Ice Sheet System Model
Inverse Methods

Mathieu MORLIGHEM[1]

[1] University of California, Irvine

Outline

Ice flow model equations

Field equations

1. Stokes flow:

$$\nabla \cdot \boldsymbol{\sigma} + \rho\, \mathbf{g} = \mathbf{0} \tag{1}$$

2. Incompressibility:

$$\nabla \cdot \mathbf{v} = 0 \tag{2}$$

3. Constitutive law:

$$\boldsymbol{\sigma}' = \boldsymbol{\sigma} + p\,\mathbf{I} = 2\mu\dot{\boldsymbol{\varepsilon}} \qquad \mu = \frac{B}{2\,\dot{\varepsilon}_e^{1-1/n}} \tag{3}$$

Ice flow model equations

Field equations

1. Stokes flow:

$$\nabla \cdot \boldsymbol{\sigma} + \rho\, \mathbf{g} = \mathbf{0} \tag{1}$$

2. Incompressibility:

$$\nabla \cdot \mathbf{v} = 0 \tag{2}$$

3. Constitutive law:

$$\boldsymbol{\sigma}' = \boldsymbol{\sigma} + p\,\mathbf{I} = 2\mu\dot{\boldsymbol{\varepsilon}} \qquad \mu = \frac{B}{2\ \dot{\varepsilon}_e^{1-1/n}} \tag{3}$$

Boundary conditions

Ice/Air interface: Free surface	$\boldsymbol{\sigma}\cdot\mathbf{n} = P_{atm}\,\mathbf{n} \simeq \mathbf{0}$	on Γ_s
Ice/Ocean interface: water pressure	$\boldsymbol{\sigma}\cdot\mathbf{n} = P_w\,\mathbf{n}$	on Γ_w
Ice/Bedrock interface (1): lateral friction	$\left(\boldsymbol{\sigma}\cdot\mathbf{n} + \alpha^2\mathbf{v}\right)_{\parallel} = \mathbf{0}$	on Γ_b
Ice/Bedrock interface (2): impenetrability	$\mathbf{v}\cdot\mathbf{n} = -\dot{M}_b\, n_z$	on Γ_b

Ice flow model equations

Field equations

1. Stokes flow:

$$\nabla \cdot \boldsymbol{\sigma} + \rho\, \mathbf{g} = \mathbf{0} \tag{1}$$

2. Incompressibility:

$$\nabla \cdot \mathbf{v} = 0 \tag{2}$$

3. Constitutive law:

$$\boldsymbol{\sigma}' = \boldsymbol{\sigma} + p\,\mathbf{I} = 2\mu\dot{\boldsymbol{\varepsilon}} \qquad \mu = \frac{B}{2\;\dot{\varepsilon}_e^{1-1/n}} \tag{3}$$

Boundary conditions

Ice/Air interface: Free surface	$\boldsymbol{\sigma} \cdot \mathbf{n} = P_{atm}\,\mathbf{n} \simeq \mathbf{0}$	on Γ_s
Ice/Ocean interface: water pressure	$\boldsymbol{\sigma} \cdot \mathbf{n} = P_w\,\mathbf{n}$	on Γ_w
Ice/Bedrock interface (1): lateral friction	$\left(\boldsymbol{\sigma} \cdot \mathbf{n} + \alpha^2 \mathbf{v}\right)_{\parallel} = \mathbf{0}$	on Γ_b
Ice/Bedrock interface (2): impenetrability	$\mathbf{v} \cdot \mathbf{n} = -\dot{M}_b\, n_z$	on Γ_b

Inverse problems

- Basal friction and ice hardness are difficult to measure
- Use extra datasets to infer unknows

$\rightarrow$ ex: surface velocities derived from InSAR

Inverse problems

- Basal friction and ice hardness are difficult to measure
- Use extra datasets to infer unknows

$\rightarrow$ ex: surface velocities derived from InSAR

PDE-constrained optimization

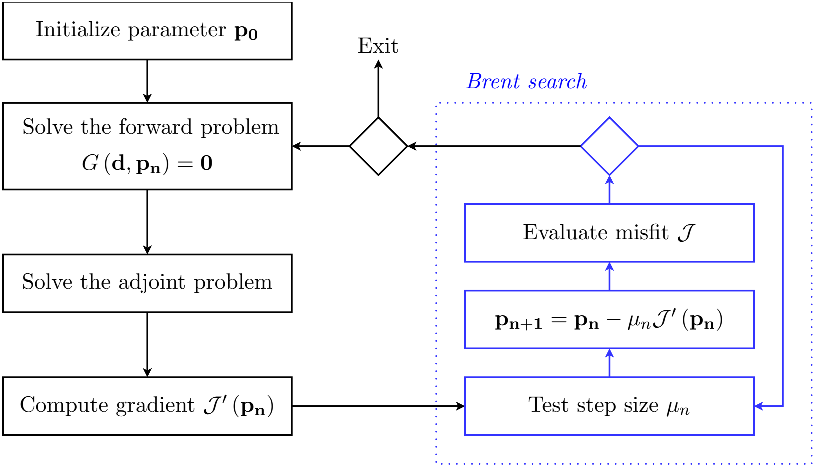
Minimize cost function

$$\mathcal{J}(\mathbf{v}, \alpha) = \frac{1}{2} \int_{\Gamma_s} \left(v_x - v_x^{obs}\right)^2 + \left(v_y - v_y^{obs}\right)^2 dS + \mathcal{R}(\alpha) \tag{4}$$

Subject to:

$$\begin{aligned} \nabla \cdot 2\mu\dot{\varepsilon} - \nabla p + \rho\mathbf{g} &= \mathbf{0} && \text{in } \Omega \\ \nabla \cdot \mathbf{v} &= 0 && \text{in } \Omega \\ \boldsymbol{\sigma} \cdot \mathbf{n} &= \mathbf{f} && \text{on } \Gamma_s \cup \Gamma_w \\ \left(\boldsymbol{\sigma} \cdot \mathbf{n} + \alpha^2 \mathbf{v}\right)_{\parallel} &= \mathbf{0} && \text{on } \Gamma_b \\ \mathbf{v} \cdot \mathbf{n} &= -\dot{M}_b\, n_z && \text{on } \Gamma_b \end{aligned} \tag{5}$$

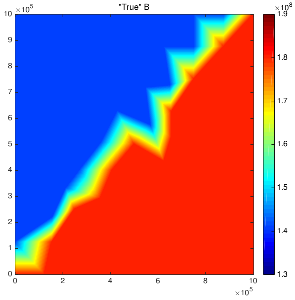
Algorithm of resolution

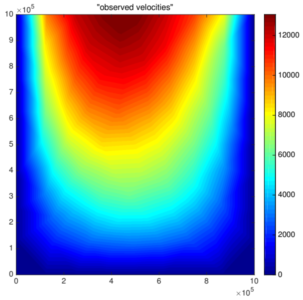


Hands on 1 (ice rigidity)

```
%Generate observations
md = model;
md = triangle(md,'DomainOutline.exp',100000);
md = setmask(md,'all','');
md = parameterize(md,'Square.par');
md = setflowequation(md,'SSA','all');
md.cluster = generic('np',2);
md = solve(md,StressbalanceSolutionEnum());
```

```
md.materials.rheology_B=1.8*10^8*ones(md.mesh.numberofvertices,1);
md.materials.rheology_B(find(md.mesh.x<md.mesh.y))=1.4*10^8;
```



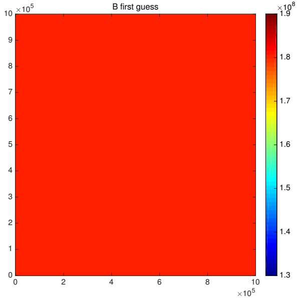


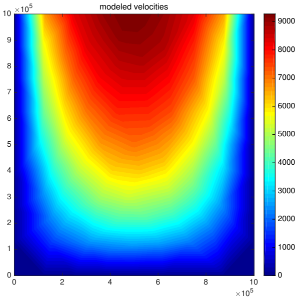
Start from constant hardness

```
%Modify rheology, now constant
loadmodel('model1.mat');
md.materials.rheology_B(:) = 1.8*10^8;

%results of previous run are taken as observations
md.inversion=m1qn3inversion();
md.inversion.vx_obs  = md.results.StressbalanceSolution.Vx;
md.inversion.vy_obs  = md.results.StressbalanceSolution.Vy;
md.inversion.vel_obs = md.results.StressbalanceSolution.Vel;

md = solve(md,StressbalanceSolutionEnum());
```





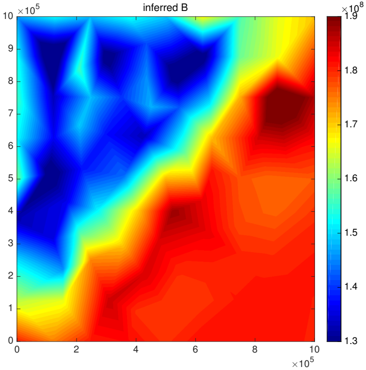
Inverse method

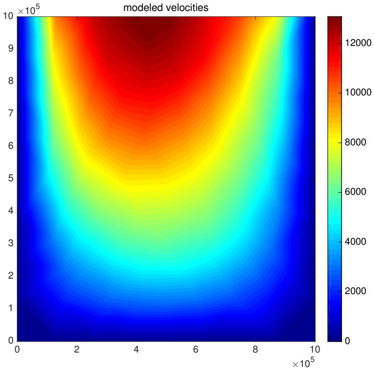
```
%invert for ice rigidity
loadmodel('model2.mat');

%Set up inversion parameters
maxsteps = 20;
md.inversion.iscontrol = 1;
md.inversion.control_parameters = {'MaterialsRheologyBbar'};
md.inversion.maxsteps = maxsteps;
md.inversion.cost_functions = 101;
md.inversion.cost_functions_coefficients = ones(md.mesh.numberofvertices,1);
md.inversion.min_parameters    = cuffey(273)*ones(md.mesh.numberofvertices,1);
md.inversion.max_parameters    = cuffey(200)*ones(md.mesh.numberofvertices,1);

%Go solve!
md.verbose=verbose(0);
md=solve(md,StressbalanceSolutionEnum());
```

Inverse method



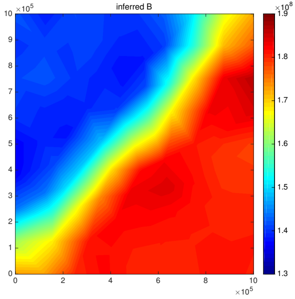


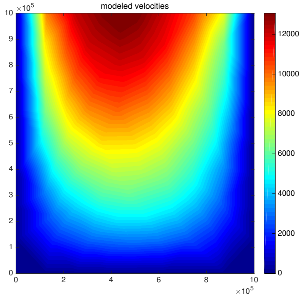
Inverse method

- Add Tikhonov regularization

$$\mathcal{J}(B) = \int_{\Gamma_s} \frac{1}{2}\left(v_x - v_x^{obs}\right)^2 + \frac{1}{2}\left(v_x - v_x^{obs}\right)^2 d\Gamma + \int_{\Omega} \frac{1}{2}\nabla B \cdot \nabla B d\Omega \qquad (6)$$

```
md.inversion.cost_functions = [101 502];
md.inversion.cost_functions_coefficients        = ones(md.mesh.numberofvertices,1);
md.inversion.cost_functions_coefficients(:,2) = 10^-16*ones(md.mesh.numberofvertices,1);
```





Hands on 2 (friction)

- Same example but now for a grounded glacier:
- Changes step 1:
  1. increase bed and surface elevation by 100 m
  2. mask is now all grounded
  3. $B = 1.8 \times 10^8$ uniform
  4. friction coefficient: 50, and 10 for $600000 < x < 400000$
- Changes step 2:
  1. friction coefficient uniform (50)
- Changes step 3:
  1. We now invert for `'FrictionCoefficient'`
  2. Do we keep the same cost function (probably not...)?
  3. we want the parameter to be between 1 and 100


Thanks!