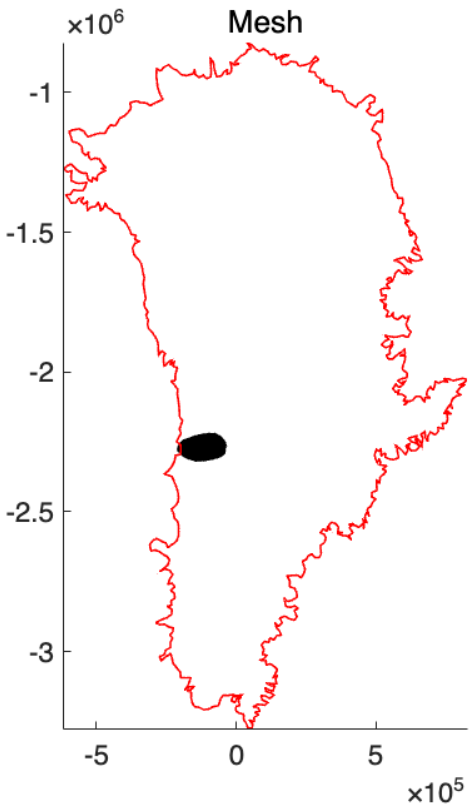
Modelling Jakobshavn Isbrae Glacier

```
% First capture the geometry from shapefile - convert .shp (GIS) to .exp
cd '/Users/rishi/Desktop/ISSM/examples/Jakobshavn'
shp2exp('Jakobshavn.shp','Jakobshavn.exp');

% Use the .exp file to make the triangular mesh
md = triangle(model,'/Users/rishi/Desktop/ISSM/examples/Jakobshavn/
Jakobshavn.exp',2000);

% plot the mesh with the greenland bondary to observed its location
plotmodel(md,'data','mesh')
hold on
expdisp('/Users/rishi/Desktop/ISSM/examples/Helheim/GreenlandOutline.exp')
```

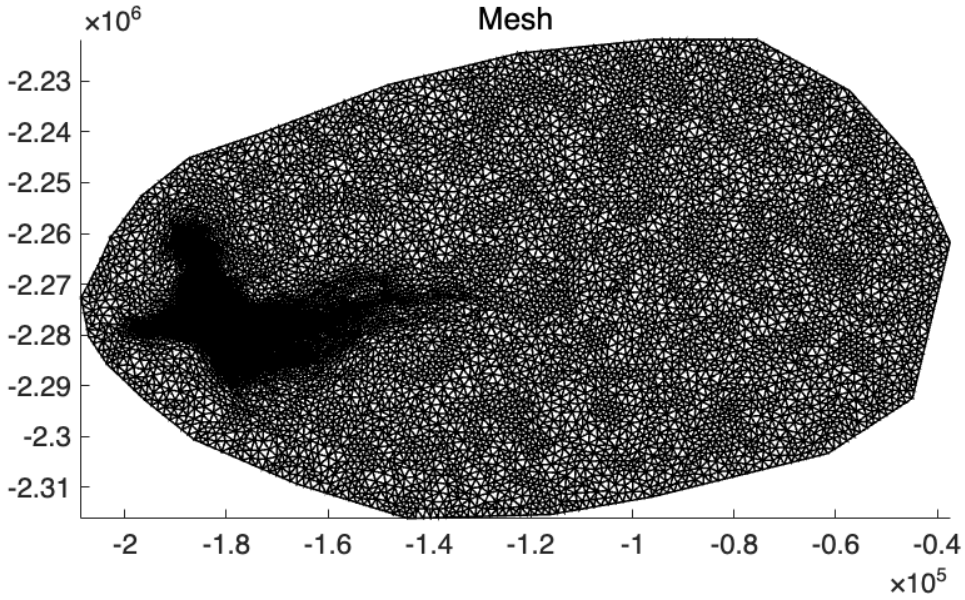


Interpolate observed surface velocities onto the current mesh

```
[velx, vely]=interpJoughinCompositeGreenland(md.mesh.x,md.mesh.y,...
    '/Users/rishi/Desktop/ISSM/examples/Data/');
vel = sqrt(velx.^2+vely.^2);

% Refine mesh based on surface velocitie s
md=bamg(md,'hmin',200,'hmax',1500,'field',vel,'err',5);
```

```
Anisotropic mesh adaptation
WARNING: mesh present but no geometry found. Reconstructing...
   new number of triangles = 18303
```

```
[md.mesh.lat,md.mesh.long] = xy2ll(md.mesh.x,md.mesh.y,+1,45,70);
md.mesh.epsg=3413;

% plot the mesh to check the triangle adjustment
plotmodel(md,'data','mesh')
```



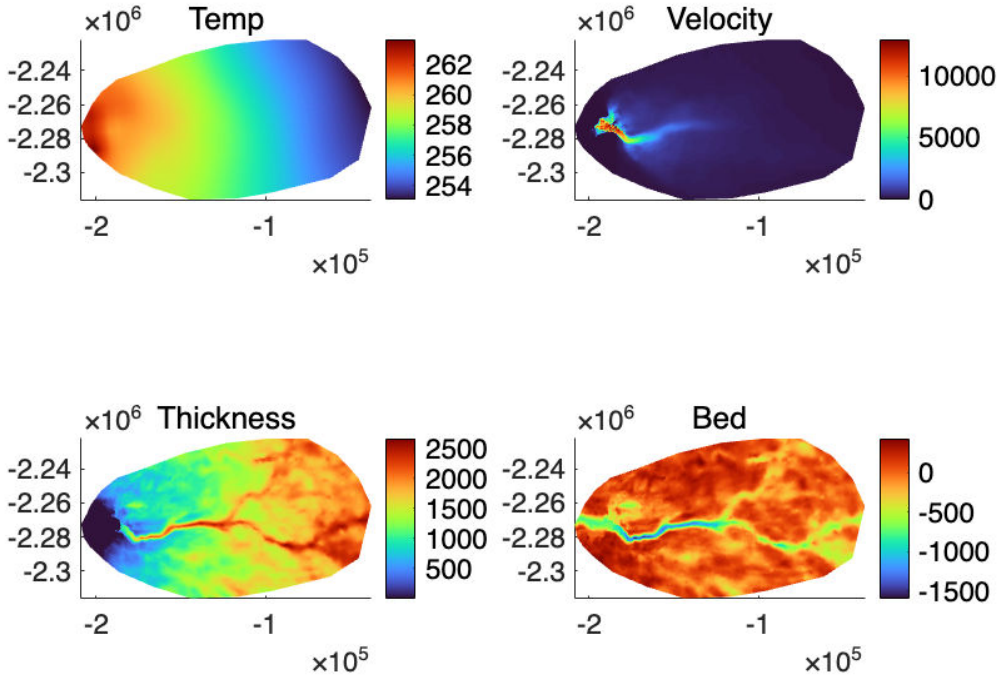
Parameterization of the model

```
% Step - 1 Interpolate the datasets from the Bedmachine-Greenland-v5
md=setflowequation(md,'SSA','all'); % Set flow law to SSA for original 2d
mesh
md=setmask(md,'','');
md=parameterize(md,'/Users/rishi/Desktop/ISSM/examples/Jakobshavn/
Greenland.par'); % can be used for any Greenland glacier
```

```
   Interpolating mask
   -- BedMachine Greenland version: v5
   -- BedMachine Greenland: loading mask
   -- BedMachine Greenland: interpolating mask
       -- Interpolation method: nearest
      reading MC bed (assumes no floating ice)
   -- BedMachine Greenland version: v5
   -- BedMachine Greenland: loading bed
   -- BedMachine Greenland: interpolating bed
```

```
    -- Interpolation method: cubic
   reading Howat surface
-- BedMachine Greenland version: v5
-- BedMachine Greenland: loading surface
-- BedMachine Greenland: interpolating surface
    -- Interpolation method: cubic
Adjusting ice mask
   -- reconstruct thickness
   reading velocities
Creating flow law parameters (assume ice is at 0°C for now)
Geothermal flux from Shapiro et al.
Setting up thermal model
Set Boundary conditions
```

```
md.miscellaneous.name = 'Jacob';


% Plot the parameters interpolated from the input data
plotmodel(md,'data', ...
    md.initialization.temperature,'title','Temp', ...
    'data', md.initialization.vel,'title','Velocity', ...
    'data',md.geometry.thickness,'title','Thickness', ...
    'data',md.geometry.bed,'title','Bed')
```



Model parameterters for running ISSM - Greenland

```
%Control general
md.inversion=m1qn3inversion(md.inversion);
md.inversion.iscontrol=1;
```

```
md.verbose=verbose('solution',false,'control',true);
md.transient.amr_frequency = 0;


%Cost functions
md.inversion.cost_functions=[101 501];
md.inversion.cost_functions_coefficients=zeros(md.mesh.numberofvertices,2);
md.inversion.cost_functions_coefficients(:,1)=50;
md.inversion.cost_functions_coefficients(:,2)= 2e-9;
pos=find(md.mask.ocean_levelset<0 | md.mask.ice_levelset>0);
md.inversion.cost_functions_coefficients(pos,1:2)=0;
md.stressbalance.spcvx(pos)=md.initialization.vx(pos);
md.stressbalance.spcvy(pos)=md.initialization.vy(pos);


%Controls
md.inversion.control_parameters={'FrictionCoefficient'}; % parameters to
invert
md.inversion.maxsteps=100; % maximum steps of inversion
md.inversion.maxiter =100; % maximum iterations
md.inversion.min_parameters=0.05*ones(md.mesh.numberofvertices,1);
md.inversion.max_parameters=300*ones(md.mesh.numberofvertices,1);
md.inversion.control_scaling_factors=1;


%Additional parameters
md.stressbalance.restol= 0.01;
md.stressbalance.reltol= 0.1;
md.stressbalance.abstol= NaN;
md.initialization.pressure=zeros(md.mesh.numberofvertices,1);


%Go solve
md.cluster=generic('name',oshostname,'np',4);
md=solve(md,'sb');
```

```
uploading input file and queuing script
launching solution sequence on remote cluster

Ice-sheet and Sea-level System Model (ISSM) version  4.24
(website: http://issm.jpl.nasa.gov forum: https://issm.ess.uci.edu/forum/)

call computational core:
   Initialize M1QN3 parameters
   Computing initial solution
```

Iter	Cost function	Grad. norm	List of contributions
1	f(x)= 146.37	0.585	146.4 5.341e-35
2	f(x)= 48.118	0.0936	48.12 6.481e-05
3	f(x)= 39.301	0.0683	39.3 8.449e-05
4	f(x)= 23.388	0.0564	23.39 0.0001908
5	f(x)= 16.65	0.0424	16.65 0.0004199

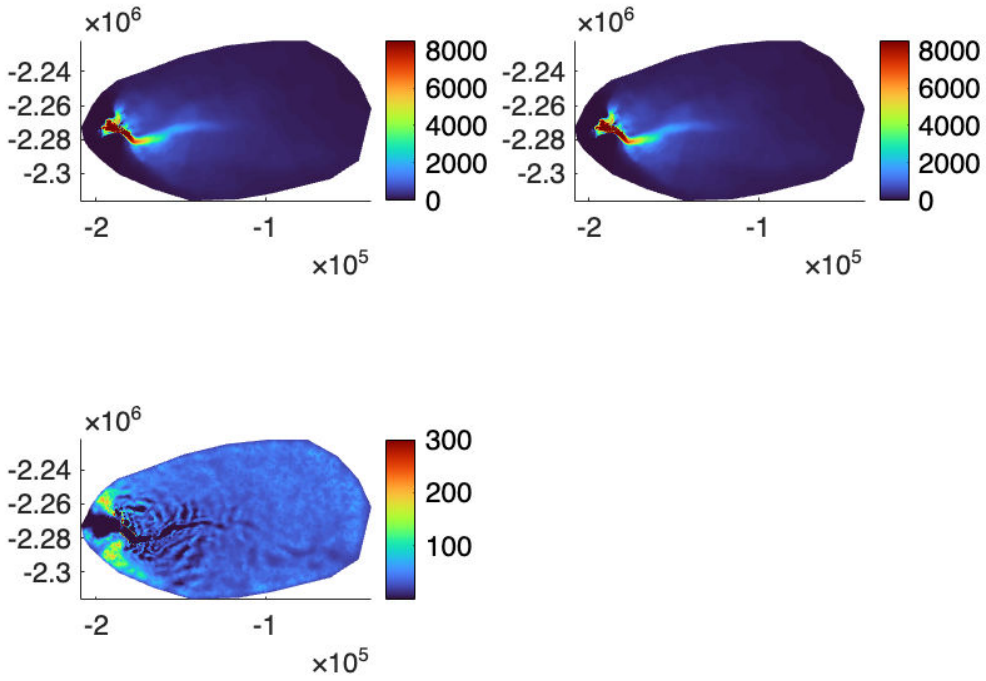
6	f(x)=	13.318	0.025	13.32	0.0004927
7	f(x)=	11.067	0.0248	11.07	0.0006574
8	f(x)=	10.308	0.0136	10.31	0.0007298
9	f(x)=	9.4613	0.00765	9.46	0.0008889
10	f(x)=	9.0675	0.00788	9.067	0.0009839
11	f(x)=	8.1679	0.00833	8.167	0.001214
12	f(x)=	7.8309	0.0124	7.83	0.001363
13	f(x)=	7.5189	0.00442	7.518	0.001307
14	f(x)=	7.3684	0.00328	7.367	0.001304
15	f(x)=	7.133	0.00329	7.132	0.001375
16	f(x)=	6.9404	0.00514	6.939	0.001483
17	f(x)=	6.772	0.00353	6.77	0.001617
18	f(x)=	6.6769	0.00254	6.675	0.001708
19	f(x)=	6.5379	0.00203	6.536	0.001894
20	f(x)=	6.4532	0.00178	6.451	0.001988
21	f(x)=	6.327	0.00211	6.325	0.002298
22	f(x)=	6.2824	0.00186	6.28	0.002354
23	f(x)=	6.2462	0.00152	6.244	0.002459
24	f(x)=	6.1828	0.003	6.18	0.002673
25	f(x)=	6.1676	0.00161	6.165	0.002713
26	f(x)=	6.1373	0.0016	6.134	0.002905
27	f(x)=	6.1142	0.00158	6.111	0.003017
28	f(x)=	6.0805	0.000915	6.077	0.003296
29	f(x)=	6.0639	0.000982	6.061	0.003409
30	f(x)=	6.038	0.00126	6.034	0.00367
31	f(x)=	6.0127	0.00101	6.009	0.003892
32	f(x)=	6.0019	0.000758	5.998	0.004067
33	f(x)=	5.9844	0.000699	5.98	0.004286
34	f(x)=	5.9704	0.000744	5.966	0.004424
35	f(x)=	5.9559	0.000639	5.951	0.004623
36	f(x)=	5.943	0.000549	5.938	0.004784
37	f(x)=	5.9305	0.000879	5.925	0.005049
38	f(x)=	5.9214	0.000562	5.916	0.005105
39	f(x)=	5.9104	0.000507	5.905	0.005321
40	f(x)=	5.9014	0.000552	5.896	0.005427
41	f(x)=	5.8906	0.000644	5.885	0.005663
42	f(x)=	5.8863	0.000418	5.881	0.00564
43	f(x)=	5.8811	0.000899	5.875	0.006092
44	f(x)=	5.8759	0.000495	5.87	0.005927
45	f(x)=	5.8671	0.000695	5.861	0.006157
46	f(x)=	5.8626	0.000337	5.856	0.006187
47	f(x)=	5.8601	0.00054	5.854	0.006294
48	f(x)=	5.8583	0.000614	5.852	0.006398
49	f(x)=	5.8512	0.000429	5.844	0.006674
50	f(x)=	5.8433	0.000361	5.836	0.007038
51	f(x)=	5.8381	0.000352	5.831	0.007253
52	f(x)=	5.8346	0.000402	5.827	0.007391
53	f(x)=	5.8322	0.000265	5.825	0.007512
54	f(x)=	5.8305	0.000257	5.823	0.007567
55	f(x)=	5.8265	0.000415	5.819	0.007761
56	f(x)=	5.8215	0.000265	5.814	0.007882
57	f(x)=	5.8165	0.000275	5.808	0.008148
58	f(x)=	5.8125	0.000268	5.804	0.008247
59	f(x)=	5.809	0.000233	5.8	0.008508
60	f(x)=	5.8055	0.000268	5.797	0.008684
61	f(x)=	5.8008	0.000302	5.792	0.008954
62	f(x)=	5.7987	0.000249	5.79	0.008958
63	f(x)=	5.7957	0.000407	5.786	0.009328
64	f(x)=	5.7921	0.000298	5.783	0.009084
65	f(x)=	5.7896	0.000686	5.78	0.009419
66	f(x)=	5.7842	0.000293	5.775	0.009428
67	f(x)=	5.7825	0.000206	5.773	0.009544
68	f(x)=	5.7821	0.00019	5.772	0.009591
69	f(x)=	5.7787	0.000306	5.769	0.009914

70	f(x)=	5.7754	0.000188	5.765 0.01006
71	f(x)=	5.7729	0.000186	5.763 0.01023
72	f(x)=	5.7711	0.00019	5.761 0.01038
73	f(x)=	5.7677	0.000317	5.757 0.01065
74	f(x)=	5.7663	0.000184	5.756 0.01072
75	f(x)=	5.7647	0.000175	5.754 0.0108
76	f(x)=	5.7634	0.000186	5.753 0.01076
77	f(x)=	5.7905	0.00166	5.779 0.0115
78	f(x)=	5.7706	0.000324	5.76 0.01085
79	f(x)=	5.7667	0.000223	5.756 0.0108
80	f(x)=	5.7652	0.000204	5.754 0.0108
81	f(x)=	5.7648	0.000199	5.754 0.0108
82	f(x)=	5.7647	0.000197	5.754 0.01079
83	f(x)=	5.7646	0.000196	5.754 0.01079
84	f(x)=	5.7646	0.000197	5.754 0.01078
85	f(x)=	5.7646	0.000197	5.754 0.01078
86	f(x)=	5.7646	0.000197	5.754 0.01077
87	f(x)=	5.7646	0.000197	5.754 0.01077
88	f(x)=	5.7646	0.000197	5.754 0.01077
89	f(x)=	5.7646	0.000197	5.754 0.01077

```
   stopped on dxmin during line search
   preparing final solution
write lock file:

   FemModel initialization elapsed time:   0.077556
   Total Core solution elapsed time:       65.1295
   Linear solver elapsed time:             45.1619 (69%)

   Total elapsed time: 0 hrs 1 min 5 sec
```

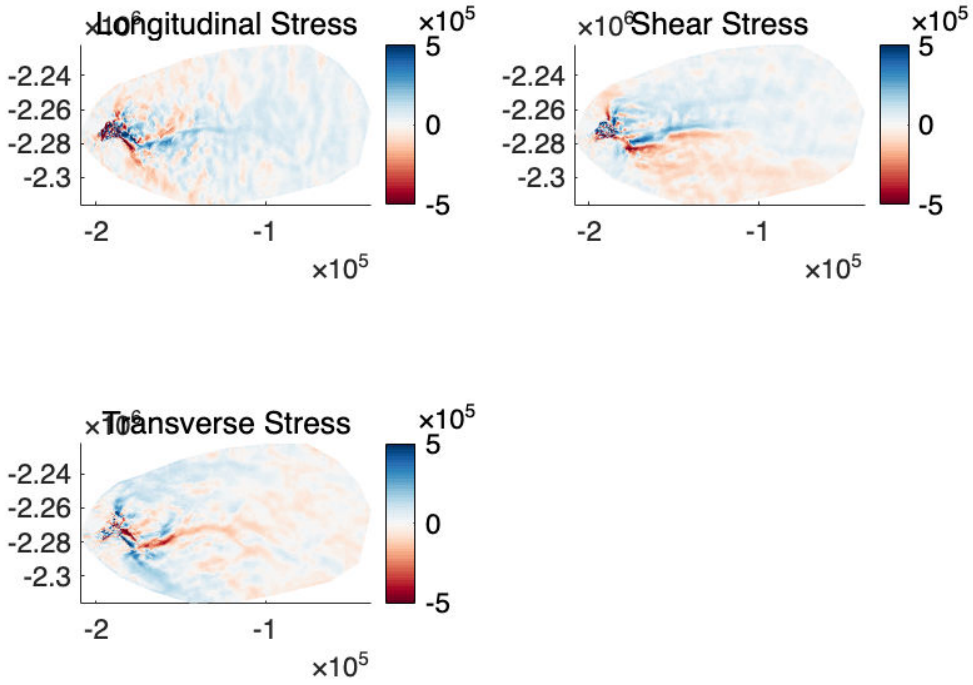
Plot the oberved and modelled velocity with friction coefficient

```
plotmodel(md,'data', md.initialization.vel,'caxis#1',[0 8500],...
'data',md.inversion.vel_obs,'caxis#2',[0 8500],...
'data',md.results.StressbalanceSolution.FrictionCoefficient)
```



Calculate Mechanical properties of ice - Stress and Strain

```
md = mechanicalproperties(md, md.results.StressbalanceSolution.Vx, ...
    md.results.StressbalanceSolution.Vy);

% plot the stresses
plotmodel(md,'data',md.results.deviatoricstress.xx,'caxis#1',[-5e5
5e5],'title','Longitudinal Stress', ...
    'data',md.results.deviatoricstress.xy,'caxis#2',[-5e5
5e5],'title','Shear Stress', ...
    'data',md.results.deviatoricstress.yy,'caxis#3',[-5e5
5e5],'title','Transverse Stress')

colormap(brewermap(50,'RdBu'))
```



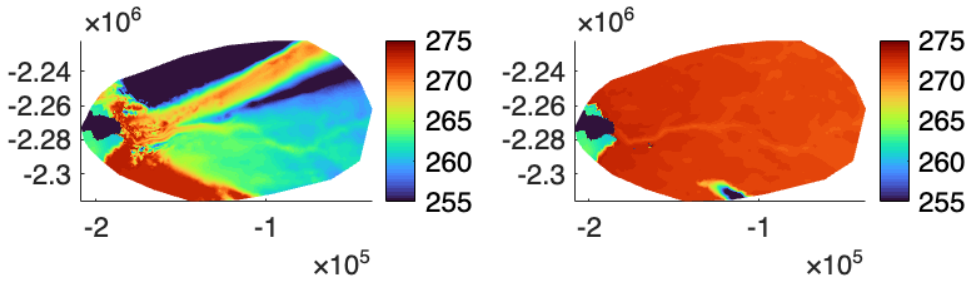
THERMAL STEADY STATE SOLUTION FOR GLACIER

```
% make the glacier layers for modelling
md=extrude(md,5,1.2);
md=setflowequation(md,'HO','all'); % Set flow law to HO for 3D mesh
md.thermal.fe = 'P1xP2';
md.inversion.iscontrol=0;


%Set single point constraints of surface temperature as boundary conditions
md.thermal.spctemperature(find(md.mesh.vertexonsurface))=...
    md.initialization.temperature(find(md.mesh.vertexonsurface));

%Initialize thermal paramertes
md.initialization.enthalpy = zeros(md.mesh.numberofvertices,1);
md.thermal.isdrainicecolumn=0;

%Run a steady state heat equation solution
md.timestepping.time_step=0;
md.cluster=generic('name',oshostname,'np',4);
md=solve(md,'thermal');
```

```
uploading input file and queuing script
launching solution sequence on remote cluster

Ice-sheet and Sea-level System Model (ISSM) version  4.24
```

```
(website: http://issm.jpl.nasa.gov forum: https://issm.ess.uci.edu/forum/)

call computational core:
write lock file:

   FemModel initialization elapsed time:   0.151572
   Total Core solution elapsed time:       8.6457
   Linear solver elapsed time:             5.53111 (64%)

   Total elapsed time: 0 hrs 0 min 8 sec
```

```
%Put results back into the model
md.initialization.temperature=md.results.ThermalSolution.Temperature;
md.initialization.enthalpy=md.results.ThermalSolution.Enthalpy;
md.materials.rheology_B = cuffey(md.initialization.temperature);

% Plot the surface temperature and temperature in basal layer
plotmodel(md,'data', md.initialization.temperature,'layer#1',4,'data', ...
md.initialization.temperature,'layer#2',1, 'caxis#all',[255 275])
```

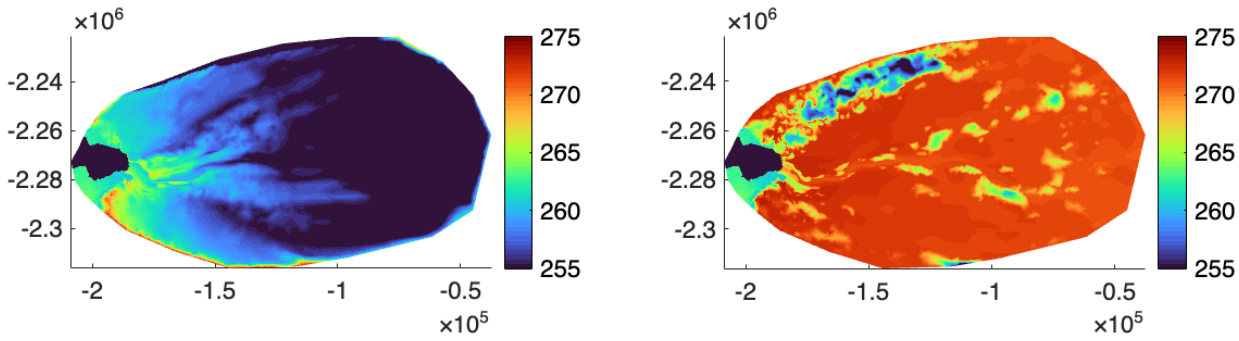



```
%Set thermal lateral boundary conditions on all ice that is entering the
domain.
%No single point constraints should be set on ice flowing out of the domain
% (i.e. ice sheet boundary or ocean locations)
md.thermal.spctemperature(md.mesh.vertexonboundary &
md.mask.ice_levelset<0)...
```

```
=md.initialization.temperature(md.mesh.vertexonboundary &
md.mask.ice_levelset<0);

%Turn off control for inversion
md.inversion.iscontrol=0;
md.timestepping.time_step=0;
md=solve(md,'ss');
```

```
uploading input file and queuing script
launching solution sequence on remote cluster

Ice-sheet and Sea-level System Model (ISSM) version  4.24
(website: http://issm.jpl.nasa.gov forum: https://issm.ess.uci.edu/forum/)

call computational core:
write lock file:

   FemModel initialization elapsed time:   0.264745
   Total Core solution elapsed time:       59.2983
   Linear solver elapsed time:             46.8285 (79%)

   Total elapsed time: 0 hrs 0 min 59 sec
```

```
%Put results back into the model
%Save steady state results in intitalization temperature
md.initialization.temperature = md.results.SteadystateSolution.Temperature;

%Save steady state results in intitalization enthalpy
md.initialization.enthalpy = md.results.SteadystateSolution.Enthalpy;

%Call the cuffey function to calculate materials rheology_B and save it in
the model
md.materials.rheology_B = cuffey(md.initialization.temperature);

plotmodel(md,'figposition','fullscreen','data',
md.initialization.temperature,...
'layer#1',4,'data',md.initialization.temperature,'layer#2',1,'caxis#all',
[255 275])
```




```
%Save the model in a temporary variable.
md3d=md;
%Collapse the new 3D model variable to 2D and save it in md.
md=collapse(md3d);
%Set the new flow equation to SSA for a 2D model
md=setflowequation(md,'SSA','all'); % Set flow law to SSA for original 2d
mesh
%Depth average the 3D model temperature, run the cuffey algorithm to
calcuate a new 2D rheology_B
md.materials.rheology_B=cuffey(DepthAverage(md3d,md3d.results.SteadystateSol
ution.Temperature));
%Restrict the maximum rheology_B to 4*10^8
md.materials.rheology_B(find(md.materials.rheology_B>4*10^8))=4*10^8;

%Cost functions
md.inversion.cost_functions=[101 103 501];
md.inversion.cost_functions_coefficients=zeros(md.mesh.numberofvertices,3);
md.inversion.cost_functions_coefficients(:,1)=50;
md.inversion.cost_functions_coefficients(:,2)=3;
md.inversion.cost_functions_coefficients(:,3)= 1e-7;
pos=find(md.mask.ocean_levelset<0 | md.mask.ice_levelset>0);
md.inversion.cost_functions_coefficients(pos,:)=0;
```

```
md.stressbalance.spcvx(pos)=md.initialization.vx(pos);
md.stressbalance.spcvy(pos)=md.initialization.vy(pos);

%Controls
md.inversion.control_parameters={'FrictionCoefficient'};
md.inversion.maxsteps=100;
md.inversion.maxiter =100;
md.inversion.min_parameters=0.05*ones(md.mesh.numberofvertices,1);
md.inversion.max_parameters=300*ones(md.mesh.numberofvertices,1);
md.inversion.control_scaling_factors=1;

%Control general
md.inversion.iscontrol=1;
md.verbose=verbose('solution',false,'control',true);

%Go solve
md.cluster=generic('name',oshostname,'np',4);
md=solve(md,'sb');
```

```
uploading input file and queuing script
launching solution sequence on remote cluster

Ice-sheet and Sea-level System Model (ISSM) version  4.24
(website: http://issm.jpl.nasa.gov forum: https://issm.ess.uci.edu/forum/)

call computational core:
   Initialize M1QN3 parameters
   Computing initial solution
```

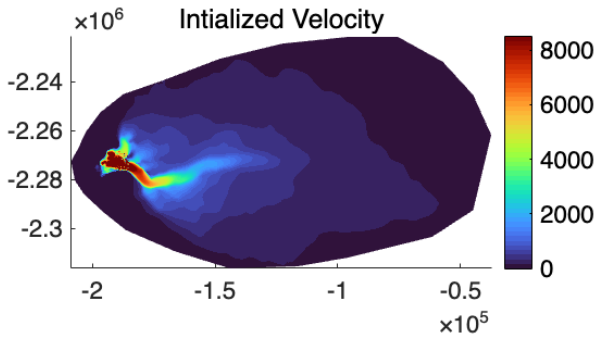
Iter	Cost function	Grad. norm	List of contributions		
1	f(x)= 424.78	0.881	143.3	281.4	2.671e-33
2	f(x)= 122.13	0.149	28.52	93.6	0.01364
3	f(x)= 94.315	0.102	23.6	70.7	0.01917
4	f(x)= 52.251	0.046	15.73	36.47	0.0428
5	f(x)= 37.172	0.0454	14.01	23.09	0.07496
6	f(x)= 28.671	0.0728	12.3	16.24	0.1216
7	f(x)= 23.635	0.0251	8.993	14.48	0.1594
8	f(x)= 21.911	0.0185	8.368	13.37	0.1734
9	f(x)= 19.034	0.0151	7.895	10.93	0.2079
10	f(x)= 17.526	0.0143	7.614	9.686	0.2263
11	f(x)= 16.149	0.0208	7.465	8.425	0.2585
12	f(x)= 15.041	0.0101	6.948	7.825	0.2686
13	f(x)= 14.498	0.00748	6.774	7.45	0.2736
14	f(x)= 13.732	0.0071	6.609	6.833	0.2901
15	f(x)= 13.29	0.0113	6.529	6.456	0.305
16	f(x)= 12.886	0.00713	6.406	6.162	0.3172
17	f(x)= 12.534	0.006	6.301	5.907	0.3263
18	f(x)= 12.251	0.00616	6.221	5.7	0.3307
19	f(x)= 11.861	0.00608	6.151	5.373	0.3375
20	f(x)= 11.592	0.00469	6.085	5.164	0.3439
21	f(x)= 11.429	0.00372	6.049	5.032	0.3483
22	f(x)= 11.223	0.00388	6	4.868	0.3542
23	f(x)= 11.043	0.00445	5.961	4.722	0.3596
24	f(x)= 10.889	0.00409	5.927	4.595	0.3667
25	f(x)= 10.811	0.00271	5.907	4.537	0.3677
26	f(x)= 10.695	0.00285	5.873	4.452	0.3699
27	f(x)= 10.587	0.0027	5.847	4.365	0.3742

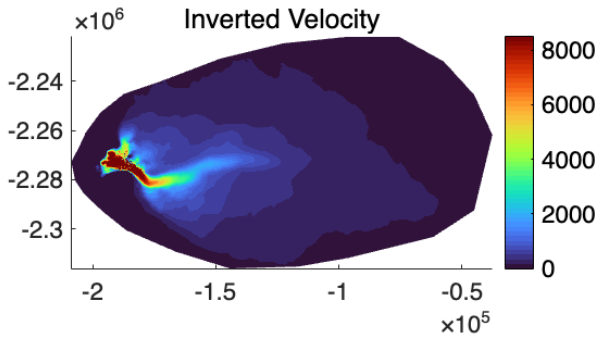
28	f(x)= 10.453	0.00389	5.825	4.25	0.378
29	f(x)= 10.375	0.00236	5.8	4.189	0.3863
30	f(x)= 10.353	0.00199	5.802	4.166	0.3857
31	f(x)= 10.272	0.00176	5.777	4.111	0.384
32	f(x)= 10.226	0.00307	5.769	4.073	0.3844
33	f(x)= 10.184	0.00172	5.75	4.047	0.3872
34	f(x)= 10.146	0.00142	5.737	4.019	0.3893
35	f(x)= 10.125	0.00154	5.734	4	0.3907
36	f(x)= 10.051	0.00156	5.714	3.941	0.3957
37	f(x)= 10.055	0.00402	5.712	3.945	0.3984
38	f(x)= 10.032	0.00203	5.703	3.932	0.3968
39	f(x)= 10.03	0.00154	5.698	3.944	0.3879
40	f(x)= 12.904	0.0282	6.902	5.622	0.3793
41	f(x)= 9.9843	0.00225	5.673	3.923	0.3889
42	f(x)= 9.973	0.00146	5.676	3.904	0.3925
43	f(x)= 9.9591	0.000965	5.678	3.891	0.39
44	f(x)= 9.9476	0.000952	5.678	3.883	0.3867
45	f(x)= 9.9137	0.00106	5.672	3.866	0.3759
46	f(x)= 9.896	0.00199	5.672	3.858	0.3658
47	f(x)= 9.8806	0.00112	5.665	3.851	0.3649
48	f(x)= 9.8753	0.00162	5.667	3.847	0.3618
49	f(x)= 9.8777	0.00149	5.67	3.846	0.3617
50	f(x)= 9.8814	0.00192	5.674	3.846	0.3618
51	f(x)= 9.884	0.00195	5.676	3.846	0.3618
52	f(x)= 9.8846	0.00197	5.676	3.847	0.3618
53	f(x)= 9.8853	0.00201	5.677	3.847	0.3618
54	f(x)= 9.8861	0.00205	5.677	3.847	0.3618
55	f(x)= 9.8867	0.00208	5.678	3.847	0.3618
56	f(x)= 9.8871	0.0021	5.678	3.847	0.3618
57	f(x)= 9.8874	0.00211	5.678	3.847	0.3618
58	f(x)= 9.8876	0.00212	5.678	3.847	0.3618
59	f(x)= 9.8878	0.00213	5.679	3.847	0.3618

```
   stopped on dxmin during line search
   preparing final solution
write lock file:

   FemModel initialization elapsed time:   0.06634
   Total Core solution elapsed time:       43.5056
   Linear solver elapsed time:             30.2389 (70%)

   Total elapsed time: 0 hrs 0 min 43 sec
```

```
%Put results back into the model
md.friction.coefficient=md.results.StressbalanceSolution.FrictionCoefficient
;
md.initialization.vx=md.results.StressbalanceSolution.Vx;
md.initialization.vy=md.results.StressbalanceSolution.Vy;
md.initialization.vel=md.results.StressbalanceSolution.Vel;
md.stressbalance.spcvx(pos)=nan;
md.stressbalance.spcvy(pos)=nan;
```

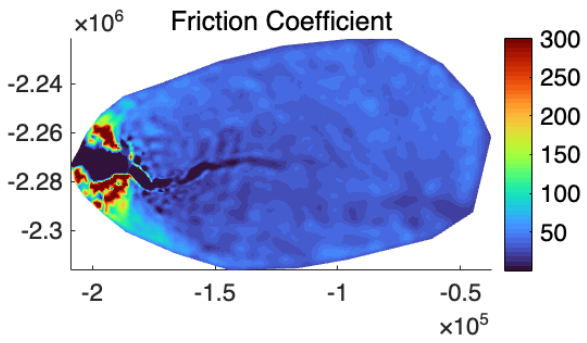
```
% Plot the surface velocity and basal friction
plotmodel(md,'data', md.initialization.vel,'title','Intialized Velocity',
...
'data',md.inversion.vel_obs,'title','Inverted Velocity','caxis#all',[0
8500])
```

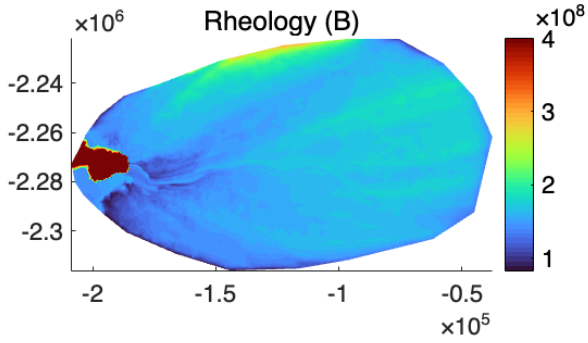




Plot other variables

```
plotmodel(md,'data',
md.results.StressbalanceSolution.FrictionCoefficient,'title','Friction
Coefficient',...
'data',md.materials.rheology_B,'title','Rheology (B)')
```


Friction Coefficient
×10^6
-2.24
-2.26
-2.28
-2.3
-2
-1.5
-1
-0.5
×10^5
300
250
200
150
100
50


Rheology (B)
×10^6
-2.24
-2.26
-2.28
-2.3
-2
-1.5
-1
-0.5
×10^5
×10^8
4
3
2
1