ISSM Workshop 2012

Ice Sheet System model

Application to SeaRISE dataset, Greenland Ice Sheet

Chris BORSTAD[1], Bao DUONG[5,1], Feras HABBAL[2,1], Daria HALKIDES[1,3], Michiel HELSEN[2], Eric LAROUR[1], Mathieu MORLIGHEM[2], Lan NGUYEN[5,1], Gilberto PÉREZ[4,1], Eric RIGNOT[2,1], John SCHIERMEIER[1], **Nicole SCHLEGEL**[1], Hélène SEROUSSI[1]

[1] Jet Propulsion Laboratory - California Institute of Technology

[2] University of California, Irvine

[3] Joint Institute for Regional Earth System Science & Engineering, UCLA

[4] University of Southern California

[5] Cal Poly Pomona

December 2012

Overview

Goals

Use your new ISSM skills to run a very coarse Greenland model. Initialize your domain with a given exp file and parameterize it with the SeaRISE netcdf dataset.

Steps:

- Mesh Greenland with given exp
- Adapt mesh using SeaRISE velocity data
- Parameterize (as the PIG model), but specify velocity of at least one point to ensure non-sigularity.
- Diagnostic: run inverse method to control drag (30 steps recommended)
- Transient: 20 year run
  - Use an appropriate time step for your resolution
  - Force SeaRISE surface mass balance for 10 years
  - For the next 10 years, simulate a warming scenario: decrease the surface mass balance linearly, reaching a decrease of 1.0 m/y by year 20
- Plot transient results

First Run Step: Mesh

The domain file DomainOutline.exp resides in directory Exp_Par. First we mesh using the triangle method as follows:

```
%Generate initial uniform mesh (resolution = 20000 m)
md=triangle(model,'./Exp_Par/DomainOutline.exp',20000);
```

This creates a new model named md and meshes the model domain at a resolution of 20000 m.

Mesh, 2/4

Adapt

Next, adapt your mesh to SeaRISE velocities. The data resides in:

```
5   ncdata='./Data/Greenland_5km_dev1.2.nc';
```

Adapt:

- Fill variable vel with interpolated velocities (Hint: you need x and y velocities plus ncfile x and y coordinates)
- Mesh adapt your model (bamg)
    - Use variable vel
    - Set hmax=400000m and hmin=5000m
- Set model lat/long using SeaRISE projection information (see projection information in Greenland_5km_dev1.2.nc - Hint: in matlab you can use ndisp)
- Save your model to a file

Mesh, 3/4

Check Your Work

Solution:

```
% Get velocities (Note: You can use ncdisp('file') to see an ncdump)
x1   = ncread(ncdata,'x1');
y1   = ncread(ncdata,'y1');
velx = ncread(ncdata,'surfvelx');
vely = ncread(ncdata,'surfvely');
vx   = InterpFromGridToMesh(x1,y1,velx',md.mesh.x,md.mesh.y,0);
vy   = InterpFromGridToMesh(x1,y1,vely',md.mesh.x,md.mesh.y,0);
vel  = sqrt(vx.^2+vy.^2);

%Mesh Greenland
md=bamg(md,'hmax',400000,'hmin',5000,'gradation',1.7,'field',vel,'err',8);

%convert x,y coordinates (Polar stereo) to lat/lon
[md.mesh.lat,md.mesh.long]=xy2ll(md.mesh.x,md.mesh.y,+1,39,71);
```

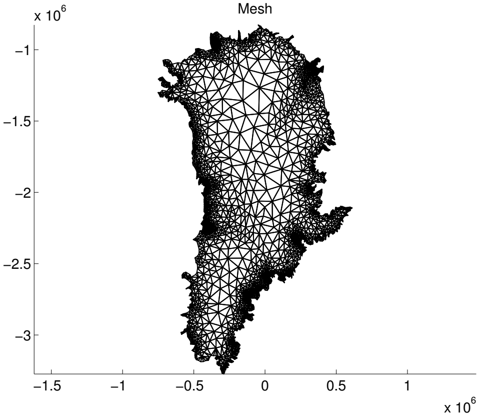
Now, plot your mesh.

```
plotmodel (md,'data','mesh');
```

Mesh, 4/4

Plot Mesh

Plot your mesh. It should look like:



Mesh

Full Solution

```
disp('    Step 1: Mesh creation');

%Generate initial uniform mesh (resolution = 20000 m)
md=triangle(model,'./Exp_Par/DomainOutline.exp',20000);

% Get velocities (Note: You can use ncdisp('file') to see an ncdump)
x1   = ncread(ncdata,'x1');
y1   = ncread(ncdata,'y1');
velx = ncread(ncdata,'surfvelx');
vely = ncread(ncdata,'surfvely');
vx   = InterpFromGridToMesh(x1,y1,velx',md.mesh.x,md.mesh.y,0);
vy   = InterpFromGridToMesh(x1,y1,vely',md.mesh.x,md.mesh.y,0);
vel  = sqrt(vx.^2+vy.^2);

%Mesh Greenland
md=bamg(md,'hmax',400000,'hmin',5000,'gradation',1.7,'field',vel,'err',8);

%convert x,y coordinates (Polar stereo) to lat/lon
[md.mesh.lat,md.mesh.long]=xy2ll(md.mesh.x,md.mesh.y,+1,39,71);

save ./Models/Greenland.Mesh_generation md;

plotmodel (md,'data','mesh');
```

Parameterization

Call the setmask function with empty arguments and then parameterize your mesh with file Exp_Par/Greenland.par. Then set your flow equation to macayeal for all.

Read though the parameter file Exp_Par/Greenland.par, similar to your PIG par file, but for Greenland. Here, we want to parameterize a full continental domain.

NB: No function is called to define boundary conditions. Instead, set spc velocities explicitly to zero, ensuring non-singularity.

```
md.diagnostic.spcvx = NaN*ones(md.mesh.numberofvertices,1);
md.diagnostic.spcvy = NaN*ones(md.mesh.numberofvertices,1);
md.diagnostic.spcvz = NaN*ones(md.mesh.numberofvertices,1);
location = 1.0e+06 *[.32011 -2.2039];
[dist pos]=min(sqrt((md.mesh.x - location(1)).^2 + (md.mesh.y - ...
      location(2)).^2));
md.diagnostic.spcvx(pos) = 0;
md.diagnostic.spcvy(pos) = 0;
md.diagnostic.spcvz(pos) = 0;
```

Parameterization

Solution

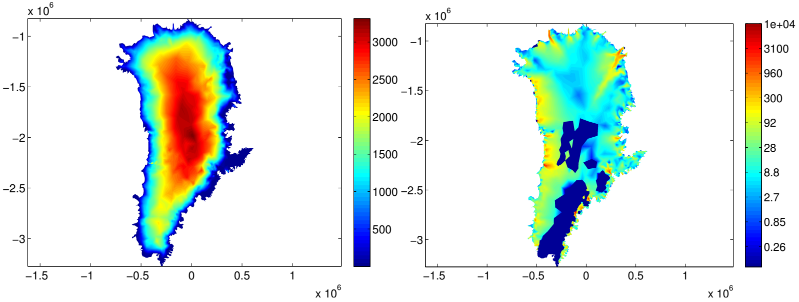
```
disp('    Step 2: Parameterization');
md = loadmodel('./Models/Greenland.Mesh_generation');

md = setmask(md,'','');
md = parameterize(md,'./Exp_Par/Greenland.par');
md = setflowequation(md,'macayeal','all');

save ./Models/Greenland.Parameterization md;
```

Now, plot thickness and velocity.

```
>> plotmodel(md,'data',md.geometry.thickness);
>> plotmodel(md,'data',md.initialization.vel,'caxis',[1e-1 1e4],'log',10);
```

Parameterization
Plot

Plot thickness and velocity. They should look like:



Diagnostic

Use control methods to inversely solve for Greenland FrictionCoefficient

Steps:

- Set cost functions
  - absolute value of surface velocity
  - log of surface velocity
  - drag coefficient gradient
- Set cost functions coefficients to 350, 0.6 and $2*10^{-6}$
- Set gradient scaling to 50
- Specify max inversion parameter = 200, min inversion parameter = 1
- Solve a 30-step Diagnostic in 2D, Macayeal
- Copy result Friction Coefficient to model (md) value
- Save your model

Startoff Code:

```
md = loadmodel('./Models/Greenland.Parameterization');

%Control general
md.inversion.iscontrol=1;
md.inversion.nsteps=30;
md.inversion.step_threshold=0.99*ones(md.inversion.nsteps,1);
md.inversion.maxiter_per_step=5*ones(md.inversion.nsteps,1);
```

NB: Remember that md.inversion can be called for help!

Diagnostic

Solution

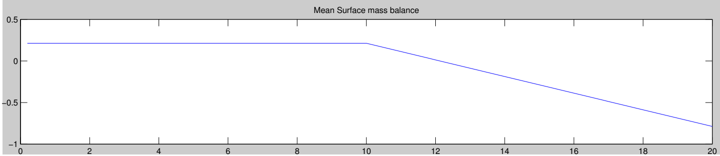
```
%Cost functions
md.inversion.cost_functions=[101*ones(md.inversion.nsteps,1) ...
      103*ones(md.inversion.nsteps,1) 501*ones(md.inversion.nsteps,1)];
md.inversion.cost_functions_coefficients=ones(md.mesh.numberofvertices,3);
md.inversion.cost_functions_coefficients(:,1)=350;
md.inversion.cost_functions_coefficients(:,2)=0.6;
md.inversion.cost_functions_coefficients(:,3)=2e-6;

%Controls
md.inversion.control_parameters={'FrictionCoefficient'};
md.inversion.gradient_scaling(1:md.inversion.nsteps)=50;
md.inversion.min_parameters=1*ones(md.mesh.numberofvertices,1);
md.inversion.max_parameters=200*ones(md.mesh.numberofvertices,1);

%Additional parameters
md.diagnostic.restol=0.01; md.diagnostic.reltol=0.1;
md.diagnostic.abstol=NaN;

%Go solve
md.cluster=generic('name',oshostname,'np',2);
md.solver=addoptions(md.solver,NoneAnalysisEnum,asmoptions);
md.solver=addoptions(md.solver,DiagnosticVertAnalysisEnum,jacobiasmoptions);
md.verbose=verbose('solution',true,'control',true);
md=solve(md,DiagnosticSolutionEnum);

%Update model friction fields accordingly
md.friction.coefficient=md.results.DiagnosticSolution.FrictionCoefficient;

save ./Models/Greenland.Control_drag md;
```

Transient

Forcing

You are ready to run a transient!

Simulate Greenland warming by forcing a temporal decrease in md.surfaceforcings.mass_balance:



Specify a transient forcing by adding a time value to the end (in the end+1 position) of the column of forcing variable values. For example, let smb be the initial values of surface mass balance. To impose the forcing above:

```
>> md.surfaceforcings.mass_balance = [ smb smb-1];
>> md.surfaceforcings.mass_balance = [ md.surfaceforcings.mass_balance; ...
    [10 20]];
```

NB: Prior to first and after last imposed time, forcing values remain constant. Between imposed times, forcings are linearly interpolated.

JPL ISSM

Transient

Solve

Set up your transient

Steps:

- Set control md.inversion.iscontrol back to 0
- Interpolate surface mass balance from SeaRISE dataset, converting from water to ice equivalent. NB: md.materials has helpful constants
- Impose SeaRise surface mass balance for 10 years then linearly decrease to 1 m/yr by year 20
- Set time step to 0.2 and output frequency to 1
- Solve a 20 year Transient in 2D, Macayeal
- Save your model

NB: Save the IceVolume in your transient results for plotting later:

```
md.transient.requested_outputs=IceVolumeEnum();
```

Transient

Solution Setup

```
md = loadmodel('./Models/Greenland.Control_drag');

%Set surface mass balance
x1   = ncread(ncdata,'x1');
y1   = ncread(ncdata,'y1');
smb = ncread(ncdata,'smb');
smb = InterpFromGridToMesh(x1,y1,smb',md.mesh.x,md.mesh.y,0);
smb = smb*md.materials.rho_freshwater/md.materials.rho_ice;
smb = [smb smb smb-1.0];
md.surfaceforcings.mass_balance = [smb;1 10 20];

%Set transient options, run for 20 years, saving every year
md.timestepping.time_step=0.2;
md.timestepping.final_time=20;
md.settings.output_frequency=1;

%Additional options
md.inversion.iscontrol=0;
md.transient.requested_outputs=IceVolumeEnum();
md.verbose=verbose('solution',true,'module',true,'convergence',true);

%Go solve
md.cluster=generic('name',oshostname,'np',2);
md=solve(md,TransientSolutionEnum);

save ./Models/Greenland.Transient md;
```

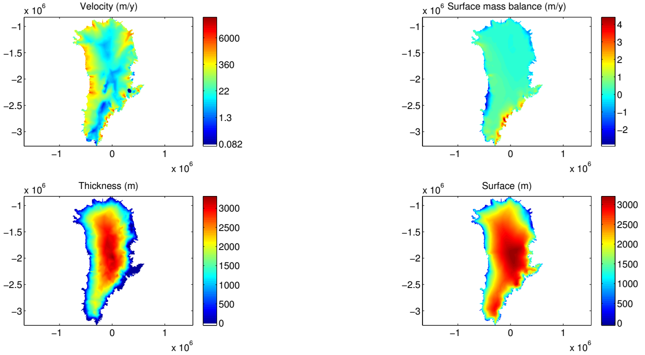
Transient Results

Plot Plan

Your results are located in md.results.TransientSolution. Plot your results.

First, plot the initial plan view of velocity, surface mass balance, thickness, and surface in four subplots on the same figure.

They should look like:



Plot

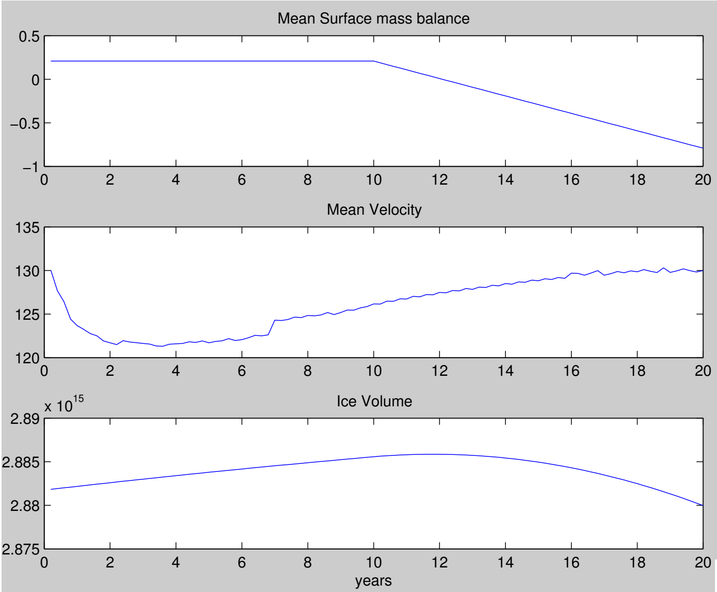
Solution

```
%Planview plots
plotmodel(md,'data',md.results.TransientSolution(end).Vel,'caxis',[1e-1 ...
      6000],...
 'log', 10, 'title', 'Velocity (m/y)','gridded',1, ...
 'data', md.results.TransientSolution(1).SurfaceforcingsMassBalance, ...
 'title', 'Surface mass balance (m/y)', 'gridded',1,...
 'data',md.results.TransientSolution(end).Thickness,...
 'title','Thickness (m)', 'gridded',1, ...
 'data',md.results.TransientSolution(end).Surface, ...
 'title', 'Surface (m)', 'gridded',1);
```

Next, plot a timeseries of mean surface mass balance, mean velocity, and ice volume. Hint to plot mean surface mass balance results:

```
%Line Plots
figure

%Plot surface mass balance
surfmb=[]; for i=1:100; surfmb=[surfmb ...
 md.results.TransientSolution(i).SurfaceforcingsMassBalance]; end
subplot(3,1,1); plot([0.2:0.2:20],mean(surfmb)); title('Mean Surface ...
      mass balance');
```

Your results should look like:



Transient run

Exercises

Extra Time? Try to something more complicated.

Addditional Suggested Exercises:

- Increase SMB instead of decrease over time
- Create an instantaneous step in SMB forcing at 10 years instead of a steady change over time
- Create a more advanced SMB forcing, like cyclic steps or a curve
- Force SMB to change only in certain areas of the ice sheet
- Add more melt in the ablation zone, but more snow in upper elevations
- Force another field transiently (e.g. friction coefficient)


Thanks!