ISSM Workshop 2012

JPL

Ice Sheet System model

Application to Pine Island Glacier

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

ISSM

December 2012

Overview

Goal

Use ISSM capabilities to run a ‘real-world’ example: Pine Island Glacier

runme.m steps [1 $\rightarrow$ 5]: (saving, loading model at each step)

1. Create numerical mesh: (BAMG)
   - Initial mesh of PIG with the provided exp outline (10 km resolution)
   - Refine/Adapt mesh (i.e. minimize error in velocity interpolation) using velocities from Rignot et al, 2011. NSIDC (highest resolution 5 km)
2. Create masks
   - Use SeaRISE thickness to specify floating vs grounded elements
3. Parameterize model
   - Using MacAyeal ice flow model
4. Control method for basal drag:
   - Inverting for coefficient of friction
5. Plot results
   - Observed velocity, modeled velocities, bed elevation, friction coefficient

Additional exercises: experiment with model parameters

- Vary temperature, coefficient of friction and re-run model (steps 1-5)
- Try Blatter-Pattyn ice flow model and plot results (steps 6-7)

Step 1: Mesh (runme.m)

The ‘exp’ file: DomainOutline.exp (provided in the directory Exp_Par) includes the Pine Island Glacier domain that will be analyzed using ISSM

- First: create an initial numerical mesh using the BAMG at 10 km resolution:

```
% Generate an initial uniform mesh (resolution = 10000 m)
md=bamg(model,'domain','Exp_Par/DomainOutline.exp','hmax',10000,'MaxCornerAngle',1);
```

This creates a new ISSM model (variable name: md) with a mesh of the provided PIG domain

- Second: refine mesh anisotropically using the observed velocity (Rignot et al, 2011)

```
% Interpolate velocities onto coarse mesh
vx_obs=InterpFromGridToMesh(x,y,flipud(vx'),md.mesh.x,md.mesh.y,0);
vy_obs=InterpFromGridToMesh(x,y,flipud(vy'),md.mesh.x,md.mesh.y,0);
vel_obs=sqrt(vx_obs.^2+vy_obs.^2);
clear vx vy;

% Adapt the mesh to minimize error in velocity interpolation
md=bamg(md,'hmax',400000,'hmin',5000,'gradation',1.7,'field',vel_obs,'err',8);
```

Step 2: Mask (runme.m)

Specify grounded vs ungrounded areas of the PIG domain

- ISSM default: ice is grounded
- setmask function: SeaRISE thickness data to specify elements on ice shelf

```
%read thickness mask from SeaRISE
x1=double(ncread(searise,'x1'));
y1=double(ncread(searise,'y1'));
thkmask=double(ncread(searise,'thkmask'));

%interpolate onto our mesh vertices
gridoniceshelf=double(¬InterpFromGridToMesh(x1,y1,thkmask',md.mesh.x,md.mesh.y,0));
clear thkmask;

%transfer to our mesh elements
elementoniceshelf=zeros(md.mesh.numberofelements,1);
elementoniceshelf(find(sum(gridoniceshelf(md.mesh.elements(:,:)),2)==3))=1;

%fill in the rest of the md.mask structure
md=setmask(md,elementoniceshelf,'');
```

Step 3: Parameterize Model 1/2 (runme.m)

Parameterize model using SeaRISE data from NetCDF: Exp_Par/Pig.par

- Geometry: surface, bedrock, thickness

```
disp('   Constructing thickness');
md.geometry.thickness=md.geometry.surface-md.geometry.bed;

%ensure hydrostatic equilibrium on ice shelf:
di=md.materials.rho_ice/md.materials.rho_water;
pos=find(md.mask.vertexonfloatingice);
md.geometry.thickness(pos)=1/(1-di)*md.geometry.surface(pos);
md.geometry.bed(pos)=md.geometry.surface(pos)-md.geometry.thickness(pos);
md.geometry.hydrostatic_ratio=ones(md.mesh.numberofvertices,1);

%Set min thickness to 1 meter
pos0=find(md.geometry.thickness≤0);
md.geometry.thickness(pos0)=1;
md.geometry.surface=md.geometry.thickness+md.geometry.bed;
```

- Temperatures

```
md.initialization.temperature=InterpFromGridToMesh(x1,y1,temp,md.mesh.x,md.mesh.y,0)+273.15+Temp_c
```

- Surface mass balance

```
md.surfaceforcings.mass_balance=InterpFromGridToMesh(x1,y1,smb,md.mesh.x,md.mesh.y,0);
md.surfaceforcings.mass_balance=md.surfaceforcings.mass_balance*md.materials.rho_water/md.material
```

Step 3: Parameterize Model 2/2 (runme.m)

- Basal friction parameters: (p,q)
- Ice rheology: ice rigidity (Patterson), Glen flow parameter

```
md.materials.rheology_n=3*ones(md.mesh.numberofelements,1);
md.materials.rheology_B=paterson(md.initialization.temperature);
```

- Observed velocity

```
vx_obs=InterpFromGridToMesh(x2,y2,flipud(velx'),md.mesh.x,md.mesh.y,0);
vy_obs=InterpFromGridToMesh(x2,y2,flipud(vely'),md.mesh.x,md.mesh.y,0);
```

- Specify ice flow model (runme.m)

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

Step 4: Control Method for Coefficient of Friction (runme.m)

Basal friction:

$$\tau_b = -\alpha^2 v_b$$

- Solve inverse method for α

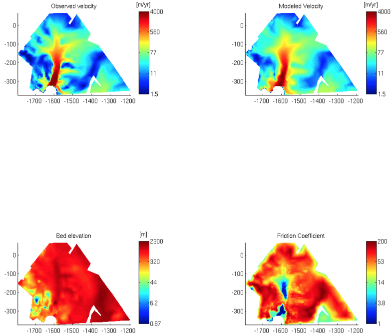
```
% Control general
md.inversion.iscontrol=1;
md.inversion.nsteps=20;
md.inversion.step_threshold=0.999*ones(md.inversion.nsteps,1);
md.inversion.maxiter_per_step=5*ones(md.inversion.nsteps,1);
md.verbose=verbose('solution',true,'control',true);

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

% Controls
md.inversion.control_parameters={'FrictionCoefficient'};
md.inversion.gradient_scaling=50*ones(md.inversion.nsteps,1);
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;

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

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

Step 5: Plot Results (runme.m)

Results in matlab: md.results.DiagnosticSolution

```
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.DiagnosticSolution.Vel,'title','Modeled Velocity',...
 'data',md.geometry.bed,'title','Bed elevation',...
 'data',md.results.DiagnosticSolution.FrictionCoefficient,'title','Friction Coefficient',...
 'colorbar#all','on','colorbartitle#1-2','[m/yr]',...
 'caxis#1-2',([1.5,4000]),...
 'colorbartitle#3','[m]', 'log#all',10);
```

Results:



Step 6-7: Blatter-Pattyn Ice Flow Model (runme.m)

Step 6: Blatter-Pattyn ice flow model

```
% Load Model
md = loadmodel('./Models/Pig.Control_drag');
md.inversion.iscontrol=0;

disp('   Extruding mesh')
number_of_layers=3;
md=extrude(md, number_of_layers, 0.9);

disp('   Using Batter-Pattyn Ice Flow Model')
md=setflowequation(md, 'pattyn', 'all');

% Solve
md=solve(md,DiagnosticSolutionEnum);
```

Step 7: Plot results

```
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.DiagnosticSolution.Vel,'title','Modeled Velocity',...
 'data',md.geometry.bed,'title','Bed elevation',...
 'data',md.friction.coefficient,'title','Friction Coefficient',...
 'caxis#1-2',([1.5,4000]),...
 'colorbar#all','on','view#all',2,...
 'colorbartitle#1-2','[m/yr]','colorbartitle#3','[m]',...
 'layer#1-3',(md.mesh.numberoflayers),'layer#4',1, 'log#all', 10);
```

Adjust Parameters (Exp_Par/Pig.par)

Effect of temperature, coefficient of friction: Try Varying!

```
% Parameters to change/Try
friction_coefficient = 10; % default [10]
Temp_change          =  0;  % default [0 K]
```

Thank you!

Any Questions?