Ice Sheet System model

Application to SeaRISE dataset, Greenland Ice Sheet

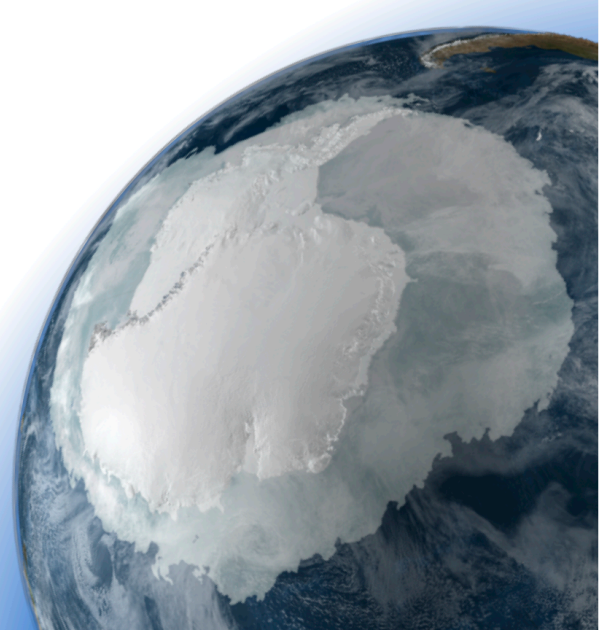
Eric LAROUR[1], Eric RIGNOT[1,3], Mathieu MORLIGHEM[1,2], Hélène SEROUSSI[1,2] Chris BORSTAD[1], Feras HABBAL[1,3], Daria HALKIDES[1,4], Behnaz KHAKBAZ[1], John SCHIERMEIER[1], **Nicole SCHLEGEL**[1]

[1]Jet Propulsion Laboratory - California Institute of Technology

[2]Laboratoire MSSMat, École Centrale Paris, France

[3]University of California, Irvine

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



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 velocities and thickness data
- Parameterize (as the Jks model), but specify velocity of at least one point to ensure non-sigularity.
- Diagnostic: run inverse method to control drag (50 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 bamg method as follows:

```
11    %Mesh parameters
12    domain =['./Exp_Par/DomainOutline.exp'];
13    hmax=400000;
14    hmin=80000;
15    gradation=1.7;
16    err=[15 200];
17
18    %Initial mesh creation
19    disp('   Interpolating fields');
20    md=bamg(model,'domain',domain,'hmax',hmin,'splitcorner',1,'KeepVertices',0);
```

This creates a new model named md and meshes the model domain at a resolution of 80000 m.

Mesh, 2/3

Adapt

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

```
5   modeldatapath=[issmdir '/projects/ModelData/SeaRISE/Greenland5km_v1.2'];
```

Adapt:

- Fill md.inversion.vx_obs, md.inversion.vy_obs, and md.inversion.vel_obs with interpolated velocities
- Fill md.geometry.thickness with interpolated thickness
- Mesh adapt your model (bamg)
  - Use the fields md.inversion.vel_obs and md.geometry.thickness
  - Set hmax=400000m and hmin=80000m
- 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

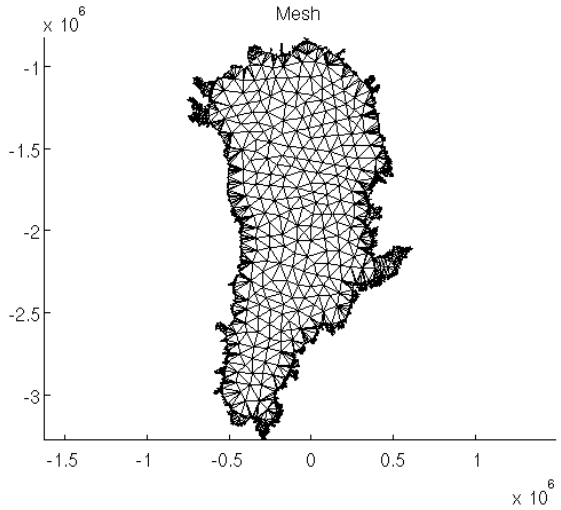
Mesh, 3/3

Check Your Work

Solution:

```
39    md=bamg(md,'hmax',hmax,'hmin',hmin,'gradation',gradation,'field',...
40     [md.inversion.vel_obs md.geometry.thickness],'err',err,...
41     'splitcorner',1,'KeepVertices',0);
42    [md.mesh.lat,md.mesh.long]=xy2ll(md.mesh.x,md.mesh.y,+1,39,71);
43
44    name='./Models/Greenland.Mesh_generation';
45    save(name,'md');
```

Plot your mesh. It should look like:



Mesh

Full Solution

```
22  ncdata=[modeldatapath '/Greenland_5km_dev1.2.nc'];
23  %Use ncdisp('file') to see an ncdump
24  x1=double(ncread(ncdata,'x1'));
25  y1=double(ncread(ncdata,'y1'));
26
27  velx=double(ncread(ncdata,'surfvelx'));
28  vely=double(ncread(ncdata,'surfvely'));
29
30  md.inversion.vx_obs=InterpFromGridToMesh(x1,y1,velx',md.mesh.x,md.mesh.y,0);
31  md.inversion.vy_obs=InterpFromGridToMesh(x1,y1,vely',md.mesh.x,md.mesh.y,0);
32  md.inversion.vel_obs=sqrt(md.inversion.vx_obs.^2+md.inversion.vy_obs.^2);
33
34  thickness=double(ncread(ncdata,'thk'));
35  md.geometry.thickness=InterpFromGridToMesh(x1,y1,thickness',...
36   md.mesh.x,md.mesh.y,0);
37
38  disp('   Mesh adaptation');
39  md=bamg(md,'hmax',hmax,'hmin',hmin,'gradation',gradation,'field',...
40   [md.inversion.vel_obs md.geometry.thickness],'err',err,...
41   'splitcorner',1,'KeepVertices',0);
42  [md.mesh.lat,md.mesh.long]=xy2ll(md.mesh.x,md.mesh.y,+1,39,71);
43
44  name='./Models/Greenland.Mesh_generation';
45  save(name,'md');
```

Parameterization

Call the setmask function with empty arguments and then parameterize your mesh with file Exp_Par/Greenland.par.

Read though the parameter file Exp_Par/Greenland.par, similar to your Jks par file.

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

```
 95    md.diagnostic.spcvx=NaN*ones(md.mesh.numberofvertices,1);
 96    md.diagnostic.spcvy=NaN*ones(md.mesh.numberofvertices,1);
 97    md.diagnostic.spcvz=NaN*ones(md.mesh.numberofvertices,1);
 98    md.diagnostic.referential=NaN*ones(md.mesh.numberofvertices,6);
 99
100    pos=find(md.inversion.vel_obs==0);
101    md.diagnostic.spcvx(pos)=0;
102    md.diagnostic.spcvy(pos)=0;
103    md.diagnostic.spcvz(pos)=0;
```

Parameterization

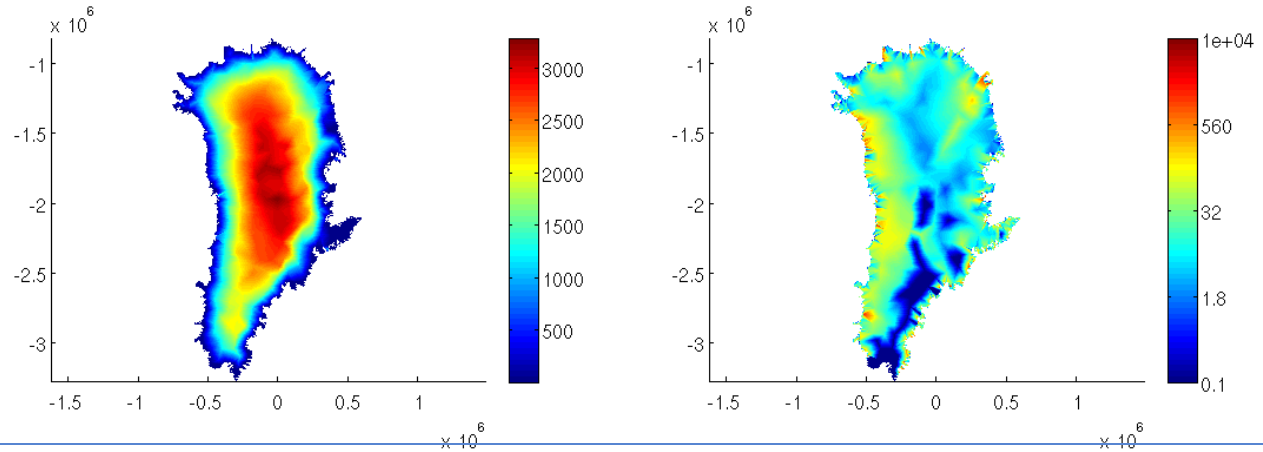
Solution

```
52    md=setmask(md,'','');
53    md=parameterize(md,'./Exp_Par/Greenland.par');
54
55    name='./Models/Greenland.Parameterization';
56    save(name,'md');
```

Plot thickness and velocity.

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

They should look like:



Diagnostic

Use control methods to inversely solve for Greenland FrictionCoefficient

Steps:

- Set cost functions
  - log of surface velocity
  - drag coefficient gradient
- Set cost functions coefficient to $8*10^{-15}$
- Specify max inversion parameter = 200, min inversion parameter = 1
- Solve a 50-step Diagnostic in 2D, Macayeal
- Copy result Vx, Vy, and Friction Coefficient to model values
- Save your model

Startoff Code:

```
62    md=setflowequation(md,'macayeal','all');
63
64    md.diagnostic.restol=0.01;
65    md.diagnostic.reltol=0.1;
66
67    md=parametercontroldrag(md,'nsteps',50);
68    md.inversion.step_threshold=.99*ones(md.inversion.nsteps,1);
69    md.inversion.gradient_scaling(:)=100;
```

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

Diagnostic

Solution

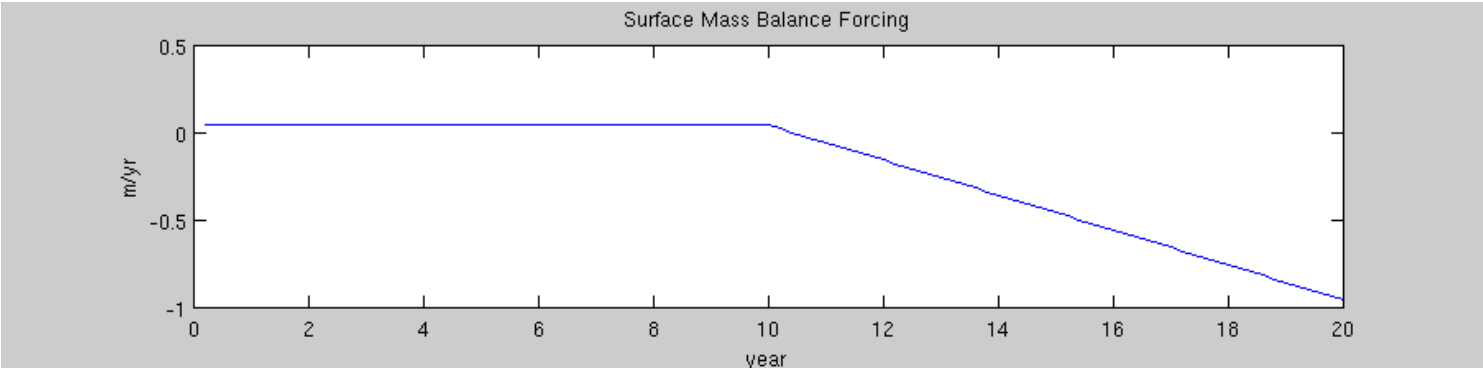
```
71   md.inversion.cost_functions=[];
72   md.inversion.cost_functions(1:floor(md.inversion.nsteps),1)=103;
73   md.inversion.cost_functions=[103*ones(md.inversion.nsteps,1)   ...
74    501*ones(md.inversion.nsteps,1)];
75   md.inversion.cost_functions_coefficients=ones(md.mesh.numberofvertices,2);
76   md.inversion.cost_functions_coefficients(:,2)=8*10^-15;
77   md.inversion.control_parameters={'FrictionCoefficient'};
78   md.inversion.min_parameters=1*ones(md.mesh.numberofvertices,1);
79   md.inversion.max_parameters=200*ones(md.mesh.numberofvertices,1);
80
81   md=setcluster(md,cluster);
82   md.inversion.iscontrol=1;
83   md.verbose=verbose('solution',true,'control',true);
84   md=solve(md,DiagnosticSolutionEnum);
85
86   md.friction.coefficient=md.results.DiagnosticSolution.FrictionCoefficient;
87   md.initialization.vx=md.results.DiagnosticSolution.Vx;
88   md.initialization.vy=md.results.DiagnosticSolution.Vy;
89
90   name='./Models/Greenland.Control_drag';
91   save(name,'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:

```
1   >> md.surfaceforcings.mass_balance = [ smb smb-1];
2   >> 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.

Transient

Solve

Set up your transient

Steps:

- Set zero basal melting rate
- Interpolate surface mass balance from SeaRISE dataset
- 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:

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

Transient

Solution Setup

```
 98  md=setflowequation(md,'macayeal','all');
 99  md.basalforcings.melting_rate=zeros(md.mesh.numberofvertices,1);
100
101  ncdata=[modeldatapath '/Greenland_5km_dev1.2.nc'];
102  x1=double(ncread(ncdata,'x1'));
103  y1=double(ncread(ncdata,'y1'));
104
105  smb=double(ncread(ncdata,'smb'));
106  smb=InterpFromGridToMesh(x1,y1,smb',md.mesh.x,md.mesh.y,0)*...
107   1000/md.materials.rho_ice;
108  smb=[smb smb smb-1.0];
109  md.surfaceforcings.mass_balance=[smb; [1 10 20]];
110
111  md.thermal.spctemperature=[md.initialization.temperature;1];
112
113  md.timestepping.time_step=0.2;
114  md.timestepping.final_time=20;
115  md.settings.output_frequency=1;
116
117  md.inversion.iscontrol=0;
118  md.transient.requested_outputs=IceVolumeEnum();
119
120  md=setcluster(md,cluster);
121  md.verbose=verbose('solution',true,'module',true,'convergence',true);
122
123  md=solve(md,TransientSolutionEnum);
124
125  name='./Models/Greenland.Transient';
126  save(name,'md');
```

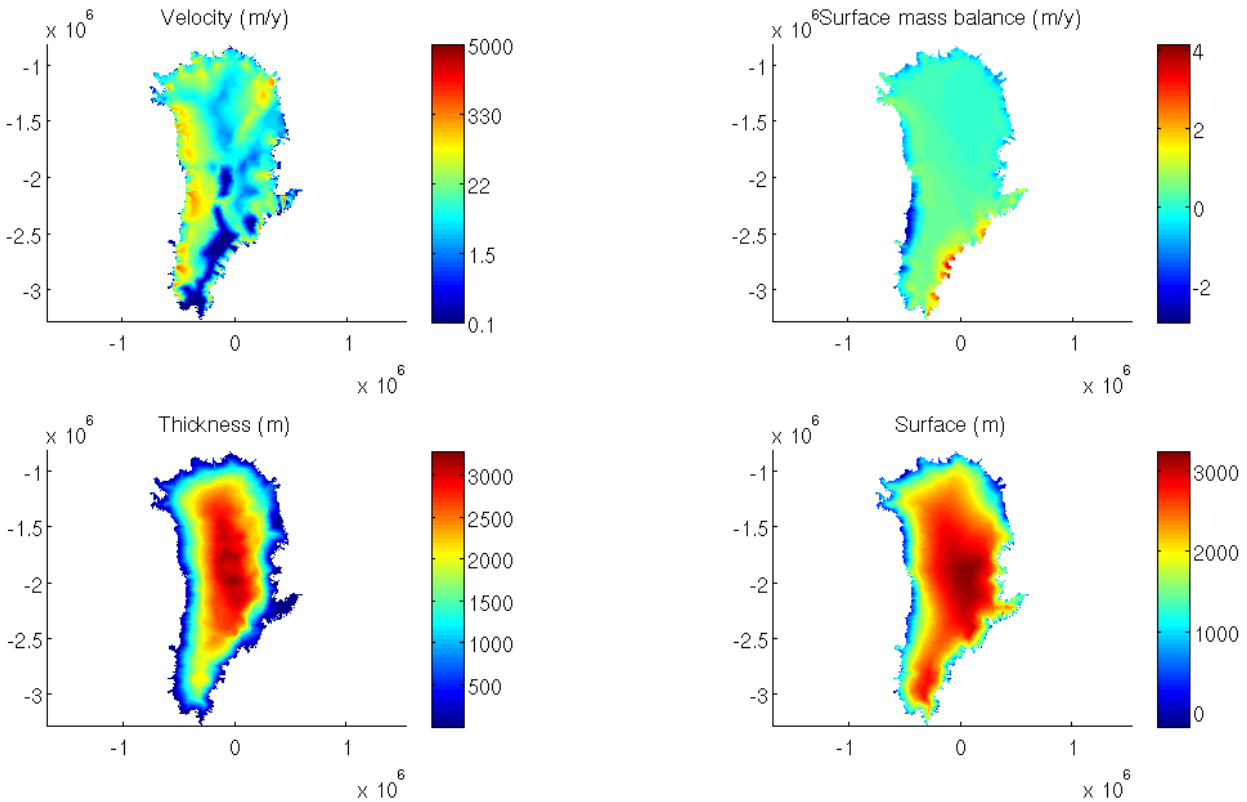
Transient Results

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 one row of three subplots.

They should look like:



Plot

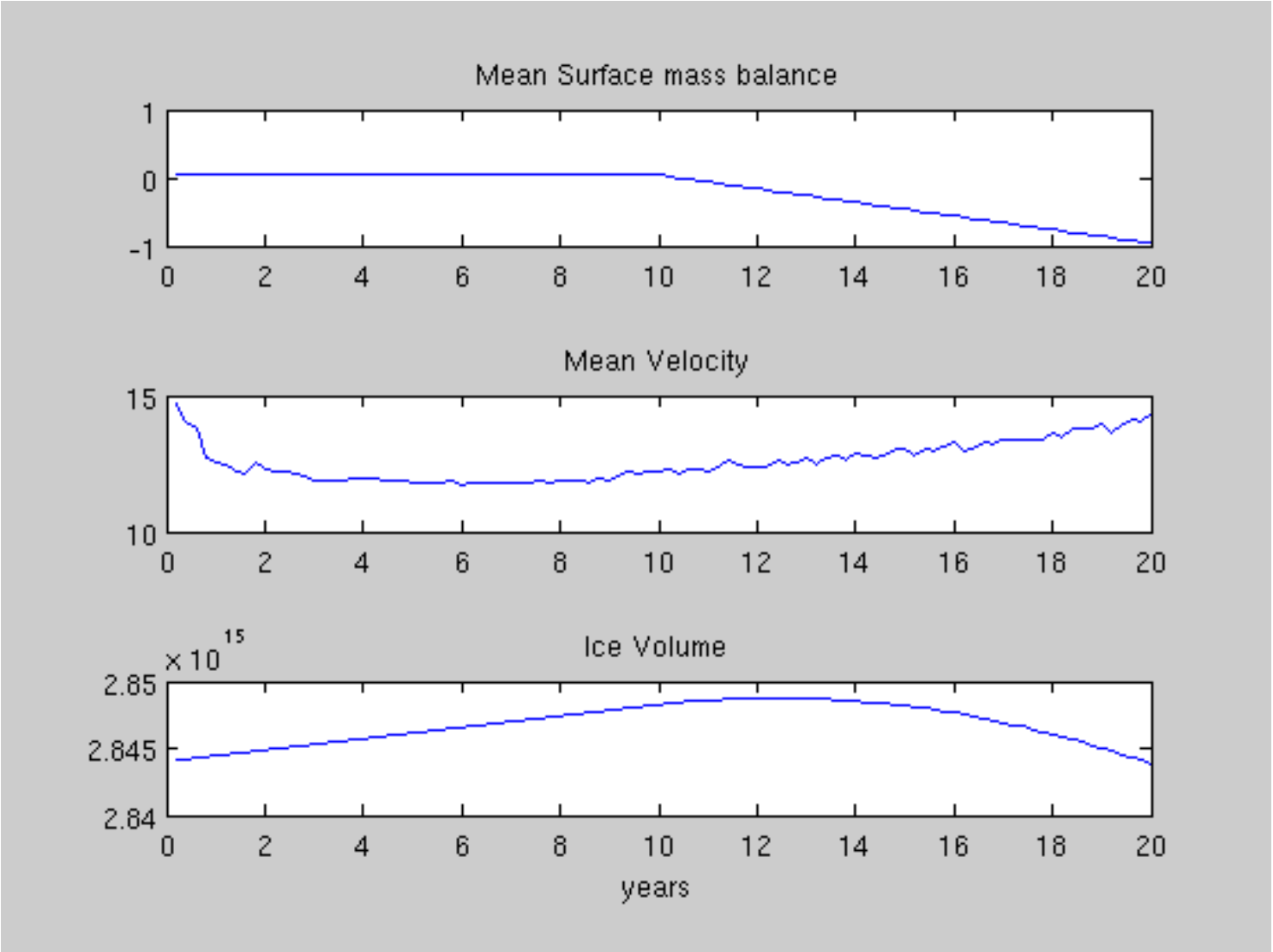
Solution

```
134    plotmodel(md,'data',md.results.TransientSolution(end).Vel,'caxis',[1e-1 ...
           5000],...
135     'log', 10, 'title','Velocity (m/y)', ...
136     'data', md.results.TransientSolution(1).SurfaceforcingsMassBalance, ...
137     'title', 'Surface mass balance (m/y)', ...
138     'data',md.results.TransientSolution(end).Thickness,...
139     'title','Thickness (m)', ...
140     'data',md.results.TransientSolution(end).Surface, ...
141     'title', 'Surface (m)');
```

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

```
145    figure
146
147    %Plot surface mass balance
148    surfmb=[]; for i=1:100; surfmb=[surfmb ...
149     md.results.TransientSolution(i).SurfaceforcingsMassBalance]; end
150    subplot(3,1,1); plot([0.2:0.2:20],mean(surfmb)); title('Mean Surface ...
           mass balance');
```

Your results should look like:




Thanks!