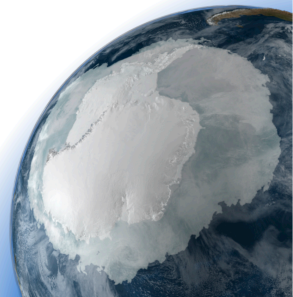


Application to Jakobshavn Isbrae, West Greenland

Eric LAROUR¹, Eric RIGNOT^{1,3}, Mathieu MORLIGHEM^{1,2}, H  l  ne
SEROUSSI^{1,2} **Chris BORSTAD**¹, Feras HABBAL^{1,3}, Daria HALKIDES^{1,4},
Behnaz KHAKBAZ¹, John SCHIERMEIER¹, Nicole SCHLEGEL¹

⁴ Joint Institute for Regional Earth System Science & Engineering, UCLA



Application
Jakobshavn

Larour et al.

Overview

Meshing

Paramaterization

Control Method
Solution

Plotting

① Meshing

② Paramaterization

③ Control Method Solution

④ Plotting

Application
Jakobshavn

Larour et al.

File runme.m

Meshing

Paramaterization

Control Method
Solution

Plotting

The **runme.m** file is a list of commands to be run in sequence at the MATLAB command prompt. Simply type "runme" at the command line to execute the commands.

- The variable *steps* defines which step(s) of the runme file to execute
- *printflag* can be switched to true for printing a plot to file
- *cluster* sets parameters for parallel computation
- *ncdata* defines the path to the SeaRISE netcdf file ("Greenland Developmental Data Set" at http://websrv.cs.umd.edu/isis/index.php/Present_Day_Greenland)

```
1 steps=[1 2 3 4];
2
3 %Hard coded parameters
4 cluster=generic('name', oshostname, 'np', 2);
5 ncdata='~/issmjpl/projects/ModelData/SeaRISE/Greenland5km_v1.2/Greenland_5km_dev1.2.nc';
```

Application
Jakobshavn

Larour et al.

File runme.m

Meshing

Paramaterization

Control Method
Solution

Plotting

Here we use *if* statement blocks to delineate different run steps. If the variable *steps* contains the value 1, the commands of the following block will be executed.

```
7 %Run Steps
8 if any(steps==1)
9
10     disp('    Step 1: Mesh creation');
```

Application
Jakobshavn

Larour et al.

First Run Step: Meshing

- 11** Create a triangle mesh with 2000 m resolution using the domain outline file "Jks3_trimmed.exp" and store it in the model named **md**.
- 14-20** Interpolated observed velocity data onto the newly-created mesh
- 23** Refine mesh in areas of higher observed velocity
- 26** Save model to file compatible with MATLAB v7.3 and later

```
8  if any(steps==1)
9
10     disp('    Step 1: Mesh creation');
11     md=triangle(model,'Jks3_trimmed.exp',2000);
12
13     %Get observed velocity field on mesh nodes
14     x1=double(ncread(ncdata,'x1'));
15     y1=double(ncread(ncdata,'y1'));
16     velx=double(ncread(ncdata,'surfvelx'));
17     vely=double(ncread(ncdata,'surfvely'));
18     vx=InterpFromGridToMesh(x1,y1,velx',md.mesh.x,md.mesh.y,0);
19     vy=InterpFromGridToMesh(x1,y1,vely',md.mesh.x,md.mesh.y,0);
20     vel=sqrt(vx.^2+vy.^2);
21
22     %refine mesh using surface velocities as metric
23     md=bamg(md,'hmin',1200,'hmax',15000,'field',vel,'err',5);
24
25     name='JKS.Mesh_generation';
26     save(name,'md','-v7.3')
27 end
```

Application
Jakobshavn
Larour et al.

First Run Step: Meshing

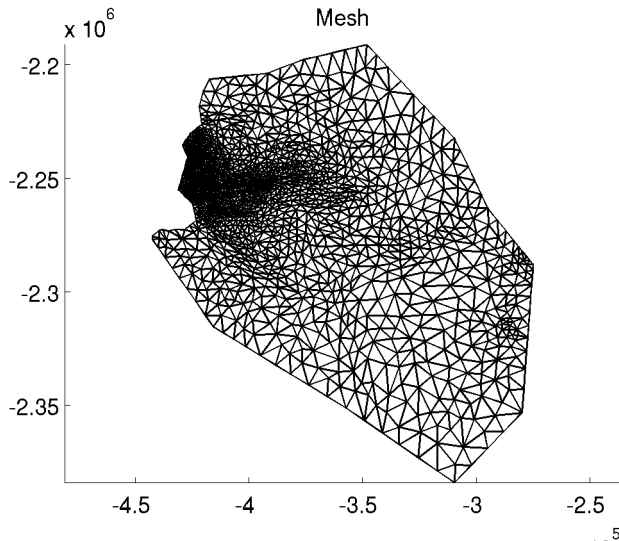
Mesh plot

Meshing

Paramaterization

Control Method
Solution

Plotting



Application
Jakobshavn

Larour et al.

Second Run Step: Parameterization

Meshing

Paramaterization

Control Method
Solution

Plotting

33 All ice is considered grounded. The **setmask** function can be used to define the boundary between grounded and floating ice.

34 Paramaterize the model using the .par file

37-39 The domain outline file "WeakB.exp" defines a shear margin with softer ice. In this region, reduce B by a factor of 0.3

```
28  if any(steps==2)
29
30      disp(' Step 2: Parameterization');
31      md=loadmodel('JKS.Mesh_generation');
32
33      md=setmask(md, '', '');
34      md=parameterize(md, 'Jak.par');
35
36      %zones of shear margin softening
37      weakb=ContourToMesh(md.mesh.elements,md.mesh.x,md.mesh.y, 'WeakB.exp', 'node', 2);
38      pos=find(weakb);
39      md.materials.rheology_B(pos)=.3*md.materials.rheology_B(pos);
40
41      name='JKS.Parameterization';
42      save(name, 'md', '-v7.3')
43  end
```

Application
Jakobshavn

Larour et al.

Contents of Jak.par File

Meshing

Paramaterization

Control Method
Solution

Plotting

Miscellaneous general parameters at head of file:

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Greenland parameters%%%%%%%%%
2
3  %Parameters for Jakobshavn tutorial
4
5  modeldatapath = ...
    ['~/issmjpl/projects/ModelData/SeaRISE/Greenland5km_v1.2'];
6
7  %Name and hemisphere
8  md.mesh.hemisphere='n';
9  md.miscellaneous.name='SeaRISEgreenland';
10
11 %some temporary parameters
12 di=md.materials.rho_ice/md.materials.rho_water;
```

Application
Jakobshavn

Larour et al.

Contents of Jak.par File

Topography

- 16-18** Read and store x and y coordinates of SeaRISE data grid.
- 20-21** Read upper ice surface topography, interpolate from SeaRISE grid to mesh nodes, store in **md.geometry**.
- 23-25** Do same for bed topography, then calculate ice thickness from surface and bed.
- 27-30** Find any places with negative thickness and set thickness to 1. Re-calculate surface data.

```
16      ncdata=[modeldatapath '/Greenland_5km_dev1.2.nc'];
17      x1=double(ncread(ncdata,'x1'));
18      y1=double(ncread(ncdata,'y1'));
19
20      surface=double(ncread(ncdata,'usrf'));
21      md.geometry.surface=InterpFromGridToMesh(x1,y1,surface',md.mesh.x,md.mesh.y,0);
22
23      bed=double(ncread(ncdata,'topg'));
24      md.geometry.bed=InterpFromGridToMesh(x1,y1,bed',md.mesh.x,md.mesh.y,0);
25      md.geometry.thickness=md.geometry.surface-md.geometry.bed;
26
27      pos0=find(md.geometry.thickness<=0);
28      md.geometry.thickness(pos0)=1;
29
30      md.geometry.surface=md.geometry.thickness+md.geometry.bed;
```

Contents of Jak.par File

Observed velocity

Meshing

Paramaterization

Control Method
Solution

Plotting

33-34 Read SeaRISE surface velocity components.

36-38 Interpolate observed velocity onto mesh nodes, calculate velocity magnitude, store in **md.inversion**.

40-43 Also store observed velocities in **md.initialization**.

```
32  disp('      reading velocities ');
33  velx=double(ncread(ncdata,'surfvelx'));
34  vely=double(ncread(ncdata,'surfvely'));
35
36  md.inversion.vx_obs=InterpFromGridToMesh(xl,y1,velx',md.mesh.x,md.mesh.y,0);
37  md.inversion.vy_obs=InterpFromGridToMesh(xl,y1,vely',md.mesh.x,md.mesh.y,0);
38  md.inversion.vel_obs=sqrt(md.inversion.vx_obs.^2+md.inversion.vy_obs.^2);
39
40  md.initialization.vx=md.inversion.vx_obs;
41  md.initialization.vy=md.inversion.vy_obs;
42  md.initialization.vz=zeros(md.mesh.numberofvertices,1);
43  md.initialization.vel=md.inversion.vel_obs;
```

Application
Jakobshavn
Larour et al.

Contents of Jak.par File

Friction and Temperature

Meshing

Paramaterization

Control Method
Solution

Plotting

46-48 Set initial friction coefficients

51-52 Read and interpolate SeaRISE temperature data to mesh nodes

```
45     disp('         friction coefficient');
46     md.friction.coefficient=50*ones(md.mesh.numberofvertices,1);
47     md.friction.q=ones(md.mesh.numberofelements,1);
48     md.friction.p=ones(md.mesh.numberofelements,1);
49
50     disp('         loading temperature ');
51     temp=double(ncread(ncdata,'surftemp'));
52     md.initialization.temperature=InterpFromGridToMesh(xl,yl,temp',md.mesh.x,md.mesh.y,0)+273.15;
```

Application
Jakobshavn
Larour et al.

Contents of Jak.par File

Flow law and surface mass balance

Meshing

Paramaterization

Control Method
Solution

Plotting

55-58 Set **B** as a function of temperature and **n** as a constant

49-50 Read and interpolate SeaRISE surface mass balance data to mesh nodes

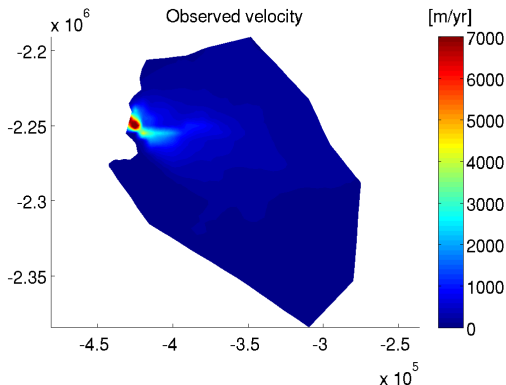
```
54  disp('      creating flow law paramters');
55  md.materials.rheology_B=paterson(md.initialization.temperature);
56  pos=find(md.materials.rheology_B≤0);
57  md.materials.rheology_B(pos)=-md.materials.rheology_B(pos);
58  md.materials.rheology_n=3*ones(md.mesh.numberofelements,1);
59
60  disp('      creating smb');
61  smb=double(ncread(ncdata,'smb'));
62  smb=InterpFromGridToMesh(x1,y1,smb',md.mesh.x,md.mesh.y,0);
63  md.surfaceforcings.mass_balance=[smb*1000/md.materials.rho_ice;1];
```

Application
Jakobshavn
Larour et al.

Third Run Step: Control Method for Basal Drag

Plot of observed velocity

This is a plot of the SeaRISE surface velocity data interpolated onto the mesh. This observed data is used in the control method.



Application
Jakobshavn

Larour et al.

Third Run Step: Control Method for Basal Drag

48 The **setflowequation** defines the material law for each element. In this case, use MacAyeal elements throughout.

51-55 Set general control method parameters.

58-60 Set cost functions

```
44  if any(steps==3)
45
46      disp(' Step 3: Control method friction');
47      md=loadmodel('JKS.Parameterization');
48      md=setflowequation(md,'macayeal','all');
49
50      %Control general
51      md.inversion.iscontrol=1;
52      md.inversion.nsteps=20;
53      md.inversion.step_threshold=0.999*ones(md.inversion.nsteps,1);
54      md.inversion.maxiter_per_step=15*ones(md.inversion.nsteps,1);
55      md.verbose=verbose('solution',true,'control',true);
56
57      %Cost functions
58      md.inversion.cost_functions_coefficients=ones(md.mesh.numberofvertices,1);
59      md.inversion.cost_functions(1:md.inversion.nsteps,1)=103;
60      md.inversion.cost_functions(floor(md.inversion.nsteps/2):md.inversion.nsteps,1)
```

Application
Jakobshavn

Larour et al.

Meshing

Parameterization

Control Method
Solution

Plotting

Third Run Step: Control Method for Basal Drag

63-73 Additional control parameters

76 Set cluster parameters for parallel solution

77-78 Solver parameters

79 Call solution sequence

```
62 %Controls
63 md.inversion.control_parameters={'FrictionCoefficient'};
64 md.inversion.gradient_scaling(1:md.inversion.nsteps)=50;
65 md.inversion.gradient_scaling(5:end)=10;
66 md.inversion.min_parameters=1*ones(md.mesh.numberofvertices,1);
67 md.inversion.max_parameters=200*ones(md.mesh.numberofvertices,1);
68
69 %Additional parameters
70 md.diagnostic.restol=0.01;
71 md.diagnostic.reltol=0.1;
72 md.diagnostic.abstol=NaN;
73 md.inversion.cost_function_threshold=NaN;
74
75 %Go solve
76 md=setcluster(md,cluster);
77 md.solver=addoptions(md.solver,NoneAnalysisEnum,asmoptions);
78 md.solver=addoptions(md.solver,DiagnosticVertAnalysisEnum,jacobiasmoptions);
79 md=solve(md,DiagnosticSolutionEnum);
80
81 name='JKS.Control_friction_macayear';
82 save(name,'md','-v7.3')
83 end
```

Application
Jakobshavn

Larour et al.

Fourth Run Step: Plotting Results

Meshing

Paramaterization

Control Method
Solution

Plotting

```
84  if any(steps==4)
85
86      disp(' Plotting')
87      md=loadmodel('JKS.Control_friction_macayeal');
88
89      plotmodel(md,'nlines',2,'ncols',2,'unit#all','km','axis#all','equal',...
90              'xlim#all',[min(md.mesh.x) max(md.mesh.x)]/10^3,...
91              'ylim#all',[min(md.mesh.y) max(md.mesh.y)]/10^3,...
92              'FontSize#all',12,...
93              'data',md.initialization.vel,'title','Observed velocity',...
94              'data',md.results.DiagnosticSolution.Vel,'title','Modeled Velocity',...
95              'colorbar#1','off','colorbar#2','on','colorbartitle#2','[m/yr]',...
96              'caxis#1-2',[0,7000],...
97              'data',md.geometry.bed,'title','Bed elevation',...
98              'data',md.results.DiagnosticSolution.FrictionCoefficient,...
99              'title','Friction Coefficient',...
100             'colorbar#3','on','colorbartitle#3','[m]','colorbar#4','on');
101
102  end
```

Application
Jakobshavn
Larour et al.

Fourth Run Step: Plotting Results

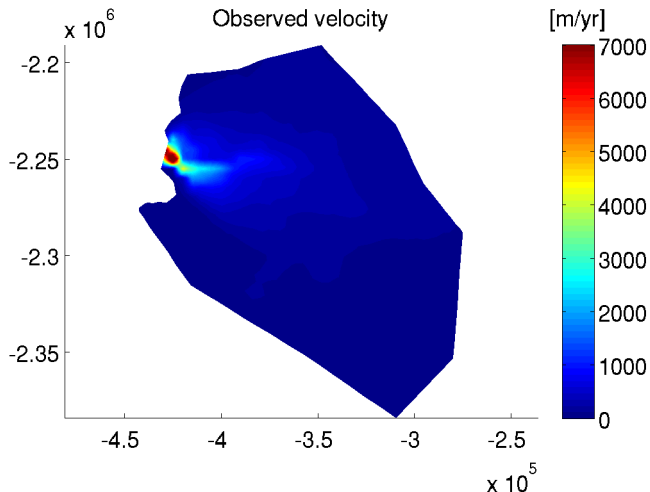
Observed velocity

Meshing

Paramaterization

Control Method
Solution

Plotting



Application
Jakobshavn
Larour et al.

Fourth Run Step: Plotting Results

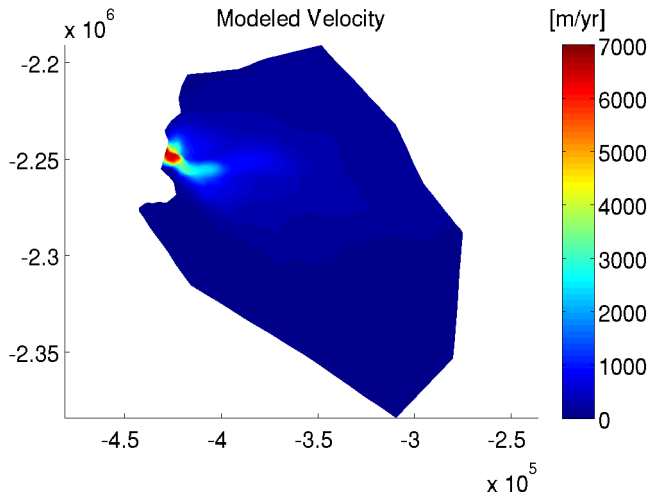
Modeled velocity

Meshing

Paramaterization

Control Method
Solution

Plotting



Application
Jakobshavn
Larour et al.

Fourth Run Step: Plotting Results

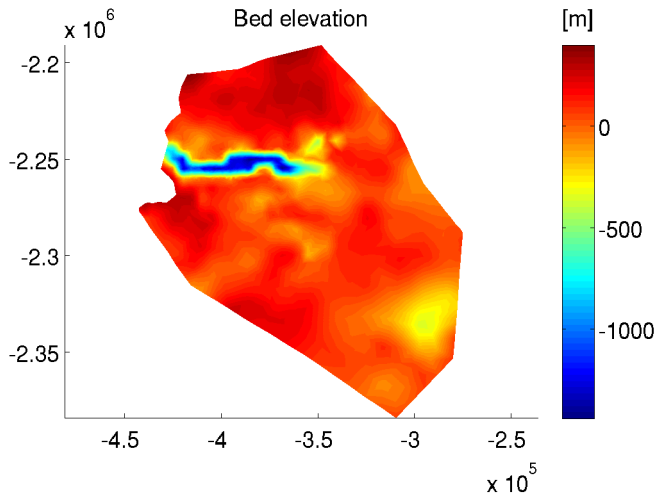
Bed elevation

Meshing

Paramaterization

Control Method
Solution

Plotting



Application
Jakobshavn
Larour et al.

Fourth Run Step: Plotting Results

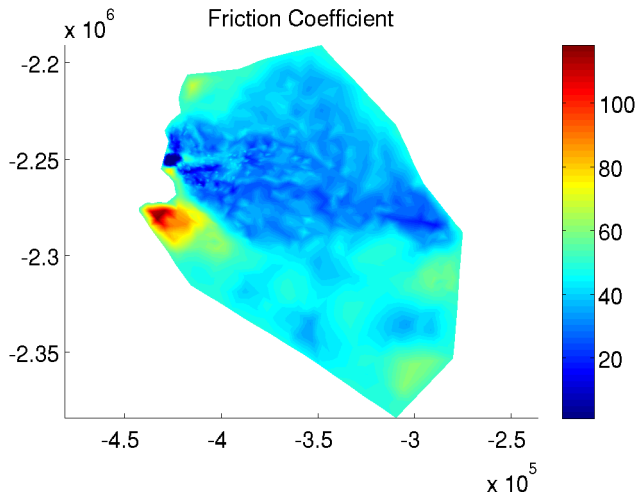
Modeled friction coefficient

Meshing

Paramaterization

Control Method
Solution

Plotting



Thanks!

