ISSM Workshop 2016

Ice Sheet System model

Application to Pine Island Glacier

Helene SEROUSSI[1]

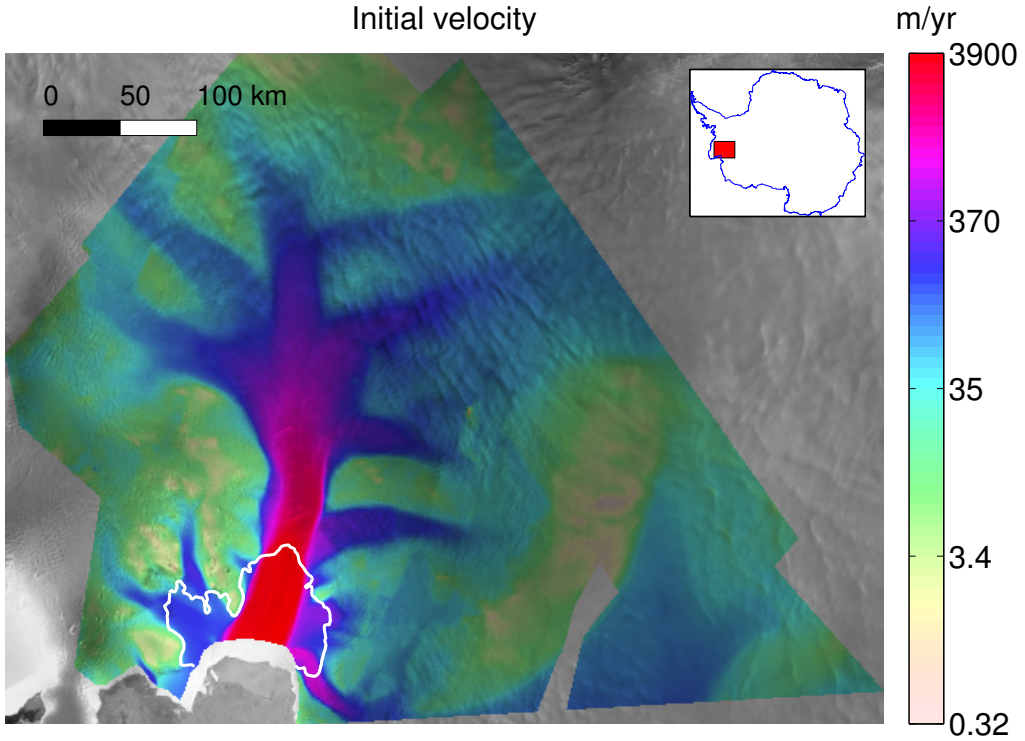
[1] Caltech's Jet Propulsion Laboratory, Pasadena

Outline

Objectives

Parameterize and run a simulation of a real glacier

0. Define modeling region
1. Create a mesh
2. Apply masks
3. Parameterize model
4. Invert friction coefficient
5. Plot results
6. Run Higher-Order model



→ Pine Island Glacier

File description

Files needed for the simulation in: `./Pig/` and `./Data/`

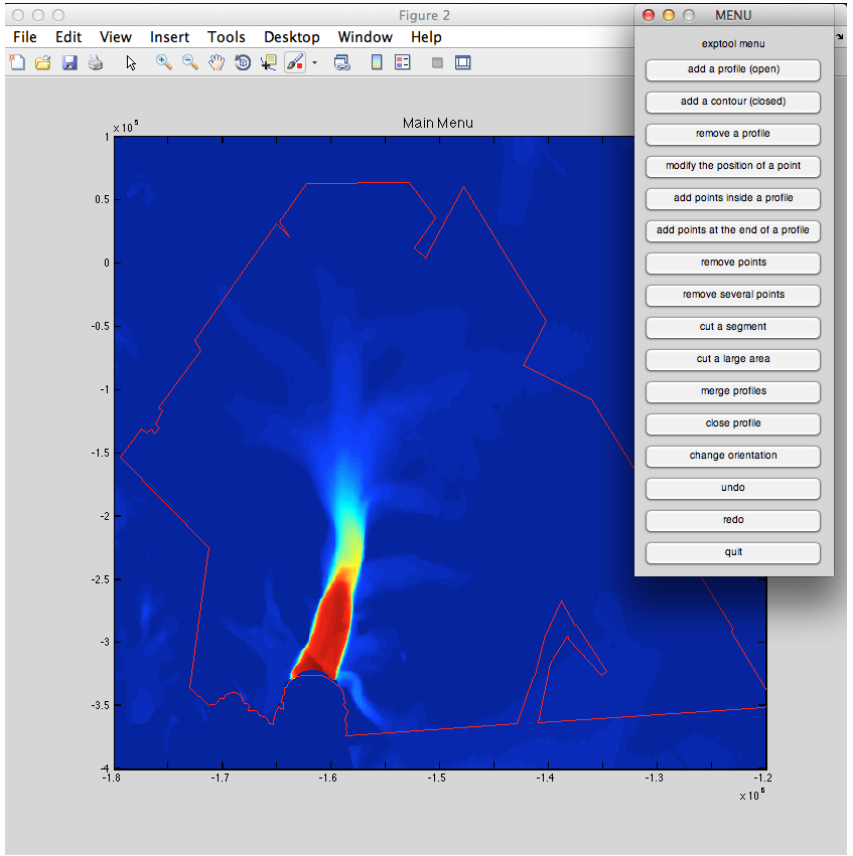
- Matlab "runme.m" file
  - Structure of the simulation
- Matlab ".par" file
  - Most parameters needed for a simulation
- Shape files ".exp"
  - Shape files to define geometric boundaries of the simulation
- Data files
  - NetCDF files
  - Any other format supported by Matlab

Step 0: Setting-up domain outline

Draw domain outline for Pine Island

- Use velocity map
- name: `./DomainOutline.exp`

1. Run `PigRegion` in Matlab
2. Create the domain with `exptool('Name')`



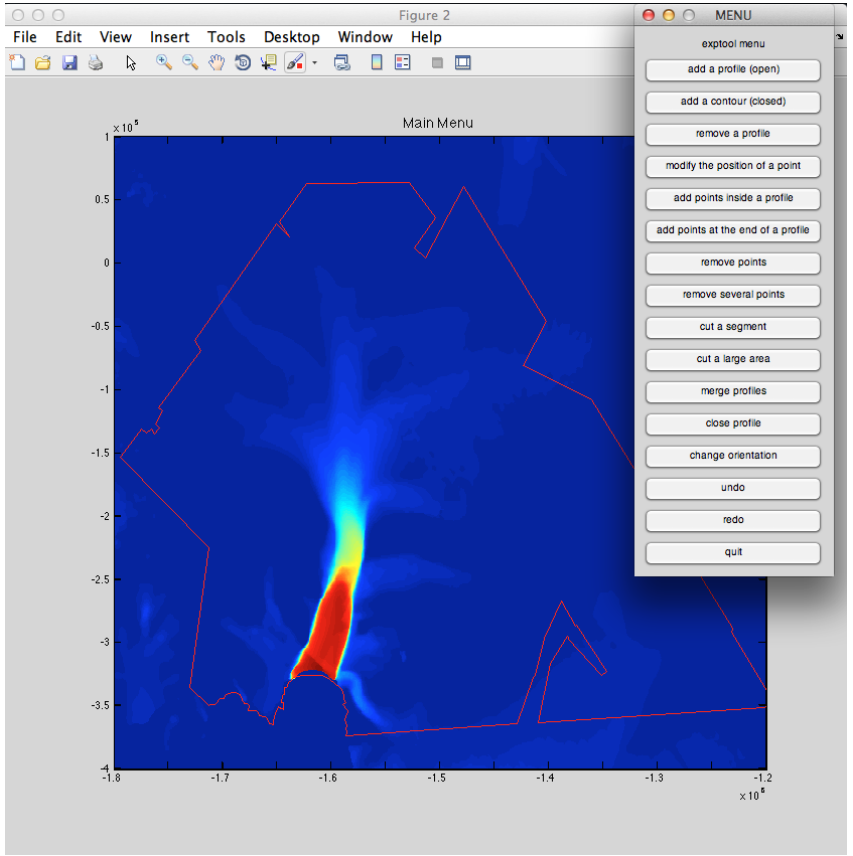
- Use same domain
- Change: `./DomainOutline.bkp` to `./DomainOutline.exp`

Step 0: Setting-up domain outline

Draw domain outline for Pine Island

- Use velocity map
- name: `./DomainOutline.exp`

1. Run `PigRegion` in Matlab
2. Create the domain with `exptool('Name')`



- Use same domain
- Change: `./DomainOutline.bkp` to `./DomainOutline.exp`

Step 1: Mesh

The mesh generation based on the velocity gradient
In `runme.m` file:

1. Uniform initial mesh of given resolution (`hinit`) with `bamg`

```
% Generate an initial uniform mesh (resolution = hinit m)
md=bamg(model,'domain',domain,'hmax',hinit);
```

2. Anisotrop mesh based on the surface velocity

```
% Adapt the mesh to minimize error in velocity interpolation
md=bamg(md,'hmax',hmax,'hmin',hmin,'gradation',gradation,'field',vel_obs,'err',err);
```

This first argument is the previous (`md`) mesh that contains the observed velocities.

3. Plot mesh

```
%ploting
plotmodel(md,'data','mesh')
```

Step 2: Mask

1. Specify grounded and floating areas
   - ISSM default: ice is grounded
   - Use field `md.mask.groundedice_levelset`
     - Grounded ice > 0
     - Floating ice < 0
     - Grounding line at the 0 level

```
%fill in the md.mask structure
md.mask.groundedice_levelset=groundedice; %ice is grounded for mask equal one
```

2. Specify where ice is present
   - ISSM default: ice everywhere
   - Use field `md.mask.ice_levelset`
     - Presence of ice < 0
     - Absence of ice > 0

```
md.mask.ice_levelset=-1*ones(md.mesh.numberofvertices,1);%ice is present when negatvie
```

Step 3: Parameterize Model

Most parameters in `./Pig.par`

SeaRise data to parameterize most model fields:

- Geometry
  - Only define two variables and compute the third
  - `find` function allow to define subsets of nodes
- Initialization parameters
- Material parameters
- Forcings
- Friction and inversion set up
- Boundary Conditions

```
md = parameterize(md,'./Pig.par');
```

Some parameters adjusted in `runme.m`

- E.g.: stress balance equation

```
% Use a MacAyeal flow model
md = setflowequation(md,'SSA','all');
```

Step 4: Inversion of basal friction

The friction coefficient β is inversed from the surface velocities

$$\tau_b = -\beta^2 N^r \|\mathbf{v_b}\|^{s-1} \mathbf{v_b}$$

- τ_b : Basal drag
- N : Effective pressure
- v_b : Basal velocity (equal surface in SSA approximation)
- r : Exponent (equals q/p of the parameter file)
- s : Exponent (equals $1/p$ of the parameter file)

The procedure for the inversion method is as follow

- Velocity is computed from the SSA approximation
- Misfit of the cost function is computed
- Friction coefficient is modified

Step 4: Inversion of basal friction

1. Inversion parameters in `md.inversion`

```
% Control general
md.inversion.iscontrol=1;
md.inversion.maxsteps=20;
md.inversion.maxiter=40;
md.inversion.dxmin=0.1;
md.inversion.gttol=1.0e-4;
md.verbose=verbose('solution',true,'control',true);

% Cost functions
md.inversion.cost_functions=[101 103 501];
md.inversion.cost_functions_coefficients=ones(md.mesh.numberofvertices,3);
md.inversion.cost_functions_coefficients(:,1)=1;
md.inversion.cost_functions_coefficients(:,2)=1;
md.inversion.cost_functions_coefficients(:,3)=8e-15;

% Controls
md.inversion.control_parameters={'FrictionCoefficient'};
md.inversion.min_parameters=1*ones(md.mesh.numberofvertices,1);
md.inversion.max_parameters=200*ones(md.mesh.numberofvertices,1);
```

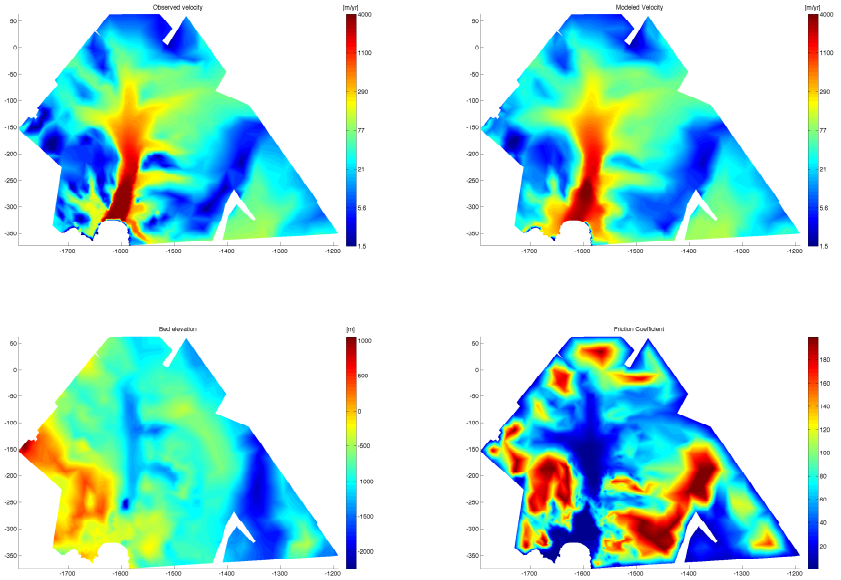
2. Solve stress balance equation

```
% Solve
md.cluster=generic('name',oshostname,'np',2);
md=solve(md,StressbalanceSolutionEnum);
```

Step 5: Plot Results

Plotting results with `plotmodel`

```
plotmodel(md,'nlines',2,'ncols',2,'unit#all','km','axis#all','equal',...
 'xlim#all',[min(md.mesh.x) max(md.mesh.x)]/10^3,...
 'ylim#all',[min(md.mesh.y) max(md.mesh.y)]/10^3,...
 'FontSize#all',12,...
 'data',md.initialization.vel,'title','Observed velocity',...
 'data',md.results.StressbalanceSolution.Vel,'title','Modeled Velocity',...
 'data',md.geometry.base,'title','Bed elevation',...
 'data',md.results.StressbalanceSolution.FrictionCoefficient,'title','Friction Coefficient',...
 'colorbar#all','on','colorbartitle#1-2','[m/yr]',...
 'caxis#1-2',([1.5,4000]),...
 'colorbartitle#3','[m]', 'log#1-2',10);
```



TIPS

`plotdoc` give most usual options of `plotmodel`

Step 6: Higher Order Ice Flow Model

Set up a 3D Higher-Order model

1. Load the previous step
   - Model load: `Control_drag`
2. Disable the inversion process
   - Change iscontrol in `md.inversion`
3. Extrude the mesh
   - `help extrude`
   - Reasonable number of layers
4. Change stress balance approximation
   - function `setflowequation`
5. Solve
   - Solving `StressBalanceSolution`
6. Save the model
   - Same as previous steps


Thanks!