numerical modeling of glaciers

Ed Bueler

June 2024

Porphyry Place, McCarthy, Alaska

version 1.0

Outline

how does the glacier surface move? (hour 1)

- basic mathematics
- some numerics

where does the velocity field come from? (hour 2)

- basic mathematics
- some numerics

online materials

- these are my new slides … please let me know how it works?
- in any case, please ask questions at any time!

online materials

my slides, notes, codes, and materials are hosted at

`github.com/bueler/mccarthy`

these slides: `slides/slides-2024.pdf`

exercises: `slides/exercises-2024.pdf`

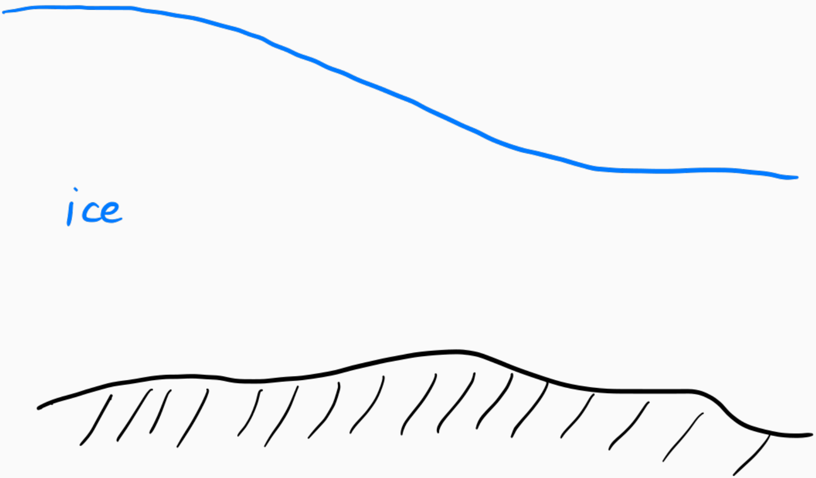
notes: `notes/notes-2024.pdf`

Python codes: `py/`

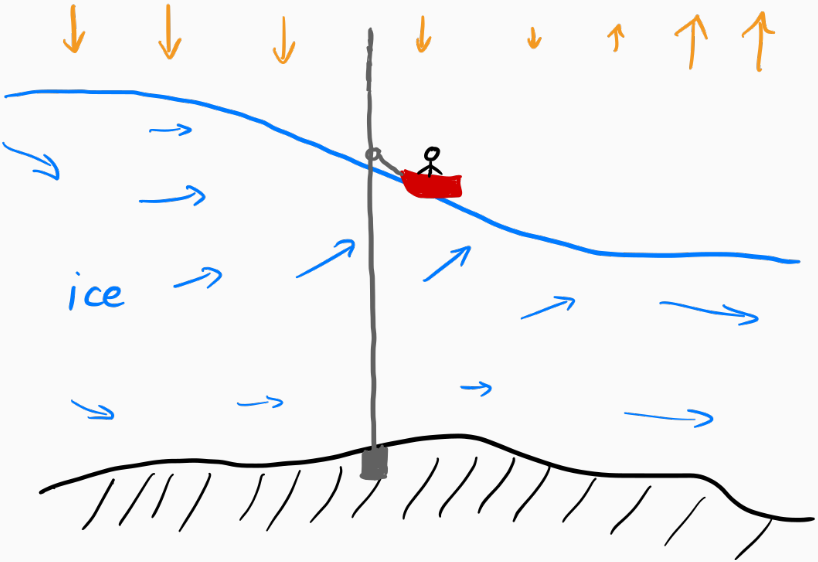
project 13,14 Python codes: `stokes/`

how does the glacier surface move?

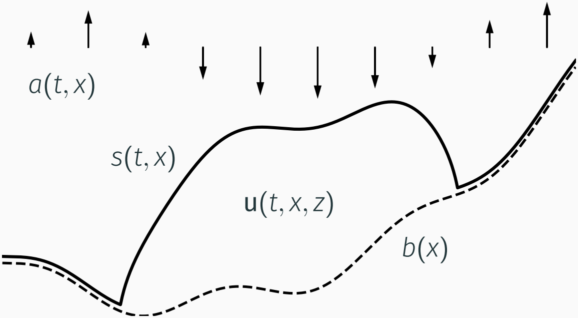
how does the glacier surface move?



how does the glacier surface move?

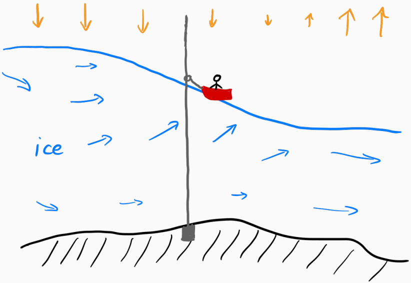


surface motion: notation



- two spatial dimensions (planar), along a flow line
  - x horizontal, z vertical
- $b(x)$: bed elevation (m) *fixed in time!*
- $s(t,x)$: ice surface elevation (m)
- $\mathbf{u}(t,x,z) = (u,w)$: ice velocity ($\mathrm{m\,s^{-1}}$)
- $a(t,x)$: surface mass balance in ice-equivalent units ($\mathrm{m\,s^{-1}}$)

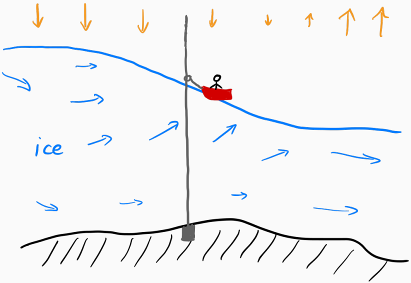
surface kinematical equation (SKE)



$$\frac{\partial s}{\partial t} =$$

the ice surface moves up and down according to

surface kinematical equation (SKE)

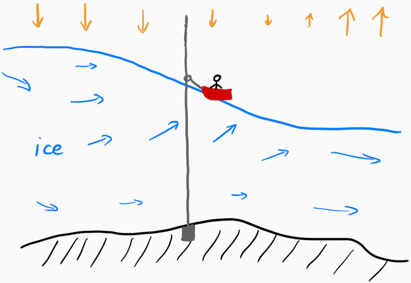


$$\frac{\partial s}{\partial t} = a$$

the ice surface moves up and down according to

- surface mass balance (SMB) *climatic mass balance (ice equiv.)*

surface kinematical equation (SKE)

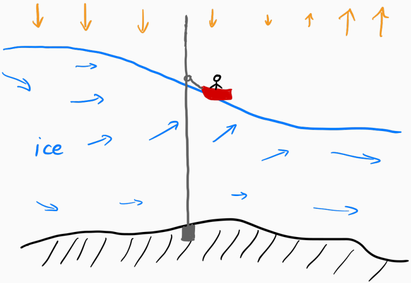


$$\frac{\partial s}{\partial t} = a + w$$

the ice surface moves up and down according to

- surface mass balance (SMB) *climatic mass balance (ice equiv.)*
- vertical component of ice velocity

surface kinematical equation (SKE)

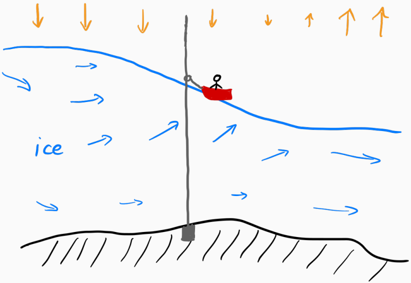


$$\frac{\partial s}{\partial t} = a + w - u\frac{\partial s}{\partial x}$$

the ice surface moves up and down according to

- surface mass balance (SMB) *climatic mass balance (ice equiv.)*
- vertical component of ice velocity
- horizontal component of ice velocity *and surface slope*

surface kinematical equation (SKE) ...alternate form



$$\frac{\partial s}{\partial t} = a + \mathbf{u} \cdot \mathbf{n}_s$$

- recall: $\mathbf{u} = (u, w)$
- $\mathbf{n}_s$ is a vector which points upward and is normal (perpendicular) to the ice surface: $\mathbf{n}_s = \left(-\frac{\partial s}{\partial x}, 1\right)$
- how would you prove the SKE? *one answer: extra slides at end*

plan for hour 1: SKE as a numerical glacier geometry model

- model: the planar surface kinematical equation (SKE)

$$\frac{\partial s}{\partial t} - \mathbf{u} \cdot \mathbf{n}_s = a \qquad \iff \qquad \frac{\partial s}{\partial t} + u \frac{\partial s}{\partial x} = a + w$$

  - assume a, u, w are given functions/values
- numerical modeling goal: use a computer program to *approximately* evolve the glacier surface $z = s(t, x)$ according to the SKE
  - one short Python program in 1D: py/surface1d.py
- to address:
  1. how to compute time-dependent solutions?
  2. accuracy and stability?
  3. how to compute steady state solutions?
  4. practical: debugging, verification, visualization?

plan for hour 1: SKE as a numerical glacier geometry model

- model: the planar surface kinematical equation (SKE)

$$\frac{\partial s}{\partial t} - \mathbf{u} \cdot \mathbf{n}_s = a \qquad \iff \qquad \frac{\partial s}{\partial t} + u \frac{\partial s}{\partial x} = a + w$$

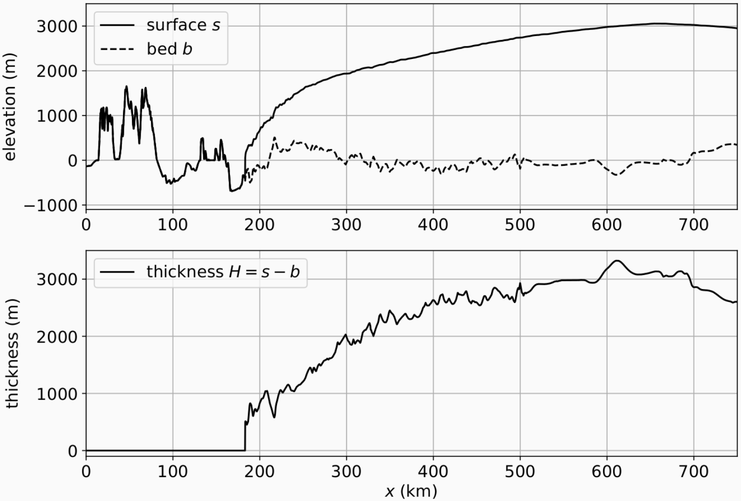
  - assume a, u, w are given functions/values $\quad \leftarrow$ this needs fixing!
- numerical modeling goal: use a computer program to *approximately* evolve the glacier surface $z = s(t, x)$ according to the SKE
  - one short Python program in 1D: py/surface1d.py
- to address:
  1. how to compute time-dependent solutions?
  2. accuracy and stability?
  3. how to compute steady state solutions?
  4. practical: debugging, verification, visualization?

claim: the central object of glacier theory is the surface kinematical equation

$$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} = a + w$$

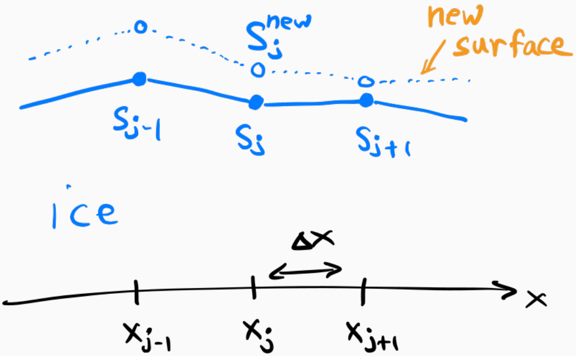
surface smoothness versus thickness smoothness

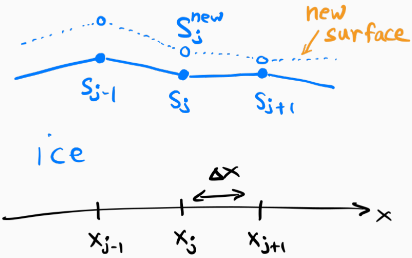
- why not use thickness $H = s - b$ to describe glacier geometry?



[Data from Morlighem et al (2017), and thanks to A. Aschwanden]

discretize the SKE 1



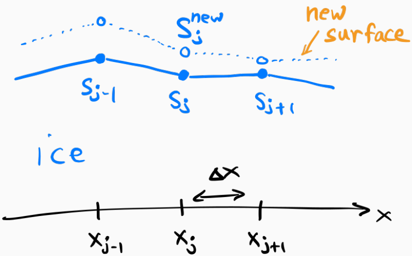


- SKE: $\dfrac{\partial s}{\partial t} + u \dfrac{\partial s}{\partial x} = a + w$
- choose time step $\Delta t > 0$ and grid spacing $\Delta x > 0$
- approximate partial derivatives by finite difference[1] quotients:

$$\frac{\partial s}{\partial t} \approx \frac{s_j^{\text{new}} - s_j}{\Delta t}, \qquad \frac{\partial s}{\partial x} \approx \frac{s_{j+1} - s_{j-1}}{2\Delta x}$$

[1] helpful textbooks include LeVeque [LeV07] and Morton & Mayers [MM05]

discretize the SKE 3



- now the SKE $\left(\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} = a + w\right)$ becomes discrete:

$$\frac{s_j^{\text{new}} - s_j}{\Delta t} + u_j \frac{s_{j+1} - s_{j-1}}{2\Delta x} = a_j + w_j$$

- we can write this as an equation which determines s_j^{new}:

$$s_j^{\text{new}} = s_j - \Delta t\, u_j \frac{s_{j+1} - s_{j-1}}{2\Delta x} + \Delta t(a_j + w_j)$$

upwinding

- however …
- if we implement this “forward-time centered-space” scheme

$$s_j^{\text{new}} = s_j - \Delta t\, u_j \frac{s_{j+1} - s_{j-1}}{2\Delta x} + \Delta t(a_j + w_j) \qquad \leftarrow \textit{bad}$$

  then bad (unstable) things happen! *exercise?*

- instead, it is known that when the “advecting velocity” is positive ($u_j \geq 0$) then the “upwind” version is conditionally stable:

$$s_j^{\text{new}} = s_j - \Delta t\, u_j \frac{s_j - s_{j-1}}{\Delta x} + \Delta t(a_j + w_j) \qquad \leftarrow \textit{useful}$$

  - upwind formula: $\dfrac{\partial s}{\partial x} \approx \dfrac{1}{\Delta x} \begin{cases} (s_j - s_{j-1}), & u_j \geq 0 \\ (s_{j+1} - s_j), & u_j < 0 \end{cases}$
  - the condition for stability is known *(more soon)*

implementation

- this upwind scheme for the SKE (assumes $u_j \geq 0$):

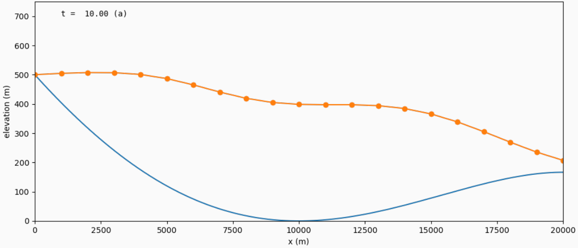
$$s_j^{\text{new}} = s_j - \Delta t\, u_j \frac{s_j - s_{j-1}}{\Delta x} + \Delta t (a_j + w_j)$$

  becomes a Python function:

```
def explicitstep(s, x, dt):
    dx = x[1] - x[0]
    snew = s.copy()
    snew[1:] -= dt * u(x[1:]) * (s[1:] - s[:-1]) / dx  # assumes u >= 0
    snew[1:] += dt * (a(x[1:]) + w(x[1:]))
    return snew
```

  - `x`, `s`, `snew` are 1D NumPy arrays
  - separate functions define $a(x)$, $u(x)$, $w(x)$
  - `x` is equally-spaced
  - all indices except zero: $j =$ `1:`
  - note that `snew[0] = s[0]` is never changed; this is a boundary condition

live demo: evolution of glacier surface



live demo

- run py/surface1d.py:

```
$ python3 surface1d.py
$ eog output/frame*.png          # any image viewer
```

- the following example modifications reveal stability issues:
  1. increase velocity: $u(x) \to 2.5\, u(x)$
  2. increase resolution: $\Delta x = 1000$ m $\to$ $\Delta x = 400$ m
  3. lengthen time-steps: $\Delta t = 1$ a $\to$ $\Delta t = 2.5$ a

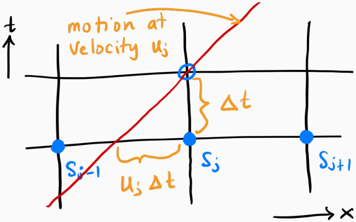
condition for time-stepping stability

- SKE: $$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} = a + w$$
- if horizontal velocity u is given then information about the surface travels at speed u
- from this (t, x) grid picture, the upwind scheme can only be stable if

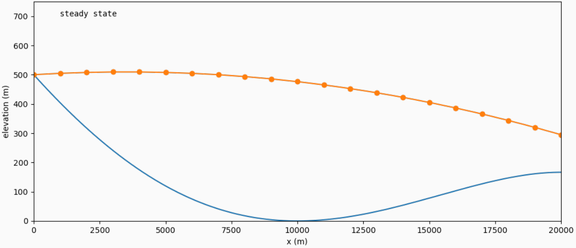
$$|u_j|\Delta t \leq \Delta x$$

  for every j
    - observed by Courant, Friedrichs, and Lewy (1928)
- rearranged, this is the CFL condition on the time step:

$$\Delta t \leq \frac{\Delta x}{\max |u_j|}$$



live demo: steady state of a glacier surface



- SKE in steady-state ($\frac{\partial s}{\partial t} = 0$): $$u\frac{\partial s}{\partial x} = a + w$$

live demo

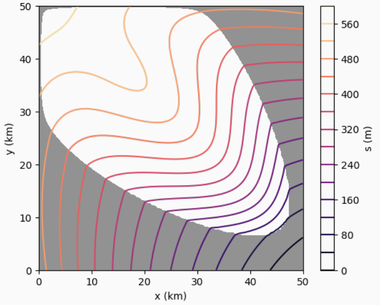
- same run as before, but view steady-state output:

```
$ eog output/steady.png
```

- the following modification reveals an issue:
    1. double SMB: $a(x) \;\rightarrow\; 2a(x)$

teaser: 2D steady state of a glacier surface

- this example uses Firedrake
  - *talk to me?*
- code: py/surface2d.py
- result: $\to$



- solves 2D steady-state SKE:

$$u\frac{\partial s}{\partial x} + v\frac{\partial s}{\partial y} = a + w$$

$$\text{or} \qquad (u, v)\cdot \nabla s = a + w$$

$$\text{or} \qquad -\mathbf{u}\cdot\mathbf{n}_s = a$$

  - where $\mathbf{n}_s = (-\frac{\partial s}{\partial x}, -\frac{\partial s}{\partial y}, 1) = (-\nabla s, 1)$
- but subject to $s \geq b$
- compare time-dependent SKE in 2D: $\displaystyle \frac{\partial s}{\partial t} - \mathbf{u}\cdot\mathbf{n}_s = a$

some questions when solving the SKE

MAJOR modeling questions

1. whence the SMB a? *Regine?*
2. whence the surface velocity u, w? *2nd hour!*

mathematical questions

3. is the time-dependent SKE well-posed for (various) given-field, initial condition, & boundary condition assumptions?
4. is the steady-state SKE well-posed for (various) … assumptions?
5. how do we understand admissibility ($s \geq b$) in the theory?

numerical questions

6. when the surface velocity u, w comes from a stress-balance submodel, is the upwind scheme still conditionally stable?
7. are there better explicit schemes?
8. how does one enforce admissibility the best way?
9. is it better to use an implicit scheme?

where does the velocity field come from?

ice in glaciers is an atypical fluid (relative to textbooks)

- if the ice were
  - faster-moving than it actually is, and
  - linearly-viscous

  then it would be a "typical" fluid like liquid water

- for liquid water one typically uses the incompressible Navier-Stokes model:

$$
\begin{aligned}
\nabla \cdot \mathbf{u} &= 0 && \textit{incompressibility} \\
\rho\left(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}\right) &= -\nabla p + \nabla\cdot\tau_{ij} + \rho \mathbf{g} && \textit{stress balance} \\
2\nu D\mathbf{u}_{ij} &= \tau_{ij} && \textit{flow law}
\end{aligned}
$$

  - note that the stress balance equation is "$ma = F$"

glaciology as computational fluid dynamics

- numerical glacier modelling is first-class computational fluid dynamics (CFD)
  - it's large-scale like atmosphere and ocean
  - … but it is weird relative to those
- consider what makes atmosphere/ocean flow exciting:
  - turbulence
  - convection
  - coriolis force
  - density stratification
- none of the above list is relevant to ice flow!
- CFD textbooks are of limited utility to glaciologists
- how do we handle slow, cold, stiff, laminar, and inert ice?

ice is a slow, shear-thinning fluid

- ice fluid is *slow* and *non-Newtonian*
    - “slow” is a technical term:

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

    - ice is non-Newtonian in a “shear-thinning” way
        - “non-Newtonian” means viscosity ν is not constant, but instead depends on the fluid state
        - specifically: higher strain rates means lower viscosity

- the standard model is (Glen-Steinemann-Nye power law) Stokes:

$$\begin{aligned} \nabla\cdot\mathbf{u} &= 0 && \textit{incompressibility} \\ 0 &= -\nabla p + \nabla\cdot\tau_{ij} + \rho\,\boldsymbol{g} && \textit{stress balance} \\ D\mathbf{u}_{ij} &= A\tau^{n-1}\tau_{ij} && \textit{flow law} \end{aligned}$$

    - ρ is ice density, $\boldsymbol{g}$ is acceleration of gravity, A is ice softness
    - the standard *time-dependent* model is SKE + Stokes

“slow” means no memory of velocity/momentum

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

$$
\begin{aligned}
\nabla \cdot \mathbf{u} &= 0 \\
0 &= -\nabla p + \nabla \cdot \tau_{ij} + \rho \mathbf{g} \\
D\mathbf{u}_{ij} &= A\tau^{n-1}\tau_{ij}
\end{aligned}
$$

- in the standard model, the velocity and pressure fields of a glacier are instantaneous functions of the glacier geometry and its boundary stresses
- thus a time-stepping ice sheet code can (and must) recompute the velocity field at every time step
    - but velocity from the previous time step is *not* required, unlike (e.g.) a weather model
    - velocity is a “diagnostic” output, not needed for (re)starting the model

plane flow Stokes

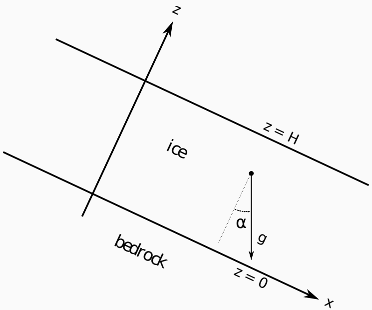
- return to a x, z plane ...
- the $n = 3$ Glen-law Stokes equations say:

$$\begin{aligned}
u_x + w_z &= 0 && \textit{incompressibility} \\
0 &= -p_x \tau_{11,x} + \tau_{13,z} && \textit{stress balance } (x) \\
0 &= -p_z \tau_{13,x} - \tau_{11,z} - \rho g && \textit{stress balance } (z) \\
u_x &= A\tau^2 \tau_{11} && \textit{flow law (diagonal)} \\
\frac{1}{2}\left(u_z + w_x\right) &= A\tau^2 \tau_{13} && \textit{flow law (off-diagonal)}
\end{aligned}$$

  - *notation*: subscripts x, z denote partial derivatives
  - *notation*: τ_{11} and τ_{33} are (deviatoric) longitudinal stresses, τ_{13} is the "vertical" shear stress
  - *observation*: $\tau_{33} = -\tau_{11}$ *why?*
- we have 5 equations in 5 unknowns $(u, w, p, \tau_{11}, \tau_{13})$
- this is complicated enough ... try a simplified situation?

slab-on-a-slope

- suppose constant thickness and constant bedrock slope with angle α
- rotate the coordinates so x is bed-parallel
- new/revised equations:



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

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

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

- for slab-on-a-slope there is *no variation in x*: $\partial/\partial x = 0$
- the 5 equations simplify:

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

slab-on-a-slope 2

- add known (but non-penetrating) basal velocity conditions:

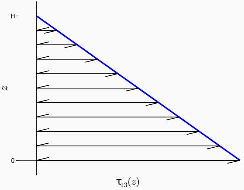
$$u(0) = u_0, \qquad w(0) = 0$$

- add zero normal stress condition at surface $z = H = s$:

$$p(H) = 0$$

- then by integrating vertically, get:

$$\begin{aligned} w(z) &= 0 \\ p(z) &= \rho g \cos\alpha (H - z) \\ \tau_{13}(z) &= \rho g \sin\alpha (H - z) \end{aligned}$$

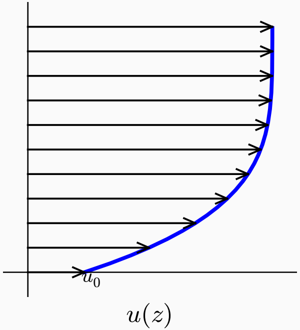


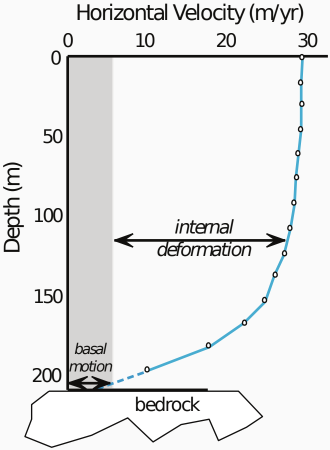
- note τ_{13} is linear in depth
- from $u_z = 2A\tau^2\tau_{13}$, integrate to get the velocity formula:

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

slab-on-a-slope 3

- do we believe these results?
- velocity formula gives figure below
- compare to observations at right





[velocity profile of the Athabasca Glacier, Canada; from inclinometry (Savage & Paterson 1963)]

plan for hour 2: surface velocity from the SIA model

- model: the nonsliding shallow ice approximation (SIA)
    - this model adopts the above slab-on-slope formula for $u(x)$ *as a general formula for glaciers*
    - it is an imperfect stress balance (momentum) model for a glacier, but it gives a meaningful velocity field in any geometry
    - there are careful shallowness scale analysis arguments ($\epsilon = [H]/[L] \to 0$) for this SIA model [Fow97]
- numerical modeling goals:
    - finite difference computations of surface velocity $u(x)$, $w(x)$ from SIA model
    - one short Python code: py/shallowuw.py
- to address:
    1. how to make the computations robust over geometries?
    2. how do you include sliding into the SIA?
    3. how does the SIA velocity field compare to the Stokes field for the same geometry?

SIA formulas for velocity

- the planar, isothermal, Glen-law, non-sliding shallow ice approximation (SIA) is the following system of equations:

$$u = -\frac{2}{n+1}A(\rho g)^n \left|\frac{\partial s}{\partial x}\right|^{n-1} \frac{\partial s}{\partial x}\left[\left(s-b\right)^{n+1} - \left(s-z\right)^{n+1}\right]$$

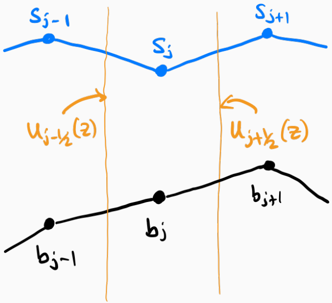
$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0$$

  - u, w here depend on x and z
  - s, b depend only on x
- note that the horizontal velocity formula is of the form

$$u = -\Gamma \frac{\partial s}{\partial x},$$

with positive coefficient Γ, so ice flows downhill

finite differences for $u(x)$



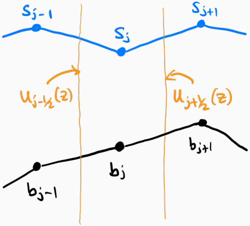
- centered-differencing the surface elevations gives slopes computed on the staggered grid:

$$\left.\frac{\partial s}{\partial x}\right|_{j+1/2} = \frac{s_{j+1} - s_j}{\Delta x}$$

- averaging gives elevations computed on the staggered grid:

$$s_{j+1/2} = \frac{s_j + s_{j+1}}{2}, \qquad b_{j+1/2} = \frac{b_j + b_{j+1}}{2}$$

finite differences for $u(x)$ 2



- notational convenience: given j, let

$$\alpha = \frac{2}{n+1} A(\rho g)^n \left| \frac{\partial s}{\partial x}\Big|_{j+1/2} \right|^{n-1} \frac{\partial s}{\partial x}\Big|_{j+1/2}$$

$$\beta = \left(s_{j+1/2} - b_{j+1/2}\right)^{n+1}$$

- horizontal velocity defined at all levels:

$$u_{j+1/2}(z) = \begin{cases} 0, & z \le b_{j+1/2} \\ -\alpha \left(\beta - (s_{j+1/2} - z)^{n+1}\right), & b_{j+1/2} < z < s_{j+1/2} \\ -\alpha\beta, & s_{j+1/2} \le z \end{cases}$$

Python code: SIA computation of horizontal velocity

- computing the surface value of the horizontal velocity at each regular grid location x_j is a Python function:

```
def u(x, b, s):
    assert np.all(b <= s)              # check admissibility
    dx = x[1] - x[0]
    dsdx = (s[1:] - s[:-1]) / dx
    Hstag = ((s[:-1] - b[:-1]) + (s[1:] - b[1:])) / 2.0
    C = (2.0 / (n+1)) * A * (rhoi * g)**n
    ult = - C * abs(dsdx[:-1])**(n-1) * dsdx[:-1] * Hstag[:-1]**(n+1)
    urt = - C * abs(dsdx[1:])**(n-1)  * dsdx[1:]  * Hstag[1:]**(n+1)
    return (ult + urt) / 2.0
```

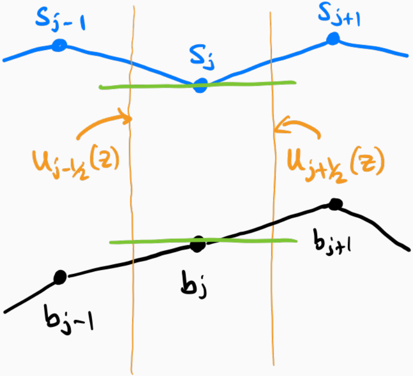
finite differences for $w(x)$

- we can integrate vertically to recover the surface value of w from $\partial u/\partial x$, by using incompressibility $\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0\right)$:

$$\begin{aligned} w(x, s(x)) &= w(x, s(x)) - w(x, b(x)) \\ &= -\int_{b(x)}^{s(x)} \frac{\partial u}{\partial x}(x, z)\, dz \end{aligned}$$

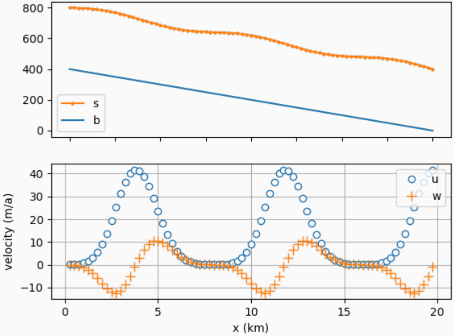
  - for a non-sliding and non-penetrating model: $w(x, b(x)) = 0$
- $w_j \approx w(x_j, s_j)$ is found by integrating vertically:

$$\begin{aligned} w_j &= -\int_{b_j}^{s_j} \frac{u_{j+1/2}(z) - u_{j-1/2}(z)}{\Delta x}\, dz \\ &= -\frac{1}{\Delta x}\left(\int_{b_j}^{s_j} u_{j+1/2}(z)\, dz - \int_{b_j}^{s_j} u_{j-1/2}(z)\, dz\right) \end{aligned}$$



- this code is more complicated *thus not shown ...*
  - see robustness aspect: integrate $u_{j\pm1/2}(z)$ from b_j to s_j, even if that puts you above or below the ice
- see the code py/shallowuw.py
  - result: visualization of $u(x), w(x)$ at the surface

live demo: SIA velocity for wavy surface



live demo

```
$ python3 shallowuw.py
$ eog output/uw.png
```

- the following modification suggests robustness:
    1. `H0` $= 100$ m and `wave_amp` $= 200$ m

time-dependent SKE + SIA approximation

- the equations of the time-dependent SIA model, using surface elevation as the variable, are the following:

$$\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} = a + w$$

$$u = -\frac{2}{n+1}A(\rho g)^n \left|\frac{\partial s}{\partial x}\right|^{n-1} \frac{\partial s}{\partial x}\left[(s-b)^{n+1} - (s-z)^{n+1}\right]$$

$$w = -\int_b^s \frac{\partial u}{\partial x}(x,z)\,dz$$

- implementing is exercise **13**
    - combine my codes py/surface1d.py and py/shallowuw.py
- important fact which I want to flesh-out: this model *diffuses* the surface elevation
    - because ice flows downhill
    - time-step conditions sufficient for stability are known *for SIA*

SKE versus mass continuity equation

- in non-sliding, incompressible ice we have equations:

$$
\begin{aligned}
\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} &= a + w && \textit{SKE} \\
\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} &= 0 && \textit{incompressible} \\
u(x, b(x)) = 0, w(x, b(x)) &= 0 && \textit{non-sliding/penetrating}
\end{aligned}
$$

- exercise 4 asks you to derive via vertical integration:

$$
\frac{\partial H}{\partial t} + \frac{\partial}{\partial x}(\bar{u}H) = a \qquad \textit{mass continuity equation}
$$

where $H = s - b$ is ice thickness and $\bar{u}$ is the vertically-averaged horizontal velocity

- main idea. for incompressible ice these are equivalent:

$$
\text{SKE} \quad \leftrightarrow \quad \text{mass continuity equation}
$$

time-dependent SIA for thickness, and diffusivity

- returning to SIA velocities …
- we can write the time-dependent equation in terms of thickness:

$$\frac{\partial H}{\partial t} = a + \frac{\partial}{\partial x}\left(D \frac{\partial s}{\partial x}\right)$$

where D is the diffusivity in the SIA:

$$D = \frac{2}{n+2} A(\rho g)^n H^{n+2} \left|\frac{\partial s}{\partial x}\right|^{n-1}$$

- analogy to Fourier's heat equation:

$$\frac{\partial H}{\partial t} = a + \frac{\partial}{\partial x}\left(D \frac{\partial s}{\partial x}\right) \quad \leftrightarrow \quad \frac{\partial T}{\partial t} = f + \frac{\partial}{\partial x}\left(D \frac{\partial T}{\partial x}\right)$$

where T is temperature and f is a heat source

questions and issues for the SIA

MAJOR modeling questions

1. what do we lose when using the SIA, versus Stokes?
2. how to add sliding to SIA?

mathematical issues

3. well-posedness not fully resolved *talk separately?*

numerical questions

4. how to do efficient implicit time-stepping?

questions and issues for the SIA

MAJOR modeling questions/answers

1. what do we lose when using the SIA, versus Stokes?
2. <u>answer:</u> add sliding via a membrane-stress-resolving balance

mathematical issues

3. well-posedness not fully resolved *talk separately?*

numerical questions

4. how to do efficient implicit time-stepping?

time-dependent SKE + Stokes approximation

- time-dependent SKE and Glen-law Stokes model:

$$\begin{aligned}
\frac{\partial s}{\partial t} - \mathbf{u} \cdot \mathbf{n}_s &= a \\
\nabla \cdot \mathbf{u} &= 0 \\
-\nabla \cdot \tau_{ij} + \nabla p &= \rho \mathbf{g} \\
D\mathbf{u}_{ij} &= A\tau^{n-1}\tau_{ij}
\end{aligned}$$

- implementing well is a major topic of current research
    - subject (as always) to $s \geq b$
    - I do not trust *anyone*'s view of time stepping (yet)

questions and issues for any Stokes model

modeling questions

1. sliding laws determine the basal stress state; what's best?
2. how to adapt Stokes to compressible (e.g. temperate) ice?

mathematical question

3. well-posedness of SKE+Stokes?

numerical questions/recommendations/issues

4. how to numerically combine SKE and Stokes in an efficient and stable manner?
5. as a practical matter, use a powerful finite element [ESW14] library
    - Firedrake? Elmer? FENiCs? Moose? ...
6. solver performance is a major concern ... I'm sure the speed will be adequate by 2050

an opinionated Summary

- numerical models are inevitable, whether you like them or not

an opinionated Summary

- numerical models are inevitable, whether you like them or not
- most of your work is to understand the math, not to write the code

an opinionated Summary

- numerical models are inevitable, whether you like them or not
- most of your work is to understand the math, not to write the code
- writing your own numerical models is of great value, but you should toss the result in the trash once you understand it

an opinionated Summary

- numerical models are inevitable, whether you like them or not
- most of your work is to understand the math, not to write the code
- writing your own numerical models is of great value, but you should toss the result in the trash once you understand it
- respect surface mass/energy balance just as much as ice dynamics

an opinionated Summary

- numerical models are inevitable, whether you like them or not
- most of your work is to understand the math, not to write the code
- writing your own numerical models is of great value, but you should toss the result in the trash once you understand it
- respect surface mass/energy balance just as much as ice dynamics
- machine learning *after* continuum mechanics

Questions?

further reading

[ESW14] H. C. Elman, D. J. Silvester, and A. J. Wathen.
Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics.
Oxford University Press, Oxford, UK, 2nd edition, 2014.

[Fow97] A. C. Fowler.
Mathematical Models in the Applied Sciences.
Cambridge Univ. Press, 1997.

[GB09] R. Greve and H. Blatter.
Dynamics of Ice Sheets and Glaciers.
Springer, Berlin, 2009.

[LeV07] R. J. LeVeque.
Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems.
SIAM Press, 2007.

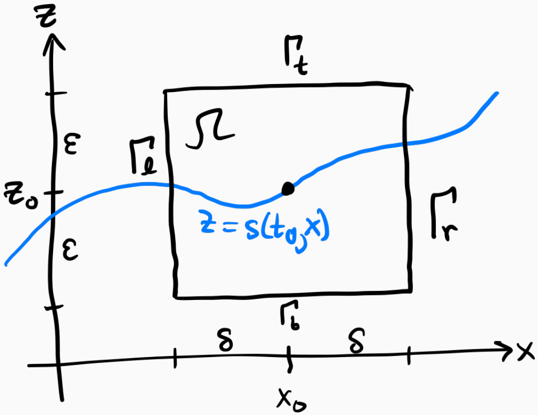
[MM05] K. W. Morton and D. F. Mayers.
Numerical Solution of Partial Differential Equations: An Introduction.
Cambridge University Press, 2nd edition, 2005.

[SH13] C. Schoof and I. J. Hewitt.
Ice-sheet dynamics.
Annual Review of Fluid Mechanics, 45:217–239, 2013.

Extra Slides

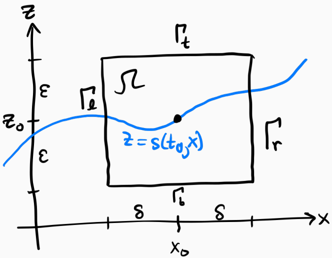
1. derive SKE from mass conservation

SKE from mass conservation



- these extra slides derive the surface kinematical equation (SKE) from mass conservation applied over a rectangle
- contrast with standard (dismissive!) derivations [GB09, SH13]

SKE from mass conservation 2

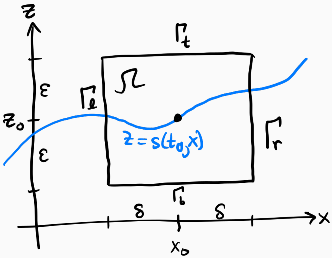


- $\Omega = $ (rectangle centered at$(x_0, z_0)) = [x_0 - \delta, x_0 + \delta] \times [z_0 - \epsilon, z_0 + \epsilon]$
  - where $s(t_0, x_0) = z_0$
- let $M(t)$ be the mass of (constant-density) ice within Ω at time t:

$$M(t) = \int_\Omega \rho_\text{i} \mathbb{1}_{\{z < s(t,x)\}} \, dx \, dz = \rho_\text{i} \int_{x_0-\delta}^{x_0+\delta} s(t,x) - z_0 + \epsilon \, dx$$

- notation: $\boldsymbol{\psi} = $ mass flux, $\hat{\mathbf{n}} = $ outward unit normal

SKE from mass conservation 3



- assumptions:
  - there are no mass sources within Ω
  - there is an interval of time t so that $z_0 - \epsilon < s(t,x) < z_0 + \epsilon$
  - the ice velocity $\mathbf{u}(t,x,z) = (u,w)$ is defined on $\{z < s(t,x)\}$
  - the SMB $a(t,x)$ is a mass flux only across Γ_t:
    - $\psi\big|_{\Gamma_t} = -\rho_i a(t,x)\hat{\mathbf{z}}$ *note $a > 0$ is precipitation*
  - the other mass fluxes are advective:
    - $\psi\big|_{\Gamma_b} = \rho_i w(t,x,z_0 - \epsilon)\hat{\mathbf{z}}$
    - $\psi\big|_{\Gamma_{\{\ell,r\}}} = \begin{cases} \rho_i u(t, x_{\{\ell,r\}}, z)\hat{\mathbf{x}}, & \text{if } z < s(t, x_{\{\ell,r\}}), \\ 0, & \text{otherwise} \end{cases}$

SKE from mass conservation 4

- mass is conserved:

$$
\begin{aligned}
\frac{dM}{dt} &= -\int_{\partial\Omega} \boldsymbol{\psi}\cdot\hat{\mathsf{n}}\,ds \\
&= \underbrace{\rho_{\mathrm{i}}\int_{x_0-\delta}^{x_0+\delta} a(t,x)\,dx}_{\Gamma_t} + \underbrace{\rho_{\mathrm{i}}\int_{x_0-\delta}^{x_0+\delta} w(t,x,z_0-\epsilon)\,dx}_{\Gamma_b} \\
&\qquad + \underbrace{\rho_{\mathrm{i}}\int_{z_0-\epsilon}^{s(t,x_0-\delta)} u(t,x_0-\delta,z)\,dz}_{\Gamma_\ell} - \underbrace{\rho_{\mathrm{i}}\int_{z_0-\epsilon}^{s(t,x_0+\delta)} u(t,x_0+\delta,z)\,dz}_{\Gamma_r}
\end{aligned}
$$

SKE from mass conservation 5

- time derivative:

$$\begin{aligned}
\frac{dM}{dt} &= \lim_{\omega\to 0} \frac{M(t+\omega) - M(t)}{\omega} \\
&= \lim_{\omega\to 0} \frac{\rho_{\mathrm{i}}}{\omega} \int_{\Omega} \mathbb{1}_{\{z<s(t+\omega,x)\}} - \mathbb{1}_{\{z<s(t,x)\}}\, dz\, dx \\
&= \lim_{\omega\to 0} \frac{\rho_{\mathrm{i}}}{\omega} \int_{x_0-\delta}^{x_0+\delta} s(t+\omega,x) - s(t,x)\, dx \\
&= \rho_{\mathrm{i}} \int_{x_0-\delta}^{x_0+\delta} \frac{\partial s}{\partial t}(t,x)\, dx
\end{aligned}$$

- assume smoothness sufficient for Taylor expansions on Ω:

$$\begin{aligned}
a(t,x) &= a(t,x_0) + O(\delta) \\
u(t,x,z) &= u(t,x_0,z_0) + O(\delta) + O(\epsilon) \\
w(t,x,z) &= w(t,x_0,z_0) + O(\delta) + O(\epsilon) \\
\frac{\partial s}{\partial t}(t,x) &= \frac{\partial s}{\partial t}(t,x_0) + O(\delta)
\end{aligned}$$

SKE from mass conservation 6

- combine mass conservation, time derivative, and Taylor:

$$\begin{aligned}2\delta\rho_\mathrm{i}\frac{\partial s}{\partial t}(t,x_0) + O(\delta^2) = 2\delta\rho_\mathrm{i} a(t,x_0) + O(\delta^2) \\ + 2\delta\rho_\mathrm{i} w(t,x_0,z_0) + O(\delta^2) + O(\delta\epsilon) \\ - \rho_i u(t,x_0,z_0)\Big[s(t,x_0+\delta) - s(t,x_0-\delta)\Big] + O(\delta\epsilon)\end{aligned}$$

- divide by $2\rho_\mathrm{i}\delta$:

$$\begin{aligned}\frac{\partial s}{\partial t}(t,x_0) = a(t,x_0) + w(t,x_0,z_0) - u(t,x_0,z_0)\frac{s(t,x_0+\delta) - s(t,x_0-\delta)}{2\delta} \\ + O(\delta) + O(\epsilon)\end{aligned}$$

- limit $\delta,\epsilon \to 0$ to get SKE at $(t,x_0,z_0) = (t,x_0,s(t,x_0))$:

$$\frac{\partial s}{\partial t} = a + w - u\frac{\partial s}{\partial x}$$