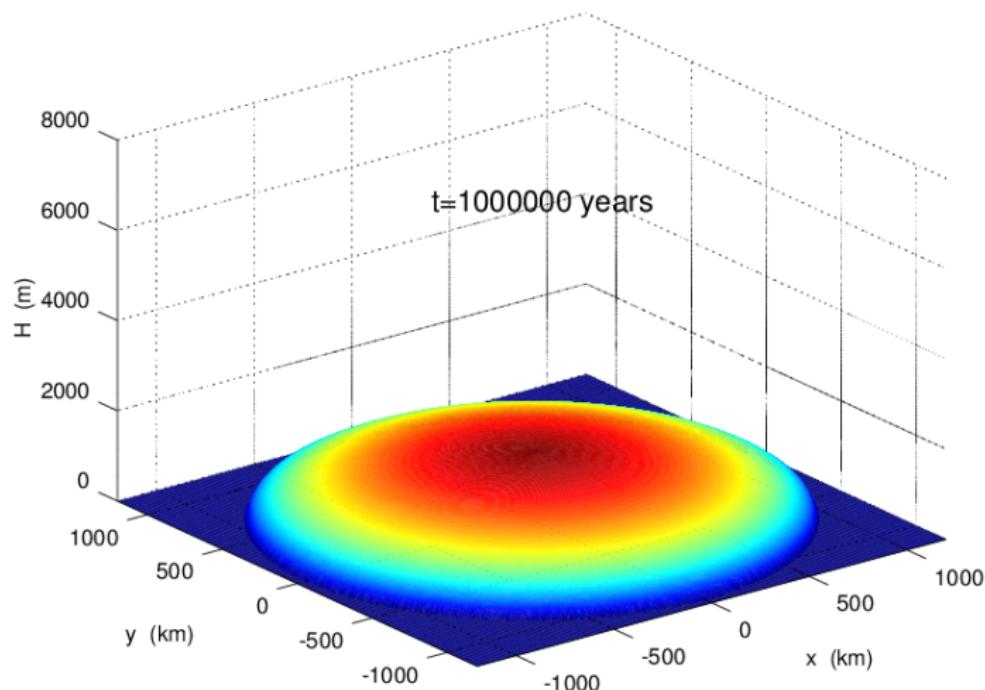
nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the movie

frames from $t = 0.3$ to $t = 10^6$ years

nearly equally-spaced in *exponential* time



Halfar solution to the SIA: the formula

- ▶ for $n = 3$ the solution formula is:

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

if H_0, R_0 are central height and ice cap radius

- ▶ ... at the “characteristic time”

$$t_0 = \frac{1}{18\Gamma} \left(\frac{7}{4} \right)^3 \frac{R_0^4}{H_0^7}$$

- ▶ it is a simple formula to use for verification!

is the Halfar solution *good for any modeling?*

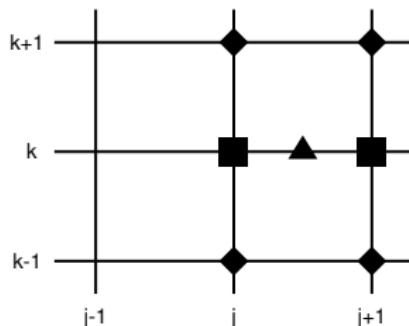
- ▶ John Nye and others (2000) compared different flow laws for the South Polar Cap on Mars
- ▶ they evaluated CO₂ ice and H₂O ice softness parameters by comparing the long-time behavior of the corresponding Halfar solutions
- ▶ conclusions:

... none of the three possible [CO₂] flow laws will allow a 3000-m cap, the thickness suggested by stereogrammetry, to survive for 10⁷ years, indicating that the south polar ice cap is probably not composed of pure CO₂ ice ... the south polar cap probably consists of water ice, with an unknown admixture of dust.

computing diffusivity in SIA

- ▶ back to numerics . . . we need diffusivity $D = \Gamma H^{n+2} |\nabla h|^{n-1}$ on the staggered grid
- ▶ various schemes proposed (Mahaffy, 1976; van der Veen 1999; Hindmarsh and Payne 1996)
- ▶ all schemes involve
 - averaging thickness H
 - differencing surface elevation h
 - in a “balanced” way, for better accuracy,

to get the diffusivity on staggered grid (triangle below)



- ▶ Mahaffy stencil →

SIA implementation: flat bed case

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

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

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

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

siaflat.m

verifying SIA code vs Halfar

```
>> verifyisia(20)
average abs error           = 22.310
maximum abs error          = 227.849
>> verifyisia(40)
average abs error           = 9.490
maximum abs error          = 241.470
>> verifyisia(80)
average abs error           = 2.800
maximum abs error          = 155.796
>> verifyisia(160)
average abs error           = 1.059
maximum abs error          = 109.466
```

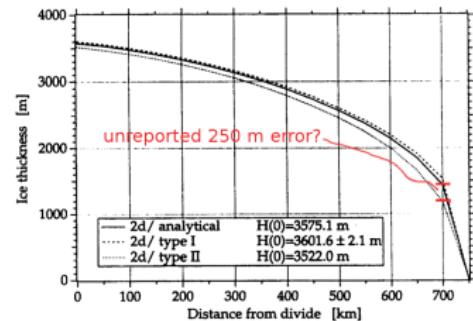
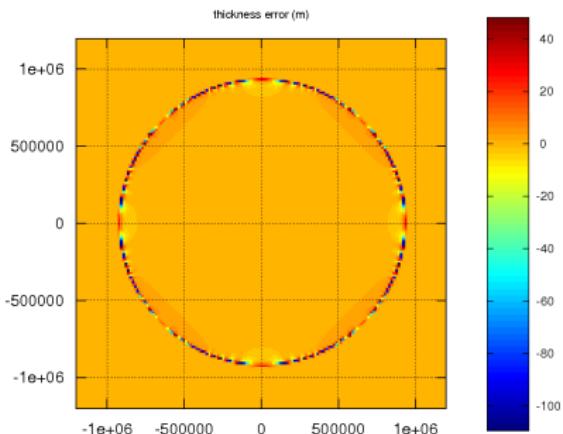
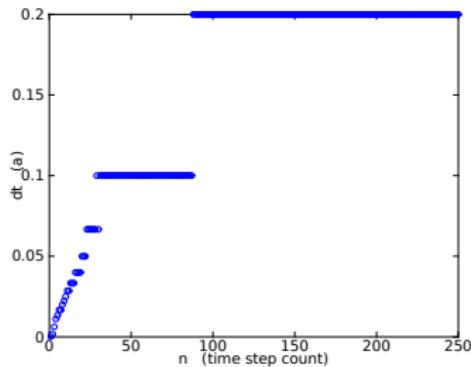
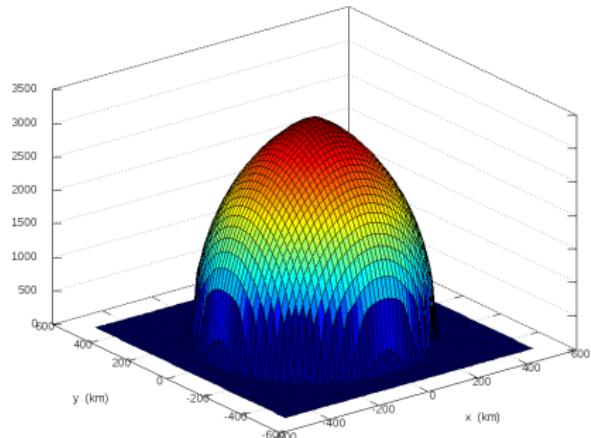
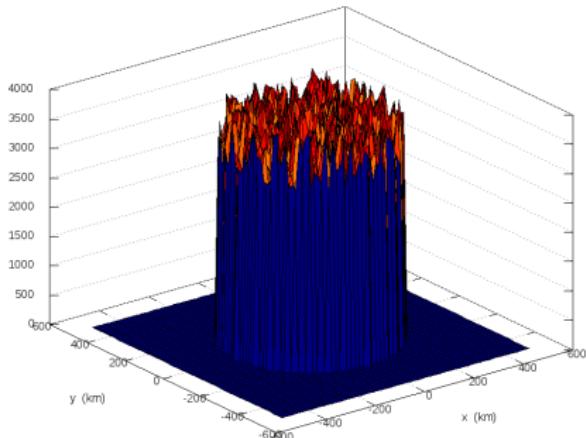


figure 2 in Huybrechts et al. (1996)

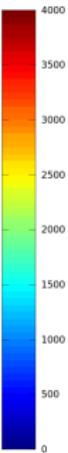
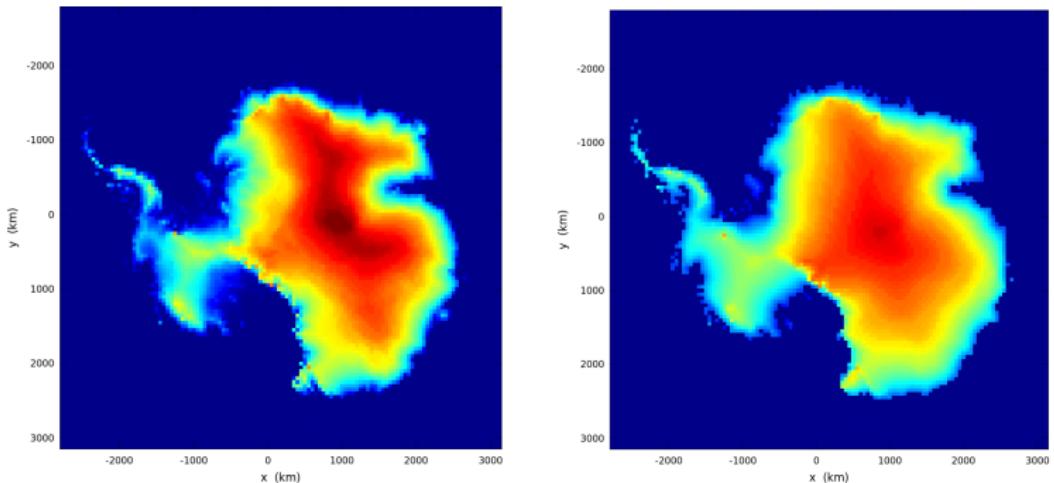
demonstrate robustness

`roughice.m` sets-up the nasty initial state at left and then calls
`siaflat.m` to evolve it for 50 years . . . it becomes nice dome at right



model the Antarctic ice sheet

- ▶ modify `siaflat.m` into `siageneral.m`:
 - observed accumulation as surface mass balance,
 - allow non-flat bed (so $H \neq h$),
 - calve any ice that is floating
- ▶ this makes a good exercise ... only 10 additional lines ...
- ▶ results from this *toy* Antarctic flow model, a 2000 model year run on a $\Delta x = 50$ km grid; runtime a few seconds



Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

shelves and streams

the most basic shallow assumption

- ▶ there are many shallow theories:
SIA, SSA, hybrids, Blatter, ...
- ▶ *all* make one assumption not required in Stokes:

the surface and base of the ice are given by functions $z = h(t, x, y)$ and $z = b(t, x, y)$
- ▶ surface overhang is not allowed
- ▶ (and most Stokes models make this assumption too)



not shallow!

three equations for geometry change

- ▶ let a be the climatic (surface) mass balance function; $a > 0$ corresponds to accumulation
- ▶ s be the basal melt rate function (basal mass balance); $s > 0$ is basal melting
- ▶ let $M = a - s$: “climatic-basal mass balance function” in glossary
- ▶ define the map-plane flux of ice,

$$\mathbf{q} = \int_b^h (u, v) dz = \bar{\mathbf{U}} H$$

- ▶ the three equations for glacier geometry change:

surface kinematical	$h_t = a - u _h h_x - v _h h_y + w _h$
base kinematical	$b_t = s - u _b b_x - v _b b_y + w _b$
mass continuity	$H_t = M - \nabla \cdot \mathbf{q}$

kinematic and mass continuity equations

- ▶ what does the “most basic shallow assumption” get you?
- ▶ *answer:* of these three equations,
 - surface kinematical
 - base kinematical
 - mass continuity

any two imply the third

- ▶ to show the above, recall:
 - the incompressibility of ice

$$u_x + v_y + w_z = 0$$

- and the Leibniz rule for differentiating integrals

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

kinematic and mass continuity equations 2

- ▶ literature is full of incomplete calculations of these equivalences
- ▶ ... usually mixed in with small-parameter arguments about shallowness
- ▶ most ice sheet models use the mass continuity equation
- ▶ ... but they could instead use the surface kinematical equation

standard recipe for ice sheet models

- ▶ the ingredients of a typical ice sheet model:
 1. numerical implementation of a stress balance: compute velocity (u, v, w)
 2. from the horizontal velocity (u, v) and the surface balance, do time-step of mass continuity equation to get H_t
 3. update surface elevation
 4. decide on time-step from size of diffusivity D , and repeat at 1.

the mass continuity equation: a summary

- ▶ the *mass continuity equation* is

$$H_t = M - \nabla \cdot (\mathbf{u}H)$$

- ▶ the numerical nature of this equation depends on the stress balance:
 - the equation is a diffusion for frozen bed, large scale flows (i.e. SIA)
 - it is *not* very diffusive for membrane stresses and no basal resistance (e.g. SSA for ice shelves)
 - it is diffusive for ice streams (but how much?)
 - there is *not* much helpful theory on this transport problem
 - ... maybe you will help find this theory!

Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

shelves and streams

flow model II: shallow shelf approximation (SSA) stress balance

SSA model applies very well to **ice shelves**

- ▶ ... for parts away from grounding lines
- ▶ ... and away from calving fronts

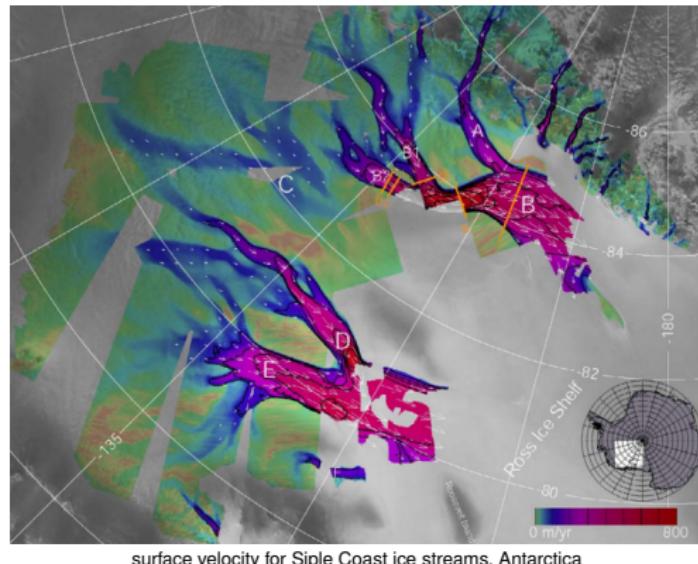


Ekström ice shelf(Hans Grobe)

shallow shelf approximation stress balance 2

SSA also applies reasonably well to **ice streams**

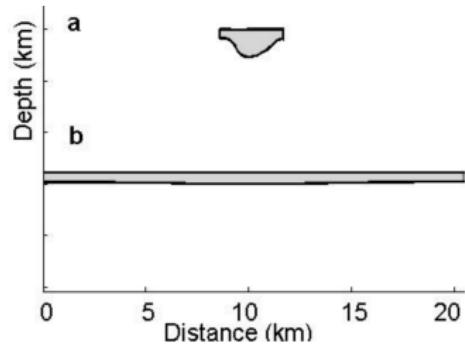
- ▶ ... with modest bed topography
- ▶ ... and weak bed strength²
- ▶ imperfect near shear margins and grounding lines



²energy conservation (esp. ice temperature and basal melt) and subglacial hydrology (esp. subglacial water pressure) are major aspects of ice stream flow ... but not addressed here

what is, *and is not*, an ice stream?

- ▶ ice streams
 - fast surface speed (up to 1 m a^{-1})
 - have concentrated vertical shear in thin layer near base ("sliding")
- ▶ outlet glaciers
 - fast surface speed (up to 10 km a^{-1})
 - uncertain how much is sliding
 - substantial vertical shear "up" in the ice column,
 - not-at-all flat bed topography
 - soft, temperate ice may play a big role
- ▶ few simplifying assumptions are appropriate for outlet glaciers



Jakobshavn Isbrae (**a**) and Whillans Ice Stream (**b**); plotted without vertical exaggeration (Truffer and Echelmeyer 2003)

SSA stress balance equation

- ▶ only plane flow case (“flow line”) shown here
- ▶ the stress balance equation determines velocity (*ice stream case*):

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

- ▶ the **red term** inside parentheses is the vertically-integrated “longitudinal” or “membrane” stress
- ▶ the **blue term** is basal resistance
- ▶ the **green term** is driving stress
- ▶ derived originally by Morland (1987), MacAyeal (1989)
- ▶ *how to think about this equation?*
- ▶ *how do you solve it numerically?*

flotation criterion and grounding line

- ▶ the inequality “ $\rho H < -\rho_w b$ ” is the **flotation criterion**
- ▶ at the grounding line $x = x_g$ the above inequality switches
- ▶ ... and the driving stress switches form:
 - on the grounded side we know $\rho H > -\rho_w b$ so

$$\rho g H h_x = \rho g H (H_x + b_x)$$

- on the floating side we know $\rho H < -\rho_w b$ so $h = (1 - \rho/\rho_w)H$ and so

$$\rho g H h_x = \rho(1 - \rho/\rho_w)g H H_x$$

- ▶ also: H, u, u_x are all continuous at $x = x_g$

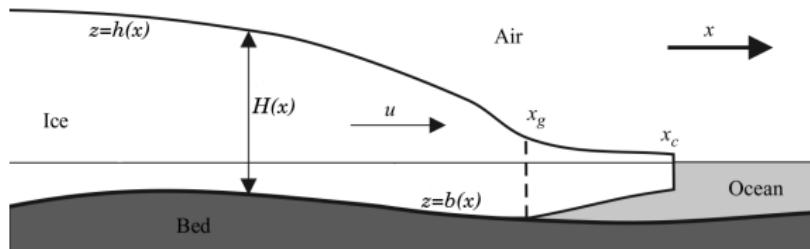
flow line model: from stream to shelf

$$u = u_0 \quad \text{at } x = 0$$

$$\left. \begin{aligned} & \left(2A^{-1/n}H|u_x|^{1/n-1}u_x \right)_x - C|u|^{m-1}u = \rho g H h_x \\ & h = H + b \end{aligned} \right\} \quad \text{on } 0 < x < x_g$$

$$\left. \begin{aligned} & \left(2A^{-1/n}H|u_x|^{1/n-1}u_x \right)_x + 0 = \rho g H h_x \\ & h = (1 - \rho/\rho_w)H \end{aligned} \right\} \quad \text{on } x_g < x < x_c$$

$$2A^{-1/n}H|u_x|^{1/n-1}u_x = \frac{1}{2}\rho(1 - \rho/\rho_w)gH^2 \quad \text{at } x = x_c$$

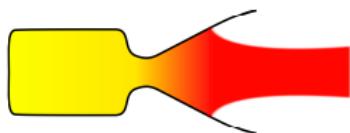
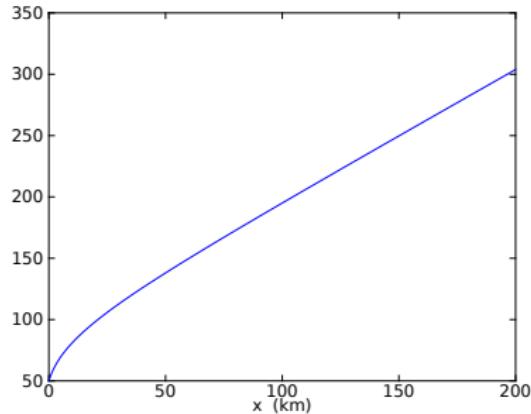
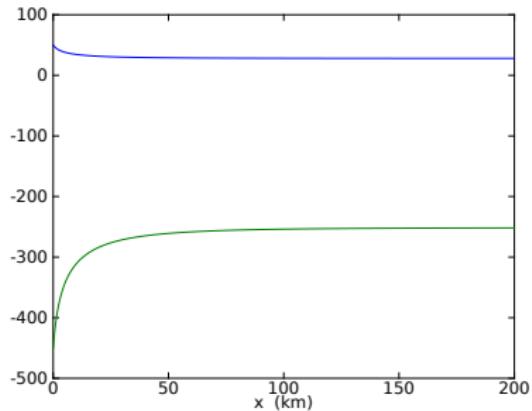


exact velocity and thickness for steady ice shelf

- ▶ limited goal here: describe a steady state, 1D ice shelf
- ▶ there is a nice **by-hand** result (van der Veen 1983; next slide): the thickness and velocity in the ice shelf can be completely determined in terms of the
 1. ice thickness H_g at the grounding line and
 2. ice velocity u_g at the grounding line
- ▶ we will use this to
 - understand the SSA better
 - verify a numerical SSA code

exact velocity and thickness for steady ice shelf 2

see `testshelf.m`



numerically solving the SSA stress balance

- ▶ here we fix ice thickness $H(x)$ and find the velocity numerically
- ▶ the stress balance is a nonlinear equation in the velocity:

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

- ▶ iteration is needed
- ▶ I'll describe the numerical method for a shelf *or* stream, but only give a code for an ice shelf

numerically solving the SSA stress balance 2

- ▶ coefficient $\bar{\nu} = A^{-1/n}|u_x|^{1/n-1}$ is the “effective viscosity”:

$$(2\bar{\nu} Hu_x)_x - C|u|^{m-1}u = \rho g H h_x$$

- ▶ *simplest iteration idea*: use old effective viscosity to get new velocity solution, and repeat until things stop changing
 - this is “Picard” iteration
 - (Newton iteration is a superior alternative)
- ▶ specifically for Picard:
 - last iterate $u^{(k-1)}$
 - define $W^{(k-1)} = 2\bar{\nu}H = 2A^{-1/n}|u_x^{(k-1)}|^{1/n-1}H$
 - current iterate (unknown) $u^{(k)}$
 - solve repeatedly:

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

solving the “inner” linear problem

- ▶ abstract the problem:

$$(W(x) u_x)_x - \alpha(x) u = \beta(x)$$

on $0 < x < L$, with boundary conditions

$$u(0) = V, \quad u_x(L) = \gamma$$

- ▶ an *elliptic* PDE boundary value problem
- ▶ $W(x), \alpha(x), \beta(x)$ are known functions in the SSA context:
 - both $W(x)$ and $\alpha(x)$ come from previous iteration
 - $\beta(x)$ is driving stress

where do you get an initial guess $u^{(0)}$?

- ▶ for floating ice, a possible initial guess for velocity comes from assuming a uniform strain rate:

$$u^{(0)}(x) = \gamma(x - x_g) + u_g$$

where γ is the value of u_x found from calving front stress imbalance

- ▶ for grounded ice, a possible initial guess for velocity is to assume ice is held by basal resistance only:

$$u^{(0)}(x) = (-C^{-1} \rho g H h_x)^{1/m}$$

numerics of the “inner” linear problem

- ▶ suppose $j = 1, 2, \dots, J + 1$, where $x_1 = x_g$ and $x_{J+1} = x_c$ are endpoints
- ▶ $W(x)$ is needed on the staggered grid; the approximation is:

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

- ▶ left-hand boundary condition: $u_1 = V$ given
- ▶ right-hand boundary condition (“ $u_x(L) = \gamma$ ”):
 - introduce notional point x_{J+2}
 -
 - $$\frac{u_{J+2} - u_J}{2\Delta x} = \gamma$$
 - using equation * in $j = J + 1$ case, eliminate u_{J+2} variable
“by-hand” before coding numerics

numerics of the “inner” linear problem 2

- so SSA stress balance has form $\mathbf{Ax} = \mathbf{b}$, namely:

$$\begin{bmatrix} 1 & & & \\ W_{3/2} & A_{22} & W_{5/2} & \\ & W_{5/2} & A_{33} & \\ & & \ddots & \\ & & & \ddots \\ & W_{J-1/2} & A_{JJ} & W_{J+1/2} \\ & & A_{J+1,J} & A_{J+1,J+1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_J \\ u_{J+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \beta_2 \Delta x^2 \\ \beta_3 \Delta x^2 \\ \vdots \\ \beta_J \Delta x^2 \\ b_{J+1} \end{bmatrix}$$

- with diagonal entries

$$A_{22} = -(W_{3/2} + W_{5/2} + \alpha_1 \Delta x^2)$$

$$A_{33} = -(W_{5/2} + W_{7/2} + \alpha_2 \Delta x^2)$$

and so on, up to A_{JJ} ,

- with special cases in last equation:

$$A_{J+1,J} = 2W_{J+1/2}$$

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

$$b_{J+1} = -2\gamma \Delta x W_{J+3/2} + \beta_{J+1} \Delta x^2$$

- this is a *tridiagonal* system

numerics of the “inner” linear problem 3

```
function u = flowline(L,J,gamma,W,alpha,beta,v0)

dx = L / J;
rhs = dx^2 * beta(:);
rhs(1) = v0;
rhs(J+1) = rhs(J+1) - 2 * gamma * dx * W(J+1);

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

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

u = A \ rhs;
```

flowline.m

testing the “inner” linear code

- ▶ before proceeding to solve nonlinear SSA problem, we can test the “abstracted” code `flowline.m`
- ▶ test by “manufacturing” solutions
 - see `testflowline.m`; not shown
- ▶ results:
 - converges at optimal rate $O(\Delta x^2)$

numerical: SSA

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

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

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

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

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

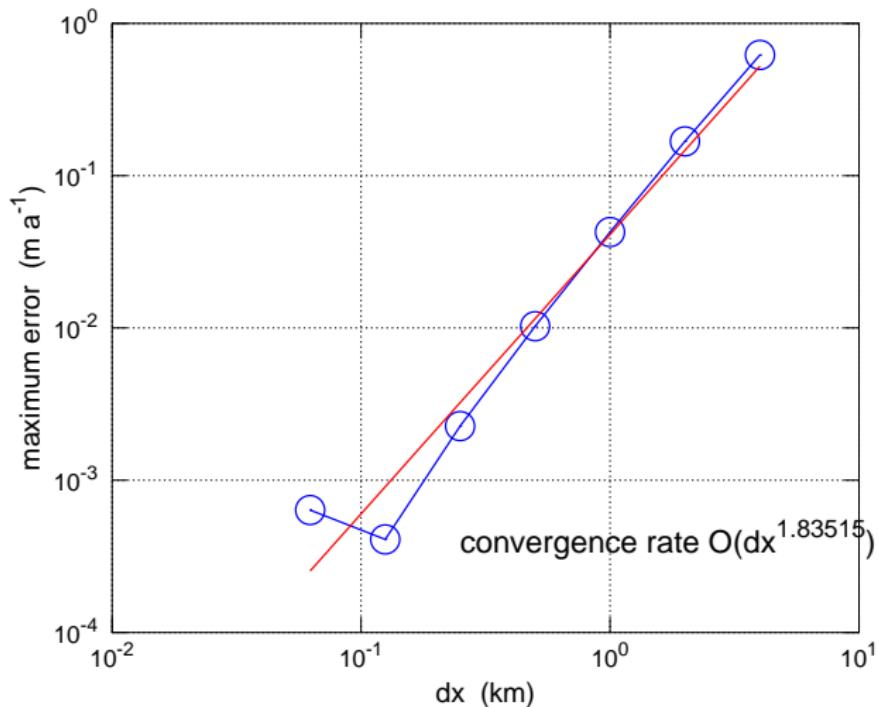
Hstag = stagav(H);
tol = 1.0e-14;
eps_reg = (1.0 / p.secpera) / p.L;
maxdiff = Inf; W = zeros(J+1,1); iter = 0;
while maxdiff > tol
    uxstag = stagslope(dx,u);
    sqr_ux_reg = uxstag.^2 + eps_reg^2;
    W(1:J) = 2 * p.A^(-1/p.n) * Hstag .* sqr_ux_reg.^(((1/p.n)-1)/2.0);
    W(J+1) = W(J);

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

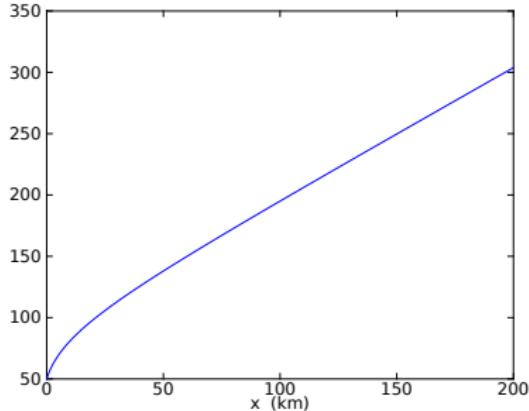
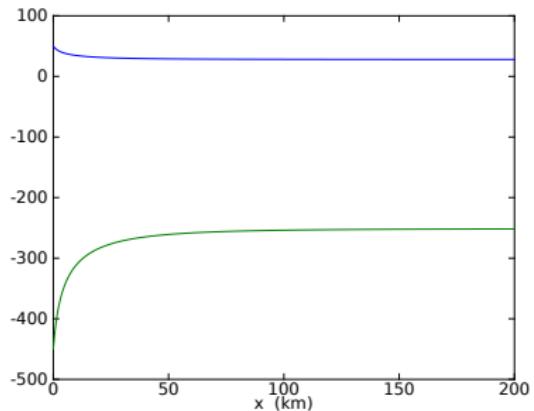
ssaflowline.m

numerical thickness and velocity for steady ice shelf

lines below are a convergence analysis of `testshelf.m`, which calls `ssaflowline.m`:



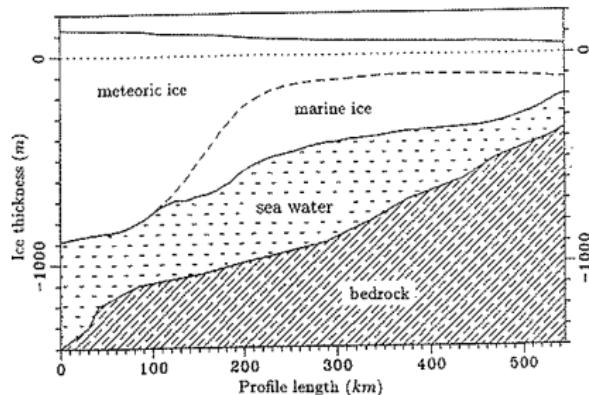
SSA model output



- ▶ *this looks suspiciously like figures for the exact solution . . .*
- ▶ yes

realistic ice shelf modeling

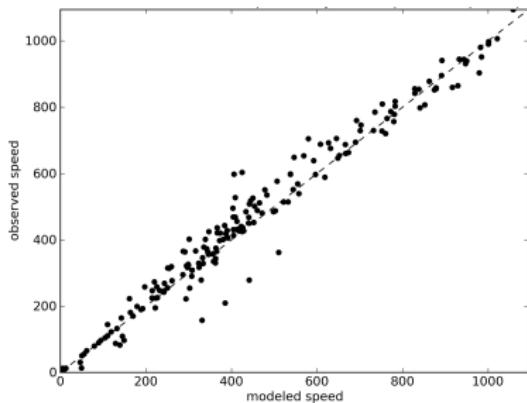
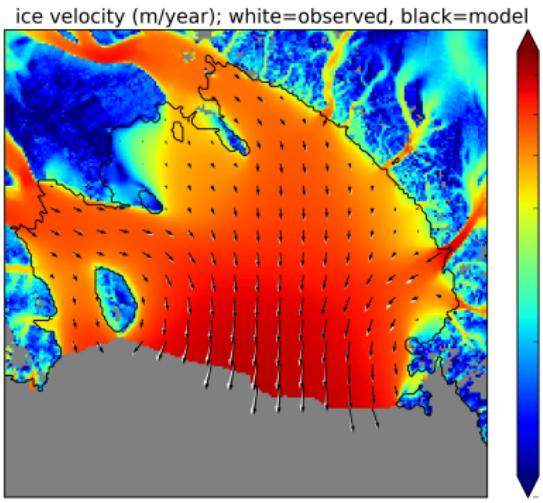
- ▶ flow lines are never very realistic
- ▶ you can add parameterized “side drag” ...
- ▶ also, ice shelves have surprises:
 - high basal melt near grounding lines
 - marine ice can freeze-on at bottom (below)
 - reverse sloped-bed (marine ice sheet) instability



from Grosfeld & Thyssen 1994

ice shelf modeling in 2D

- ▶ nonetheless “diagnostic” (static geometry) ice shelf modeling in 2d has been quite successful
- ▶ observed surface velocities validate SSA stress balance model
 - e.g. Ross ice shelf example below using PISM
 - ... but many models can do this



numerical solution of stress balances: a summary

- ▶ stress balance equations (e.g. SSA or Stokes) determine velocity from geometry and boundary conditions
 - nonlinear so iteration is necessary
 - at each iteration a sparse matrix “inner” problem is solved . . . give it to a matrix solver software package

conclusion: general principles

- ▶ learn a version-control system (git)
- ▶ modularize your codes
- ▶ test the parts, and make testing repeatable (make test)
- ▶ if you are stuck, write documentation for what you have

conclusion: a last take-away

- ▶ this thing



does not know what you *intend*, i.e. your physical modelling goal

- it does “know” your discrete equations, the program you wrote
- ▶ you *can* aspire for your computer to “know” your continuum model, the one that you write as equations with derivatives
 - achieve this through verification with exact solutions