

Numerical modelling of glaciers, ice sheets, and ice shelves

Ed Bueler

Dept of Math & Stat, University of Alaska Fairbanks
elbueler@alaska.edu

McCarthy, Alaska, June 2018

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

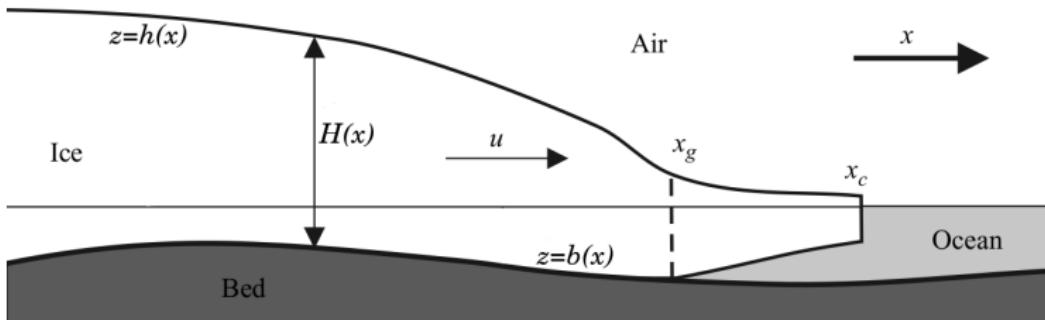


figure modified from Schoof (2007)

- ▶ 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 Glen flow law exponent (usually = 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 based on 18 Matlab/Octave codes
- ▶ several will appear in these lectures, but not all
- ▶ each is ~ 1/2 page
- ▶ please give them a try!
 - .zip and .tar.gz forms available from USB drive
 - and online:

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

Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

shelves and streams

free advice

ice in glaciers is a *fluid*

- ▶ what's a fluid?
- ▶ is it just a collection of particles?



ice in glaciers is a *fluid*

- ▶ what's a fluid?
- ▶ is it just a collection of particles?
- ▶ its a mathematical abstraction
- ▶ ... with equations relating these fields:
 - a scalar *density* $\rho(t, x, y, z)$
 - a scalar *pressure* $p(t, x, y, z)$
 - a vector *velocity* $\mathbf{u}(t, x, y, z)$



ice in glaciers is an atypical fluid

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

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

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

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

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

- stress balance equation is “ $ma = F$ ”
- unclaimed \$1 million prize for showing this is a good model

glaciology as computational fluid dynamics

- ▶ **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 stratification
- ▶ 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 *slow* and *non-Newtonian*

- “slow” is a technical term:

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

- ice is non-Newtonian in a “shear-thinning” way
 - ▶ higher strain rates means lower viscosity
 - ▶ viscosity ν is not constant

- ▶ thus the standard model is Glen-law Stokes:

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

incompressibility

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

stress balance

$$\mathbf{D}_{ij} = A \tau^{n-1} \tau_{ij}$$

flow law

“slow” means no memory of velocity/momentum

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

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

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

$$\mathbf{D}_{ij} = A\tau^{n-1} \boldsymbol{\tau}_{ij}$$

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

¹you don't need . . . to know which way the wind blows

plane flow Stokes

- ▶ suppose we work in a x, z plane, such as the centerline of a glacier, or a cross-flow plane
- ▶ now the $n = 3$ 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)

- *notation:* subscripts x, z denote partial derivatives, τ_{13} is the “vertical” shear stress, τ_{11} and $\tau_{33} = -\tau_{11}$ are (deviatoric) longitudinal stresses

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

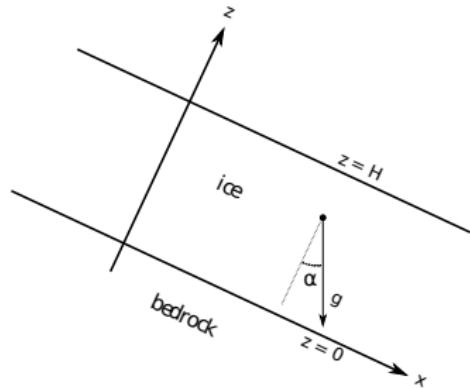
slab-on-a-slope

- ▶ suppose constant thickness
- ▶ tilt bedrock by angle α
- ▶ rotate the coordinates
- ▶ get these new expressions:

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

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

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



- ▶ for this **slab-on-a-slope** there is *no variation in x*: $\partial/\partial x = 0$
- ▶ the equations simplify:

$w_z = 0$	$0 = \tau_{11}$
$\tau_{13,z} = -\rho g \sin \alpha$	$u_z = 2A\tau^2 \tau_{13}$
$p_z = -\rho g \cos \alpha$	

slab-on-a-slope 2

- ▶ add boundary conditions:

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

- ▶ by integrating vertically, get:

$$w = 0$$

$$p = \rho g \cos \alpha (H - z)$$

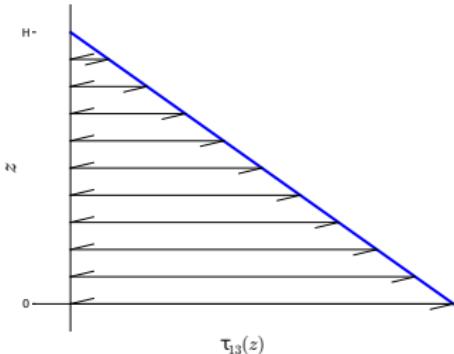
$$\tau_{13} = \rho g \sin \alpha (H - z)$$

- ▶ τ_{13} is linear in depth

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

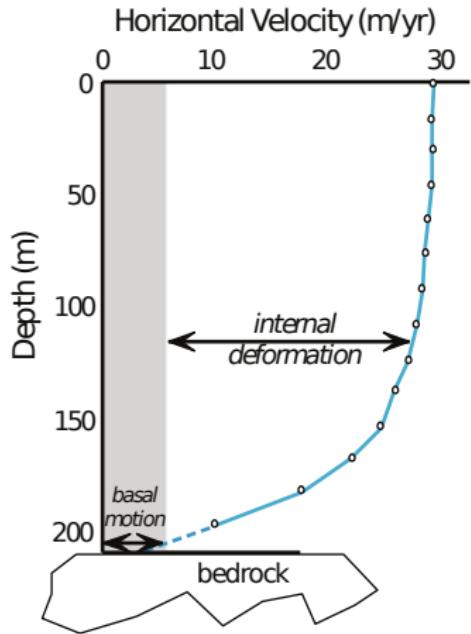
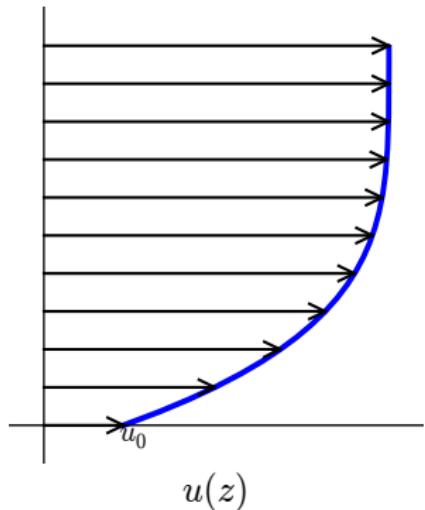
$$u(z) = u_0 + 2A(\rho g \sin \alpha)^3 \int_0^z (H - z')^3 dz'$$

$$= u_0 + \frac{1}{2} A (\rho g \sin \alpha)^3 (H^4 - (H - z)^4)$$



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

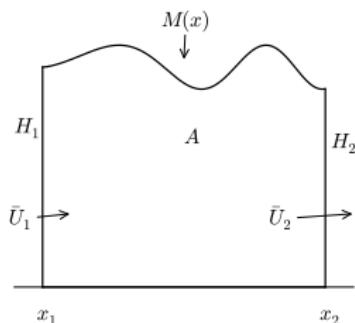
Q: having computed the velocity $u = u(t, x, z)$... so what?

A: velocity partly determines the new ice shape through **mass conservation**

► suppose our ice has variable thickness $H(t, x)$

► define the vertical average of velocity:

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



► with climatic mass balance $M(x)$, consider change of area in the figure:

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

◦ area A in 2D becomes volume V in 3D

► assume width $dx = x_2 - x_1$ is small, so $A \approx dx H$, and divide by dx to get the *mass continuity (conservation) equation*

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

combine equations so far

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

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

- ▶ note $\sin \alpha \approx \tan \alpha = -h_x$
- ▶ 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 the “shallow ice approximation” (SIA)
 - it’s a very rough explanation of it

Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

shelves and streams

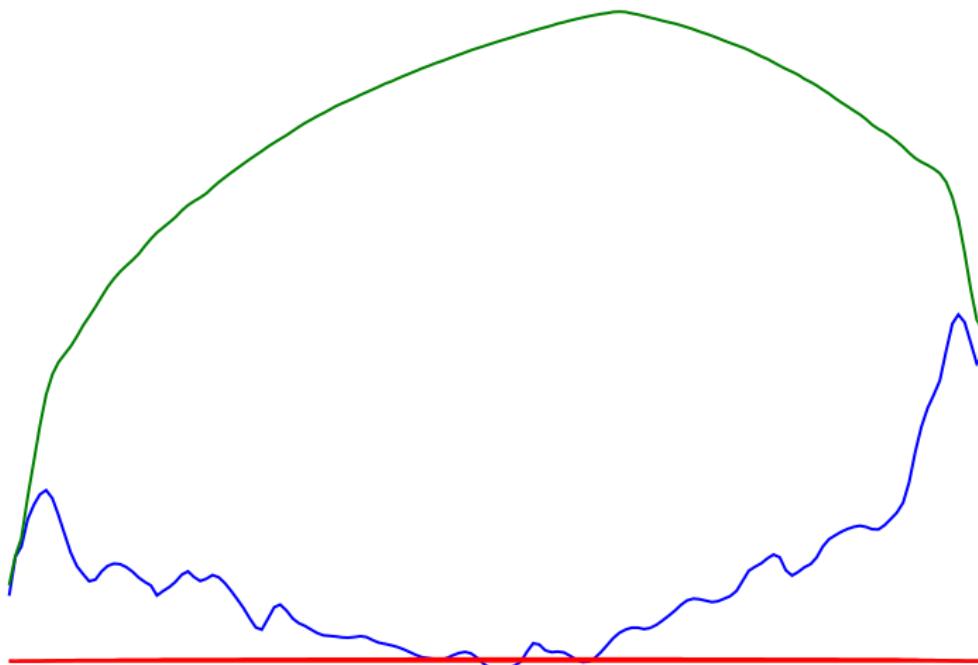
free advice

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”

- ▶ in red is a Greenland cross section
 - at 71° N: width = 760 km, thickness = 3200 m
- ▶ green and blue: standard vertically-exaggerated cross section



flow model I: shallow ice approximation (SIA)

a model which applies to

- ▶ small depth-to-width ratio (“shallow”) grounded ice sheets
- ▶ for *not* too rough/steep bed topography
- ▶ when flow is *not* dominated by sliding at the base



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

SIA model equations

- ▶ simple slogan: *the SIA uses the formulas from slab-on-a-slope*
 - better explanation: expand the Stokes equations in aspect ratio
- ▶ shear stress approximation:

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

- ▶ then horizontal velocity $\mathbf{u} = (u, v)$ satisfies 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 (non-sliding case):

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

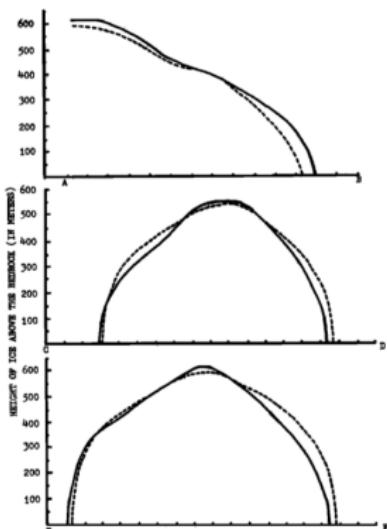
- ▶ add mass continuity: $H_t = M - (\bar{\mathbf{u}} H)_x$

SIA thickness equation

- ▶ thus the non-sliding, isothermal SIA for thickness evolution:

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

- H is ice thickness, h is ice surface elevation
 - ... also b is bed elevation and $h = H + b$
 - M combines surface and basal mass (im)balance
 - $\Gamma = 2A(\rho g)^n / (n+2)$ is a constant
- ▶ numerically solve (1) and you have a usable model for ... *the Barnes ice cap*, a boring ice sheet (Mahaffy, 1976)



good questions:

how to solve (1) numerically?

how to *think* about it?

heat equation

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

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

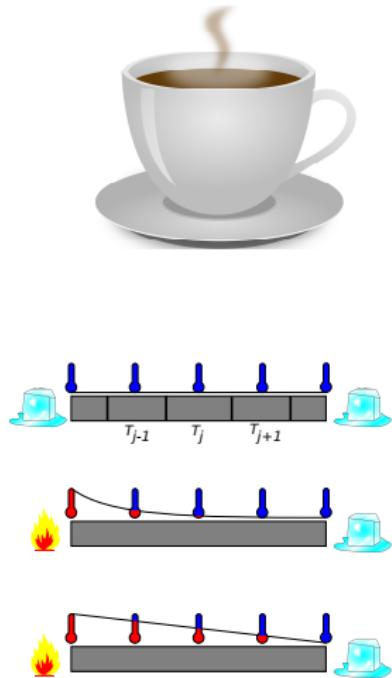
where T is object temperature and K relates to material and geometry of object

- ▶ Newton's law for segments of a rod has T_{ambient} as neighbor average:

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

- ▶ the limit as segments shrink is the *heat equation*:

$$T_t = DT_{xx}$$



major analogy: SIA versus 2D heat equation

- ▶ general 2D 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 geometry
- ▶ concerns about 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 finite differences

- ▶ numerical schemes for heat equation are good start for SIA
- ▶ only *finite difference* (FD) schemes in these slides
- ▶ first, for differentiable $f(x)$ *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 for FD schemes*: combine expressions like these to approximate derivatives from the values of functions on a grid

partial derivatives

- ▶ we want FD approximations of partial derivatives
- ▶ for example, with any function $u = u(t, x)$:

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

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

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

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

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

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

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

explicit scheme for heat equation

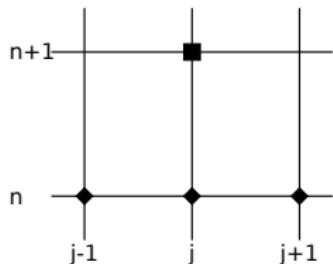
- ▶ consider 1D heat equation $T_t = DT_{xx}$
- ▶ thus this is approximately true:

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

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

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

- ▶ “stencil” at right →



explicit scheme for heat equation

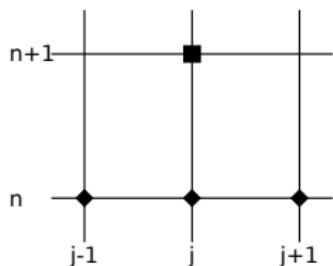
- ▶ consider 1D heat equation $T_t = DT_{xx}$
- ▶ thus this is approximately true:

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

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

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

- ▶ “stencil” at right →



explicit scheme in 2D

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

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

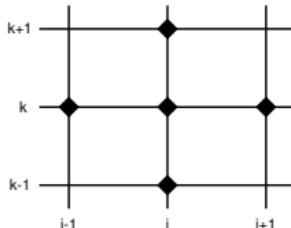
- ▶ again: $T_{jk}^n \approx T(t_n, x_j, y_k)$
- ▶ the 2D explicit scheme is

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

- ▶ which becomes a computable iteration:

$$T_{jk}^{n+1} = T_{jk}^n + \mu_x (T_{j+1,k}^n - 2T_{jk}^n + T_{j-1,k}^n) + \mu_y (T_{j,k+1}^n - 2T_{jk}^n + T_{j,k-1}^n)$$

where $\mu_x = D\Delta t / \Delta x^2$ and $\mu_y = D\Delta t / \Delta y^2$



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

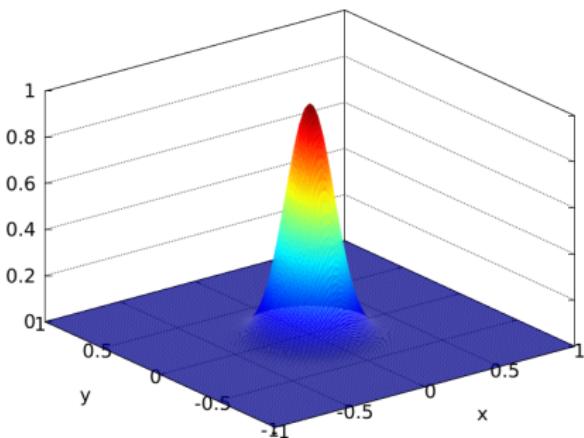
- ▶ solves $T_t = D(T_{xx} + T_{yy})$ on square $-1 < x < 1, -1 < y < 1$
- ▶ example uses initial condition $T_0(x, y) = e^{-30r^2}$
- ▶ code uses “colon notation” to remove loops (over space)

the look of success

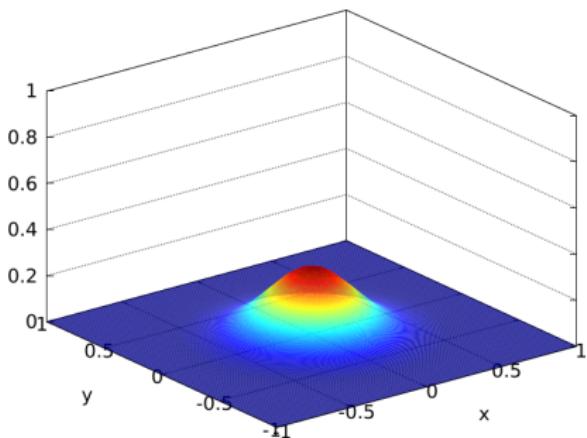
► » `heat(1.0, 30, 30, 0.001, 20)`

solves $T_t = D(T_{xx} + T_{yy})$ for $T(t, x, y)$ with $D = 1$, 30×30 grid, $\Delta t = 0.001$, and $N = 20$ time steps

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

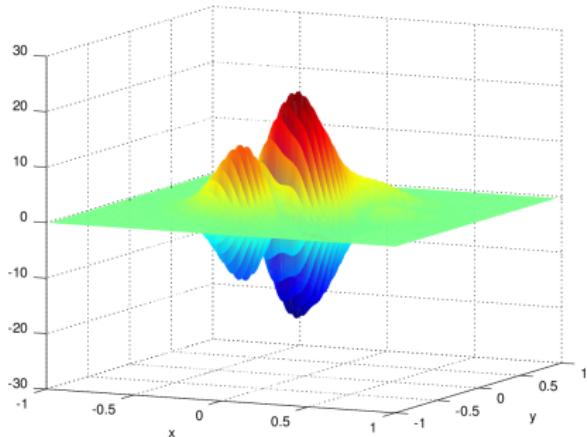
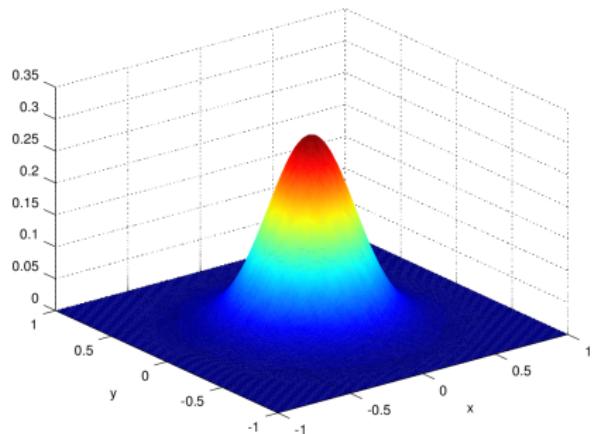


approximate solution $T(0.02, x, y)$



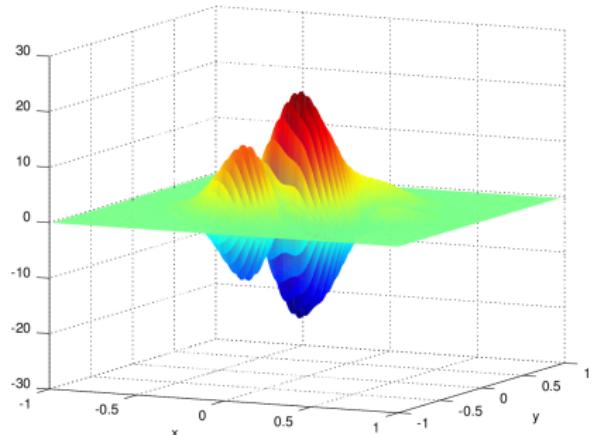
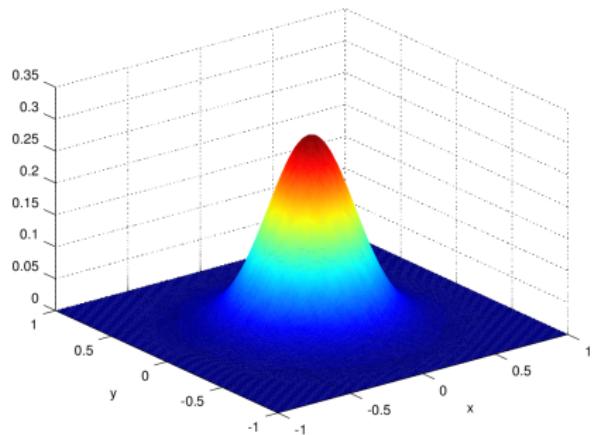
the look of instability

results from solving $T_t = D(T_{xx} + T_{yy})$ on the *same* space grid and at the *same* time, but with slightly-different time steps:



the look of instability

results from solving $T_t = D(T_{xx} + T_{yy})$ on the *same* space grid and at the *same* time, 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

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

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

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

- this condition is a sufficient *stability criterion*
- ▶ averaging is stable because averaged wiggles are smaller than wiggles
- ▶ instability is because *the time step is 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
```

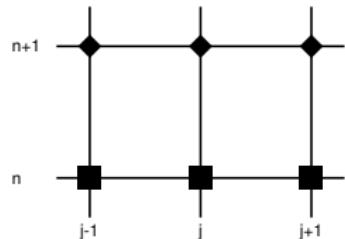
heatadapt.m

- ▶ 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:

*...forget about small efficiencies ...
premature optimization is the root of all evil*

 - I've followed this advice: Bueler et al. (2005) and PISM use adaptive explicit, Bueler (2016) used implicit

variable diffusivity and staggered grids

- ▶ since SIA has diffusivity which varies in space ...
- ▶ consider the generalized diffusion/heat equation:

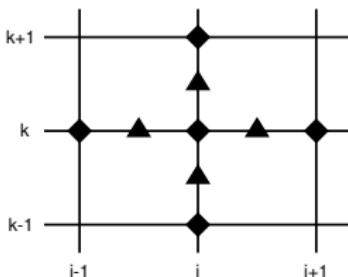
$$T_t = F + \nabla \cdot (D(x, y) \nabla (T + b))$$

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

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

where $X = T + b$

- ▶ in stencil at right →
 - diamonds: T, b
 - 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 $T_t = \nabla \cdot (D(x,y) \nabla(T + b))$
- ▶ user supplies diffusivity on staggered grid

interruption: verification

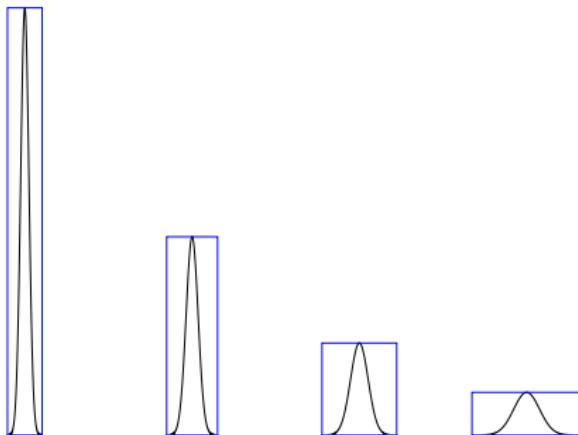
- ▶ how do you make sure your *implemented* numerical ice flow code is correct?
 - *technique 1 = infallibility*: don't make any mistakes
 - *technique 2 = intercomparison*: compare your model with others, and hope that the outliers are the ones with errors
 - *technique 3 = verification*: build-in a comparison to an exact solution, and actually measure the numerical error
- ▶ where to get exact solutions for ice flow models?
 - textbooks: Greve and Blatter (2009), van der Veen (2013)
 - 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)

the Green's function of the heat equation

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

$$T_t = DT_{xx}$$

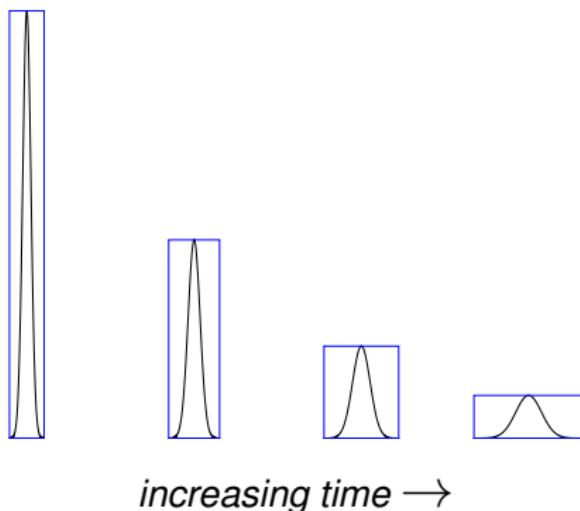
- ▶ many exact solutions are known!
- ▶ Green's function solution shown below
 - a.k.a. "fundamental solution" or "heat kernel"
- ▶ starts at time $t = 0$ with a "delta function" $T(0, x) = \delta_0(x)$
- ▶ then it spreads out over time



increasing time →

the Green's function of the heat equation 2

- ▶ the Green's function can be found by a method which generalizes to the SIA: 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



similarity solutions

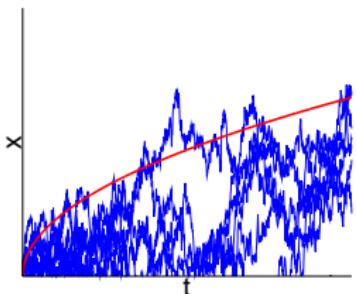
- ▶ “similarity” variables for the 1D heat equation are

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

- ▶ derive from ODE: $\phi(s) = C e^{-s^2/(4D)}$
- ▶ thus the Green's function of heat equation in 1D is

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

- ▶ *a tangent?*: Einstein (1905) discovered 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
 - for almost 20 years no one cares, and P. Halfar leaves the scene
- ▶ Halfar's 2D solution for Glen flow law with $n = 3$ has scalings

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

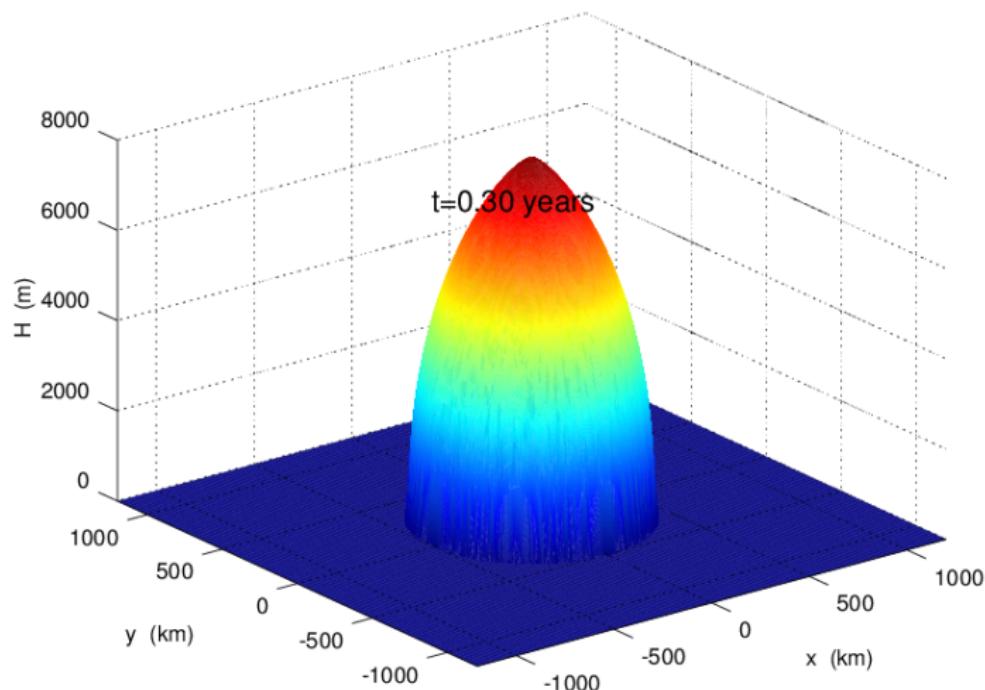
and a simple invariant shape function $\phi(s)$

- ▶ immediate conclusion (**movie follows**): the diffusion of ice really slows down as the shape flattens out!

Halfar solution: the movie

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

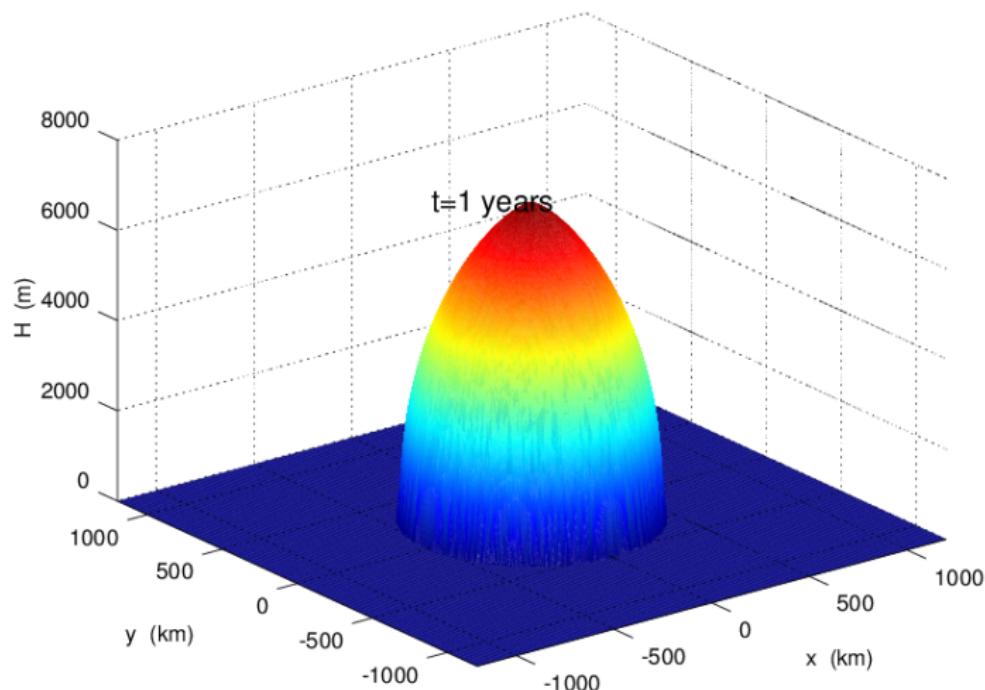
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

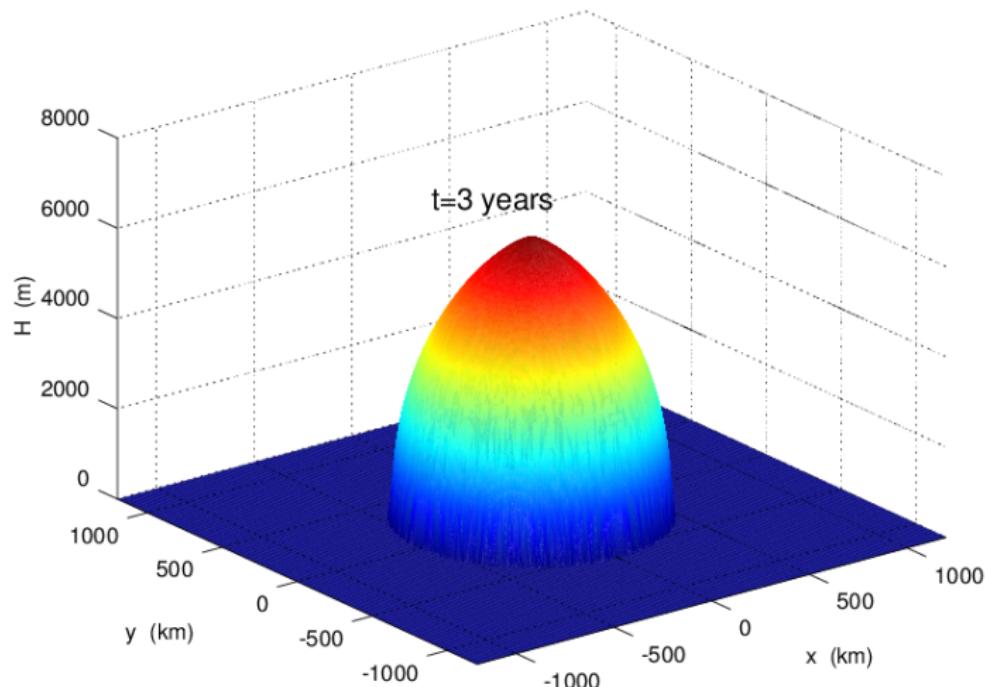
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

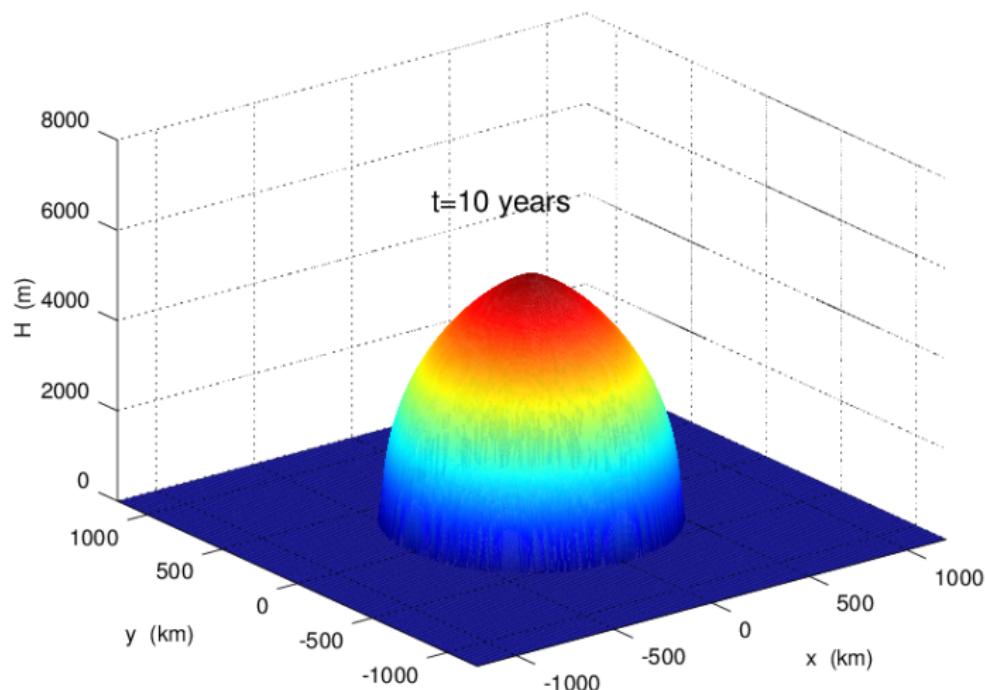
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

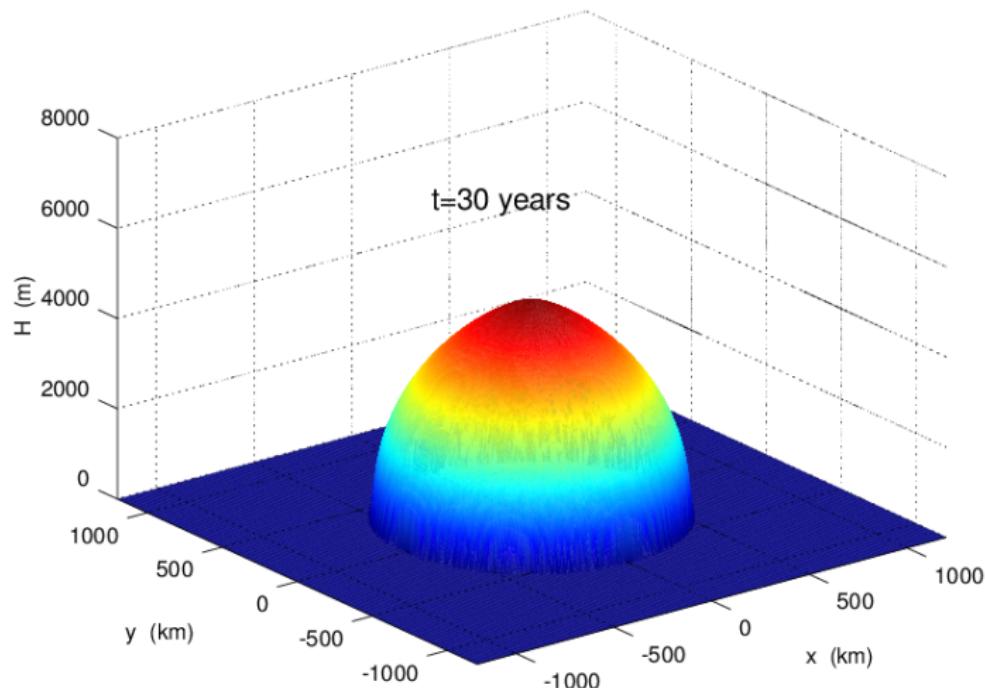
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

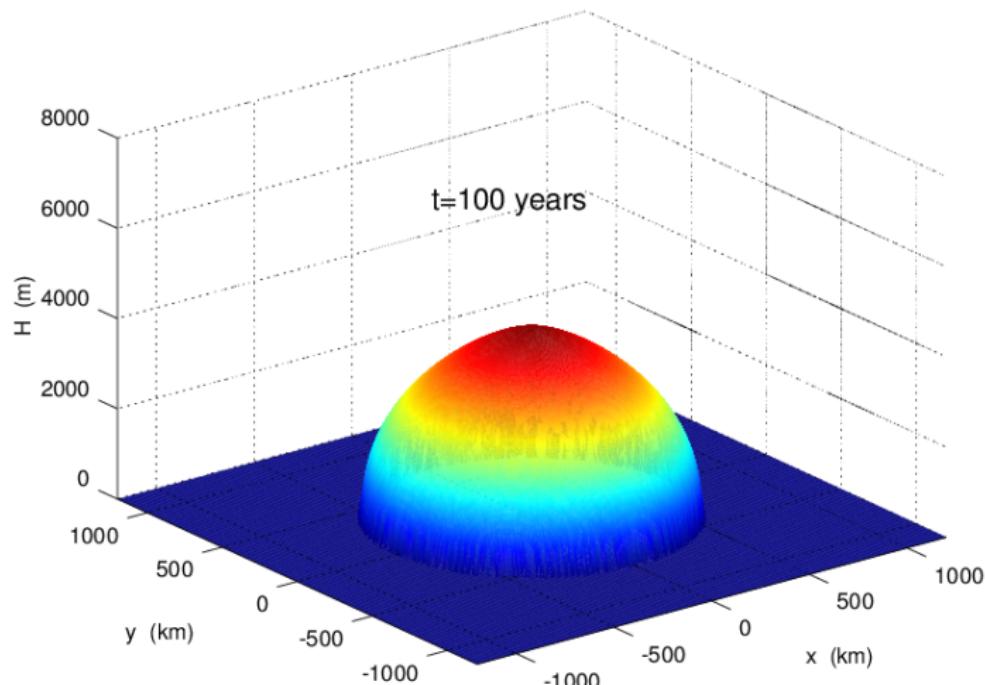
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

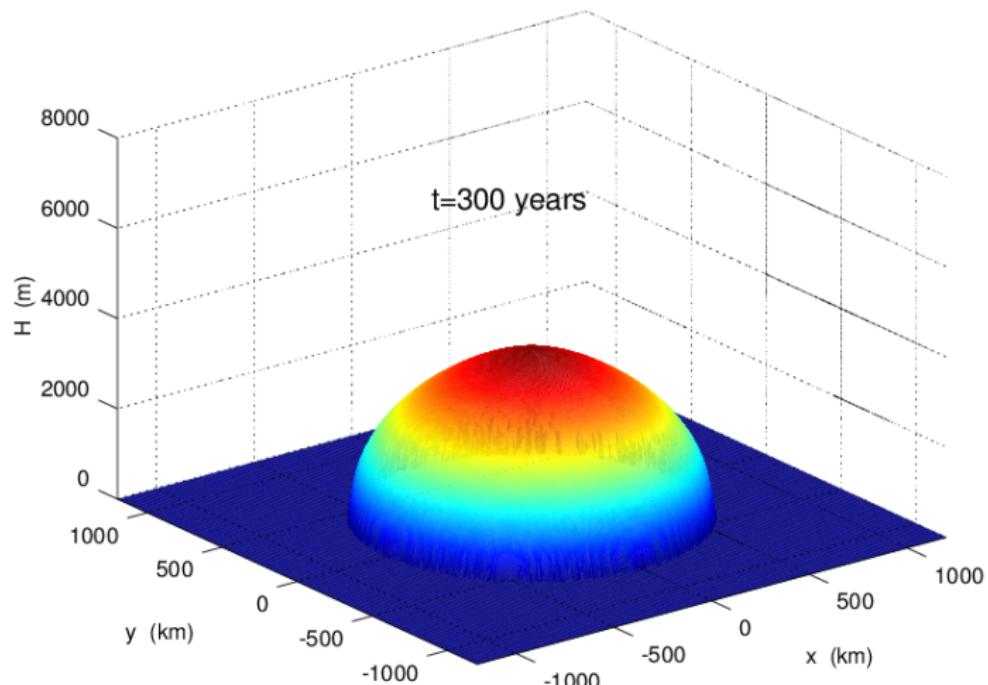
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

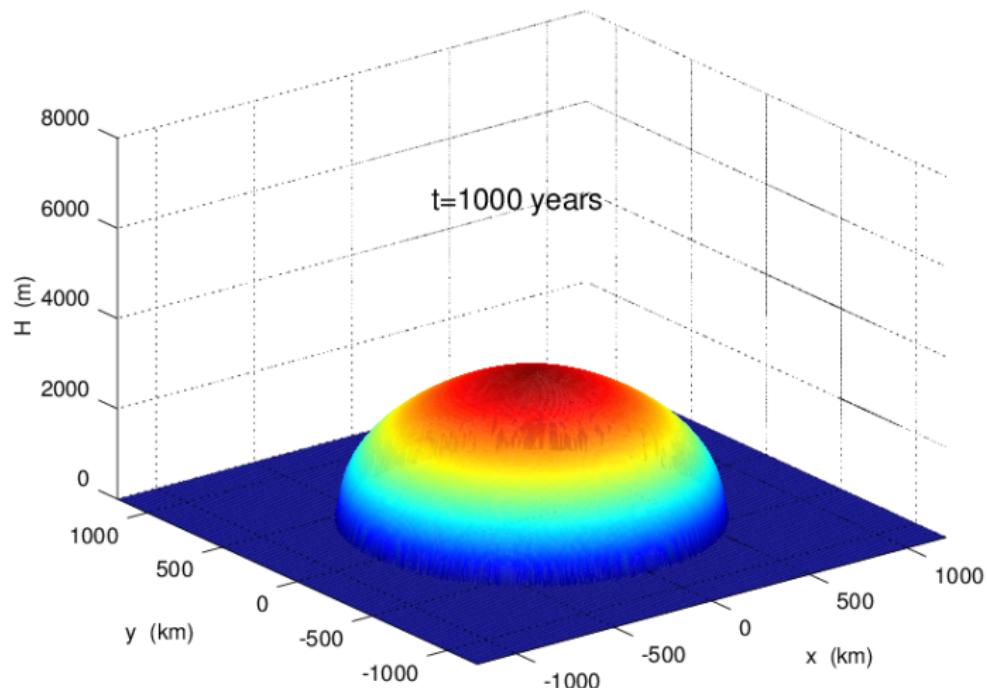
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

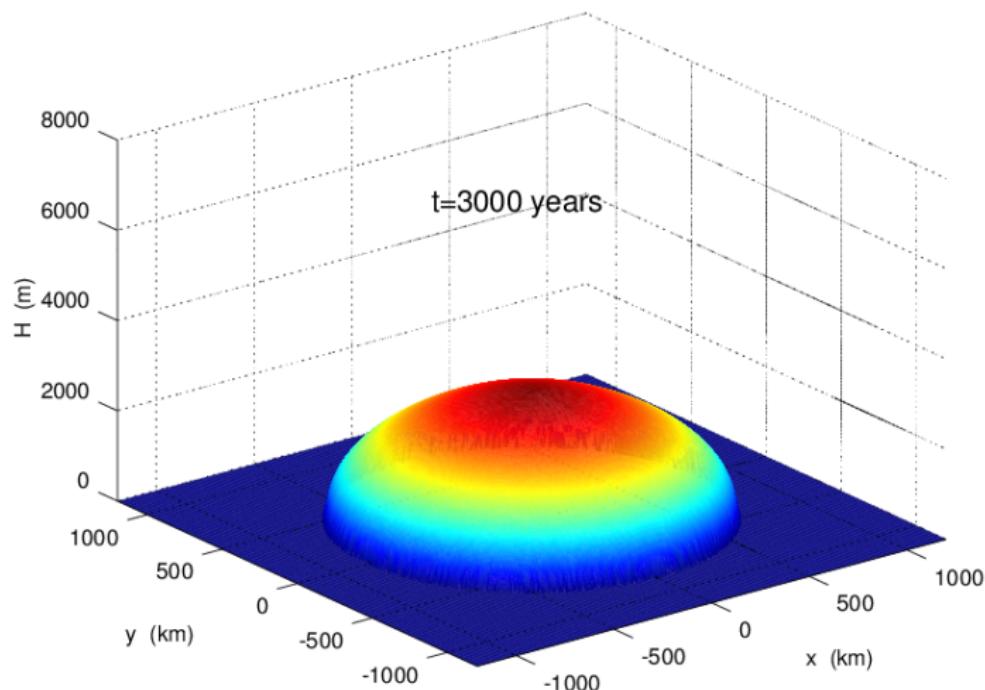
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

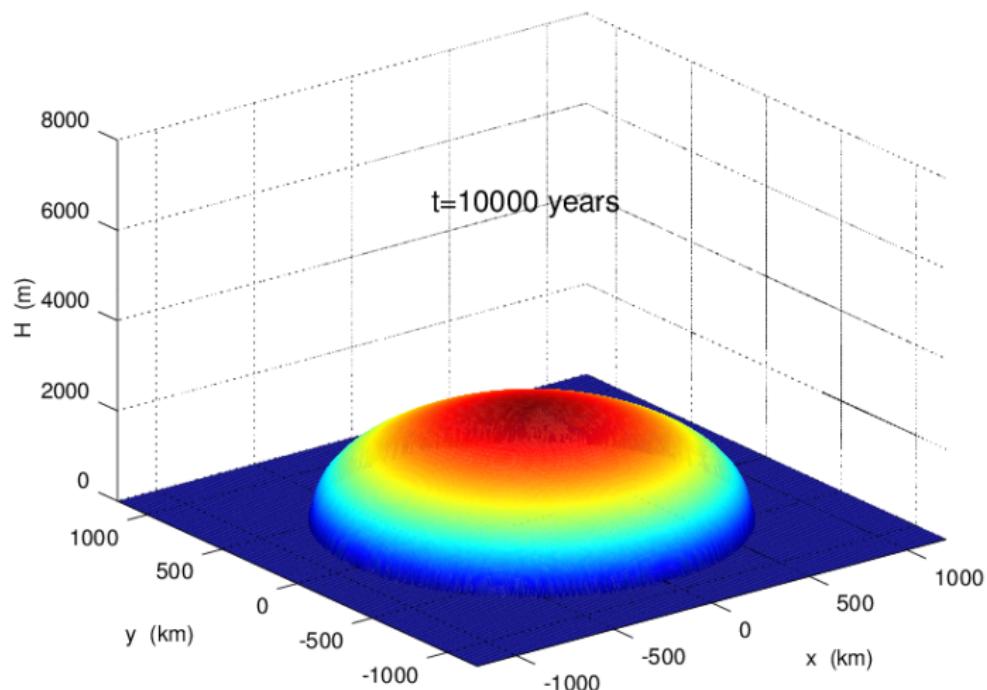
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

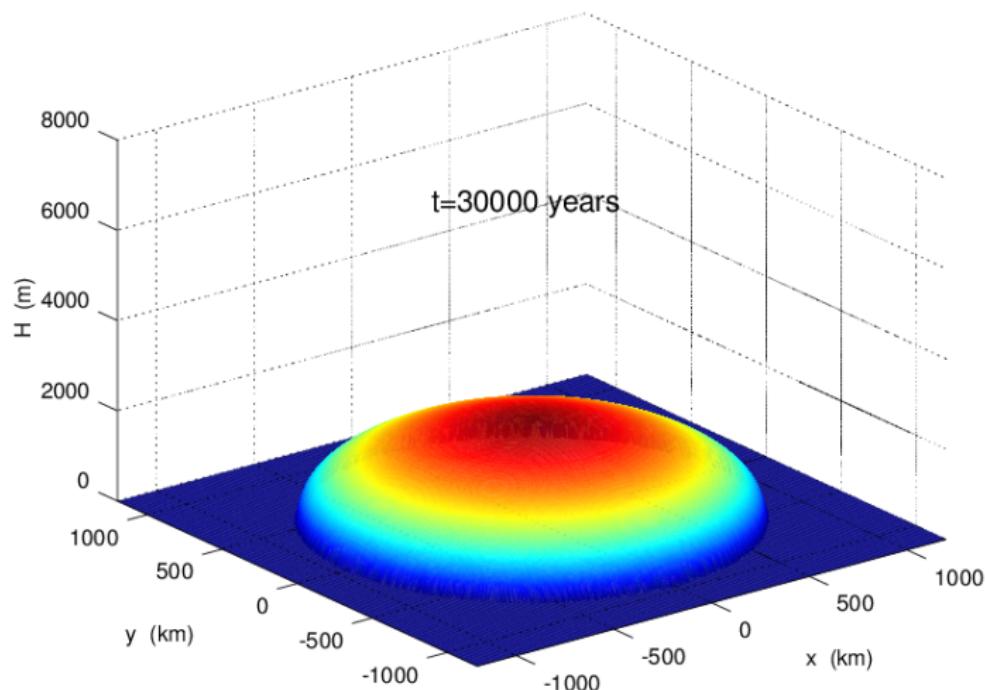
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

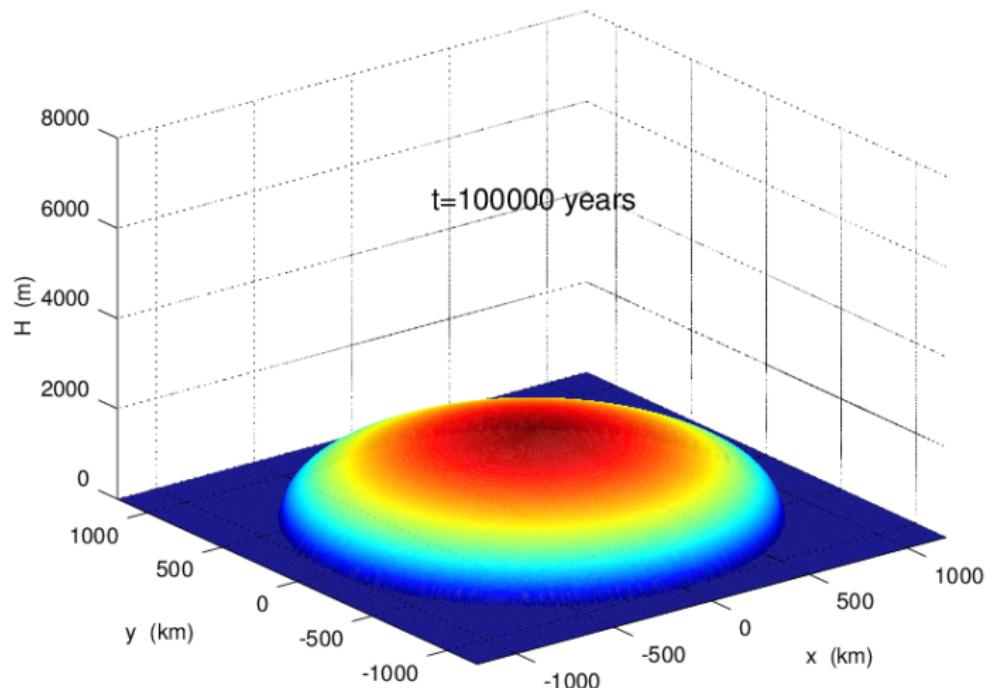
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

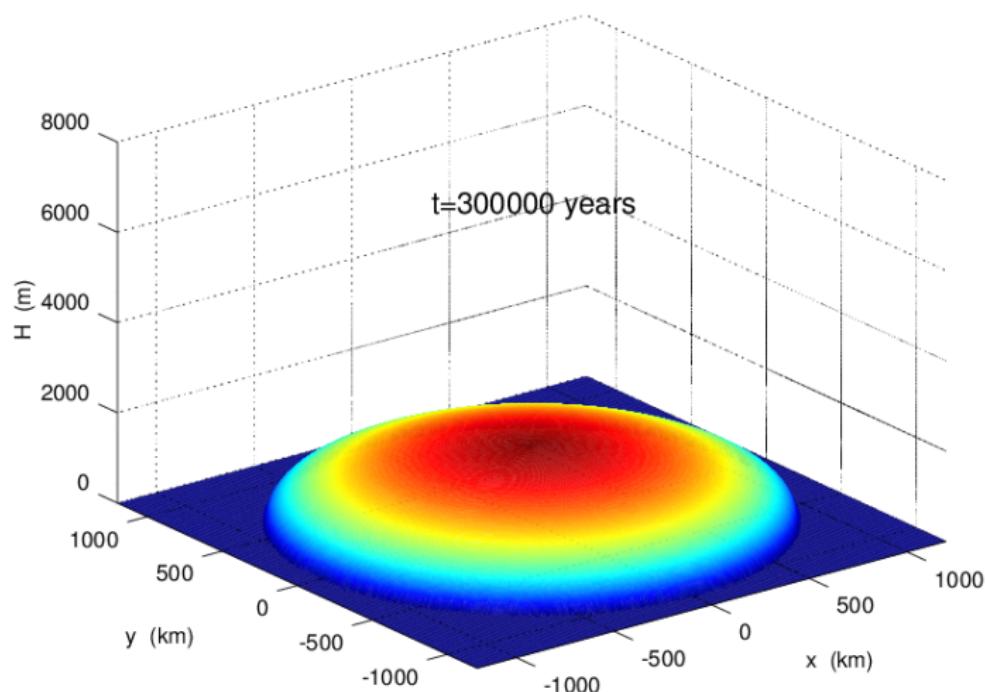
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

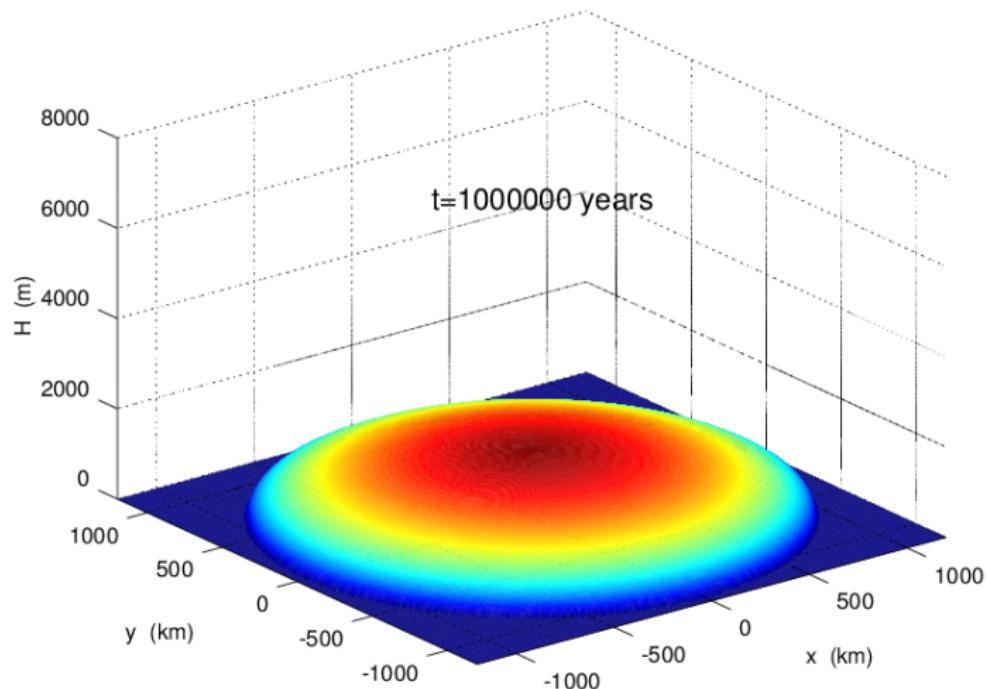
nearly equally-spaced in *logarithmic* time



Halfar solution: the movie

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

nearly equally-spaced in *logarithmic* time



Halfar solution: 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!
 - it finally appears in a textbook: van der Veen (2013)

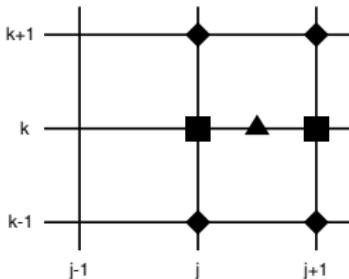
is the Halfar solution useful for modelling?

- ▶ 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
- ▶ direct sampling says: yup!

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; Hindmarsh and Payne 1996
- ▶ all schemes seek accurate diffusivity by:
 - averaging thickness H
 - differencing surface elevation h
 - in a “balanced” way
- ▶ Mahaffy stencil is compact →



SIA implementation: flat bed case

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

g = 9.81;      rho = 910.0;      secpера = 31556926;
A = 1.0e-16/secpера;      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(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

- ▶ run siaflat.m using Halfar solution for verification:

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

- ▶ caution ... how much more effort is halving the grid spacing?

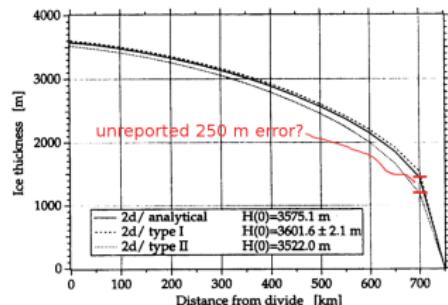
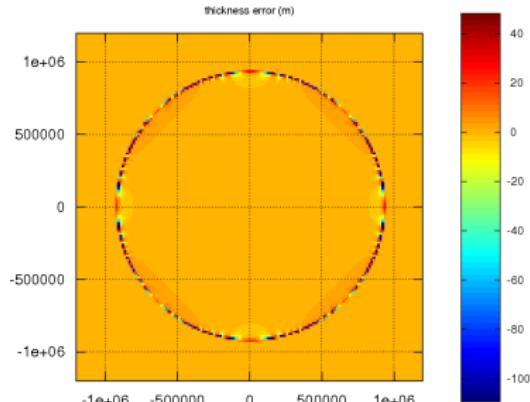
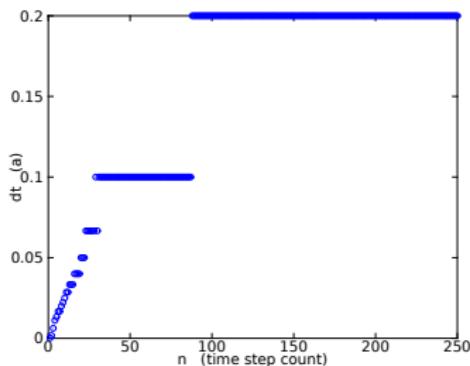
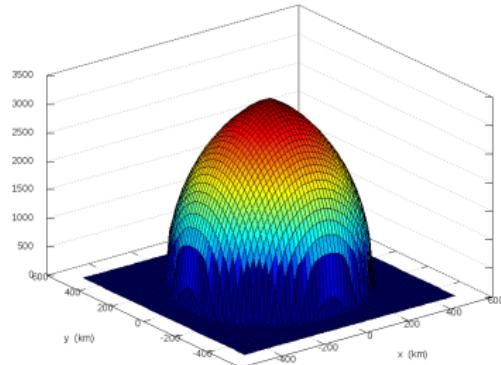
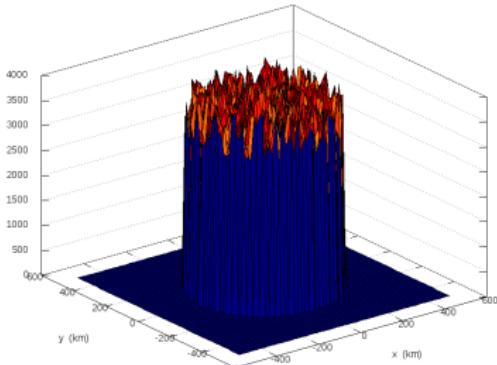


figure 2 in Huybrechts et al. (1996)

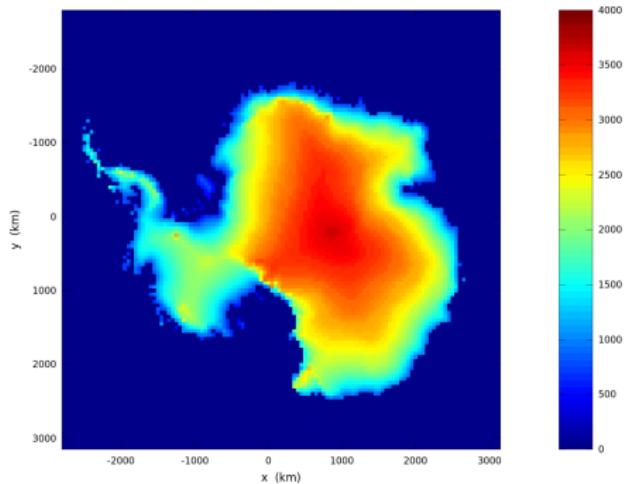
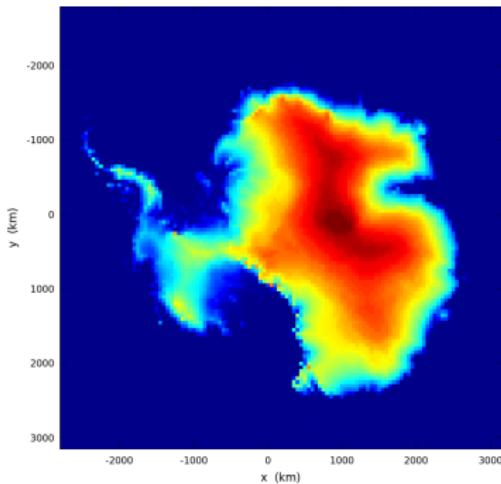
demonstrate robustness

- ▶ `roughice.m` sets-up the nasty initial state at left
- ▶ calls `siaflat.m` to evolve it for 50 years . . . adaptive time steps!
- ▶ get 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
- ▶ makes a good exercise . . . only add 10 new lines of code
- ▶ 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

free advice

three equations for geometry change

- ▶ let a be the climatic (surface) mass balance function
 - $a > 0$ is accumulation and $a < 0$ is ablation
- ▶ let s be the basal melt rate (mass balance) function
 - $s > 0$ is basal melting and $s < 0$ is freeze-on
- ▶ 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	$H_t = M - \nabla \cdot \mathbf{q}$

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

- namely $z = h(t, x, y)$ and $z = b(t, x, y)$
- surface overhang is not allowed
- most Stokes models make this assumption too



kinematic and mass continuity equations

Q: what does the “most-basic shallow assumption” get you?

A: any two of these three equations imply the third:

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	$H_t = M - \nabla \cdot \mathbf{q}$

► to show the equivalences recall:

- the incompressibility of ice

$$u_x + v_y + w_z = 0$$

- and the Leibniz rule for differentiating integrals

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

► it's an exercise

the mass continuity equation: a summary

- ▶ most ice sheet models use the mass continuity equation and the base kinematical equation
- ▶ regarding how to think about the *mass continuity equation*

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

a summary:

- its character depends on the stress balance
 - it *is* a transport equation
 - ...but it is a diffusion for frozen bed, large scale flows (SIA)
 - ▶ if your fancy Stokes model is not diffusive in this case ... *it's wrong*
 - it is not very diffusive for membrane stresses and low basal resistance (e.g. SSA for ice shelves)
 - additional mass continuity equations are needed for liquid water on surface and base ... with much shorter time scales ... which gets complicated
-
- ▶ there is *not* much helpful theory on the transport problems in glaciology ... maybe you will help find this theory!

standard recipe for ice sheet models

1. use stress balance to compute velocity
 - often: first get (u, v) , then w from incompressibility
 2. somewhere in here do other processes: thermodynamics, basal melt, calving, ...
 3. decide on time-step Δt from diffusivity D *(or: fixed Δt)*
 4. from velocity, surface balance, and base balance do time-step of mass continuity equation to get H_t
 5. update surface elevation: $h \leftarrow h + H_t \Delta t$
 6. repeat at 1.
- this paradigm *is* explicit time stepping of the whole model

Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

shelves and streams

free advice

flow model II: shallow shelf approximation (SSA)

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

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

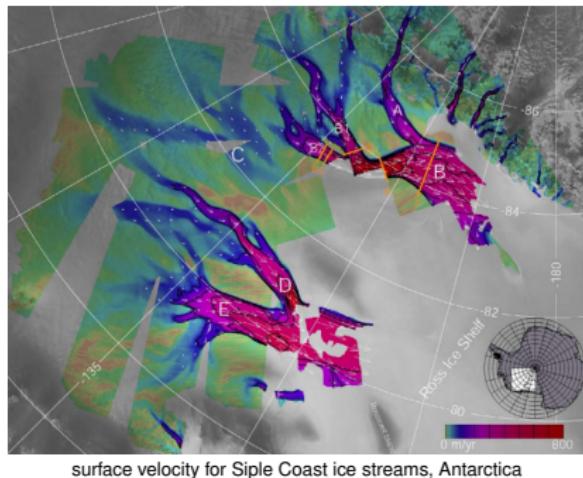


Ekström ice shelf (Hans Grobe)

application of SSA to ice streams

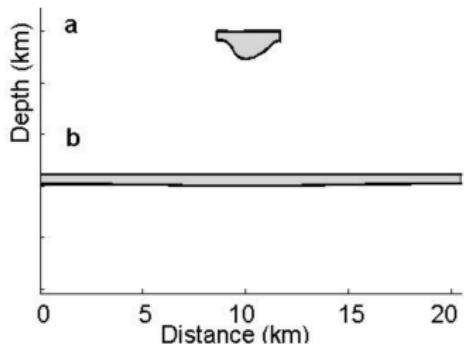
SSA also applies reasonably well to **ice streams**

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



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

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



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

a. outlet glaciers:

- ▶ fast surface speed (up to 10 km a^{-1})
- ▶ uncertain how much is sliding
- ▶ vertical shear in much of ice column
- ▶ *not flat bed topography*
- ▶ soft temperate ice may play big role
- ▶ **simplifications are dubious**

b. ice streams:

- ▶ fast surface speed (up to 1 km a^{-1})
- ▶ very concentrated vertical shear in thin layer near base ("sliding")
- ▶ cold ice nearly down to bed

SSA stress balance equation

- ▶ only plane flow case (“flow line”) shown here
- ▶ the stress balance equation, 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; $C = 0$ in ice shelf
- ▶ the **green term** is driving stress
- ▶ this equation determines velocity u
- ▶ derived originally by Morland (1987), MacAyeal (1989)

once again, good questions:

how to solve (2) numerically?

how to *think* about it?

flotation criterion and grounding line

- ▶ the inequality “ $\rho H < -\rho_w b$ ” is the **flotation criterion**

- assumes $z = 0$ is sea level
- grounding line $x = x_g$ is where $\rho H = -\rho_w b$
- H, u, u_x are all continuous at $x = x_g$

- ▶ surface elevation and driving stress according to flotation:
grounded:

$$\rho H > -\rho_w b$$

$$h = H + b$$

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

floating:

$$\rho H < -\rho_w b$$

$$h = (1 - \rho / \rho_w) H$$

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

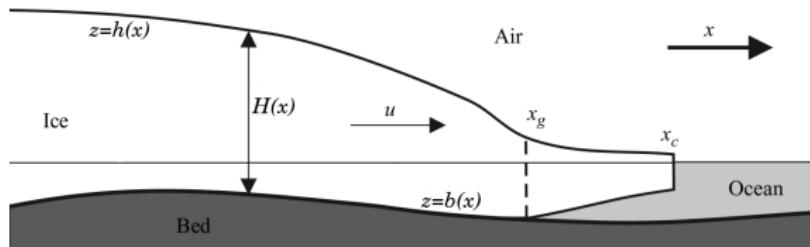
flow line model: from stream to shelf

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

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

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

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

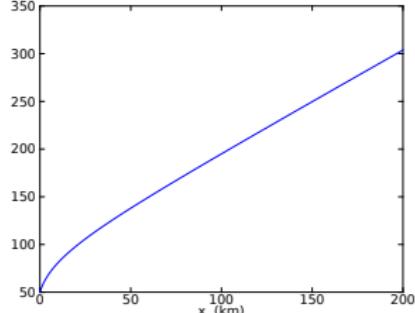
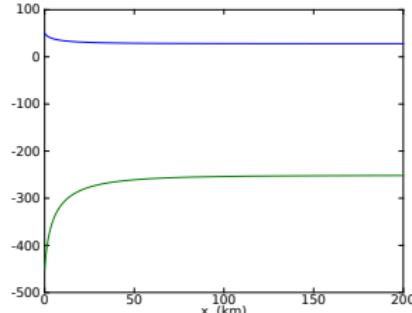


steady ice shelf

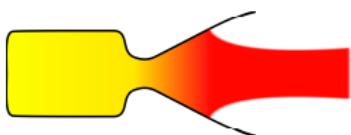
- ▶ I have limited goal here:

describe a steady state, 1D ice shelf

- ▶ *by-hand* result (van der Veen 1983): steady SSA ice shelf found from thickness H_g and velocity u_g at the grounding line
 - plots of the exact formulas:



- an ice shelf is a kind of fluid jet
- ▶ we will use this to
 - understand the SSA better
 - verify a numerical SSA code



numerical SSA stress balance example

I will:

- ▶ fix the ice thickness $H(x)$ from the exact formula
- ▶ find the velocity numerically
- ▶ describe the numerical method for a shelf *or* stream
- ▶ only give a code for an ice shelf

numerics of SSA stress balance

- ▶ the stress balance is a nonlinear equation in the velocity:

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

- thus **iteration is needed**

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

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

- ▶ *simplest iteration idea*: use old $\bar{\nu}$ to get new velocity solution, and repeat until things stop changing

- this is “Picard” iteration (Newton iteration generally superior)
 - from last iterate $u^{(k-1)}$ define

$$W^{(k-1)} = 2\bar{\nu} H = 2A^{-1/n}|u_x^{(k-1)}|^{1/n-1}H$$

- solve for current iterate $u^{(k)}$:

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

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

- ▶ *for floating ice*, assume a uniform strain rate:

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

- ▶ *for grounded ice*, assume ice is held by basal resistance only:

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

abstract the “inner” linear problem

- ▶ abstract problem:

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

- on $0 < x < L$
- at grounding line $x_g = 0$: $u(0) = u_g$
- at calving front $x_c = L$: $u_x(L) = \gamma$

- ▶ an *elliptic* 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

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:

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

- ▶ left-hand boundary condition: $u_1 = u_g$ given
- ▶ right-hand boundary condition:
 - introduce notional point x_{J+2}
 - approximate “ $u_x(L) = \gamma$ ” by

$$\frac{u_{J+2} - u_J}{2\Delta x} = \gamma$$

- use $j = J + 1$ case of $*$ to eliminate u_{J+2} (by-hand)

numerics of the “inner” linear problem 2

- thus iterate of SSA stress balance has form $A\mathbf{x} = \mathbf{b}$:

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

- diagonal entries

$$A_{22} = -(W_{3/2} + W_{5/2} + \alpha_1 \Delta x^2), \quad A_{33} = -(W_{5/2} + W_{7/2} + \alpha_2 \Delta x^2), \quad \dots$$

- special cases in final 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 abstracted linear code

- ▶ first we test the linear code `flowline.m` for the abstract problem
- ▶ test by “manufacturing” solutions
 - see `testflowline.m`
 - result: converges at optimal rate $O(\Delta x^2)$
- ▶ then write Picard iteration loop to create SSA nonlinear solver
 - `ssaflowline.m` on next slide

numerical SSA implementation

```
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 = [(h(2)-h(1))/dx; (h(3:J+1)-h(1:J-1))/(2*dx); (h(J+1)-h(J))/dx];

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;

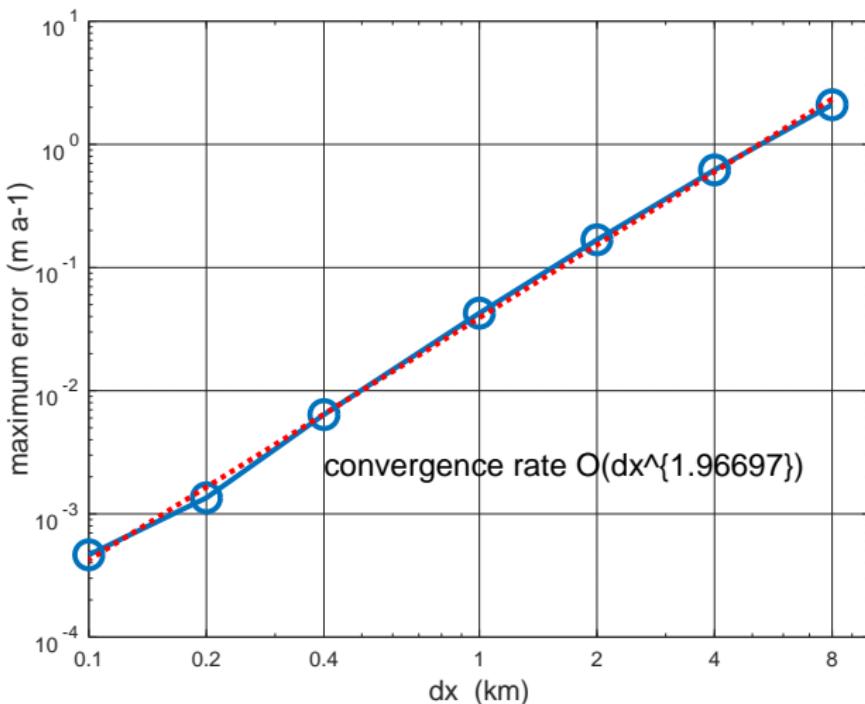
Hstag = 0.5 * (H(1:end-1) + H(2:end));
tol = 1.0e-14;
eps_reg = (1.0 / p.secpera) / p.L;
maxdiff = Inf; W = zeros(J+1,1); iter = 0;
while maxdiff > tol
    uxstag = (u(2:end) - u(1:end-1)) / dx;
    sqr_ux_reg = uxstag.^2 + eps_reg.^2;
    W(1:J) = 2 * p.A^(-1/p.n) * Hstag .* sqr_ux_reg.^(((1/p.n)-1)/2.0);
    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
```

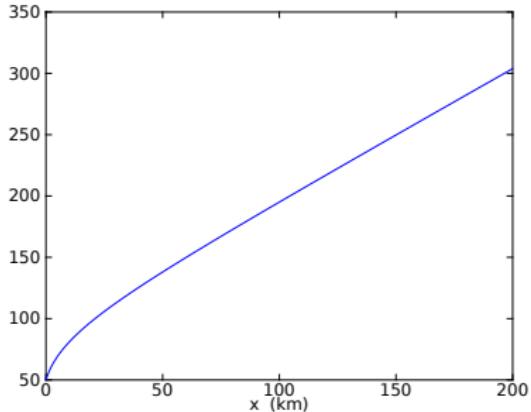
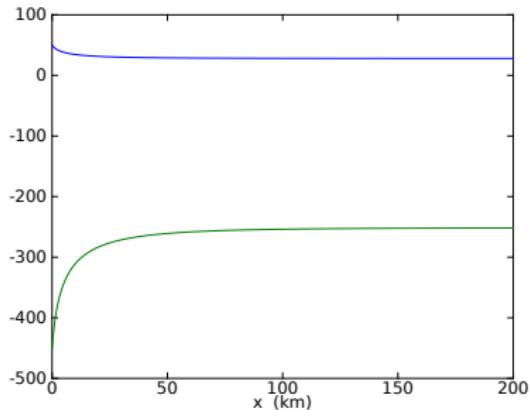
ssaflowline.m

numerical convergence

convergence analysis of `ssaflowline.m`:



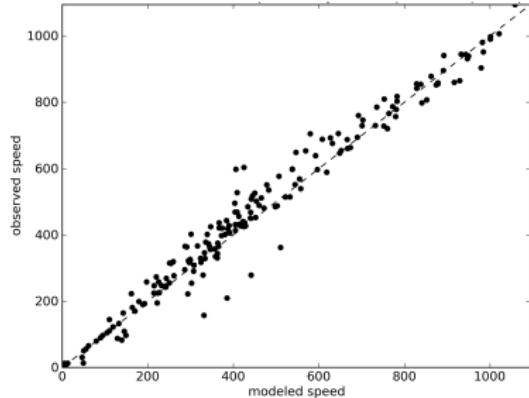
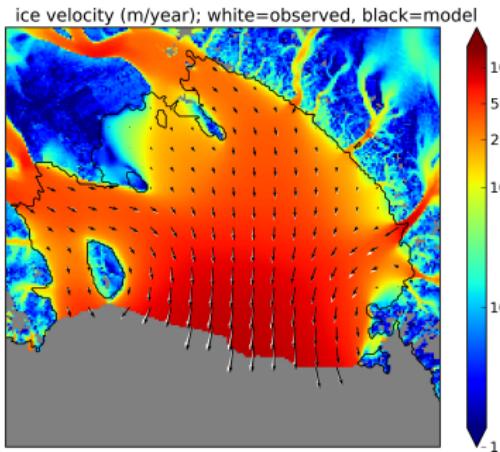
SSA numerical model output



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

numerical ice shelf modeling in 2D

- ▶ flow lines are never very realistic
 - you can add parameterized “side drag”
- ▶ “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
 - ... 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
- ▶ for ice the equations are nonlinear so iteration is necessary
 - at each iteration a sparse matrix “inner” problem is solved
 - give it to a linear algebra package; don’t write it yourself
- ▶ when geometry and boundary stresses are *known*, many modern stress balance solvers do well
 - e.g. ice shelves
 - model shallowness often is *not* the issue
 - see: Aschwanden et al. (2016), *Complex Greenland outlet glacier flow captured*

Outline

introduction: a view from outside glaciology

shallow ice sheets

mass continuity

shelves and streams

free advice

best practices for numerical modeling

- ▶ learn a version-control system git
- ▶ put your project on a public host from the beginning
 - to hell with your secretive advisor github.com
- ▶ modularize your codes
- ▶ test the parts
 - make testing easily repeatable make test
- ▶ if you are stuck:
 - write documentation for what you have
 - go find exact solutions and literature about submodels

- ▶ this thing



does not know what you *intend*, 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 PDEs
 - achieve this through verification with exact solutions