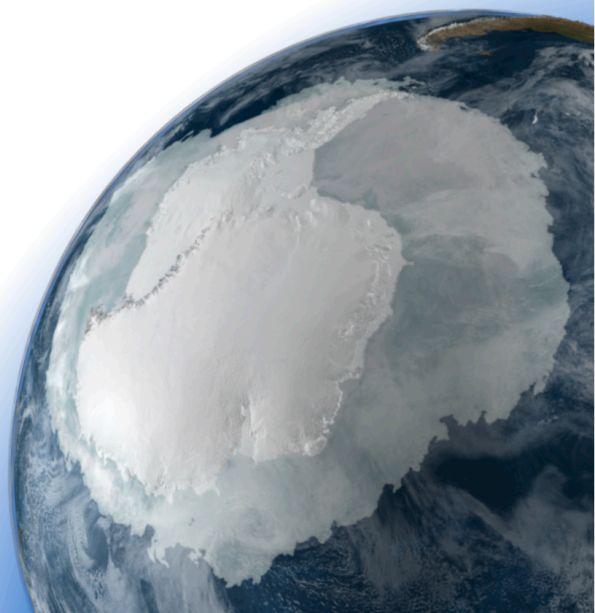
ISSM Workshop 2016

Ice Sheet System Model

Uncertainty quantification of PIG

Nicole SCHLEGEL[1],

[1] University of California, Los Angeles, CA



©Copyright 2016. All rights reserved

Overview

1. Introduction
2. Sampling Analysis
3. Sensitivity Analysis
4. Plot results
5. Conclusion
6. Additional Exercises

Introduction

Follow on study of Pine Island Glacier, to assess how errors in model inputs propagate through a 2D SSA steady-state ice flow model.

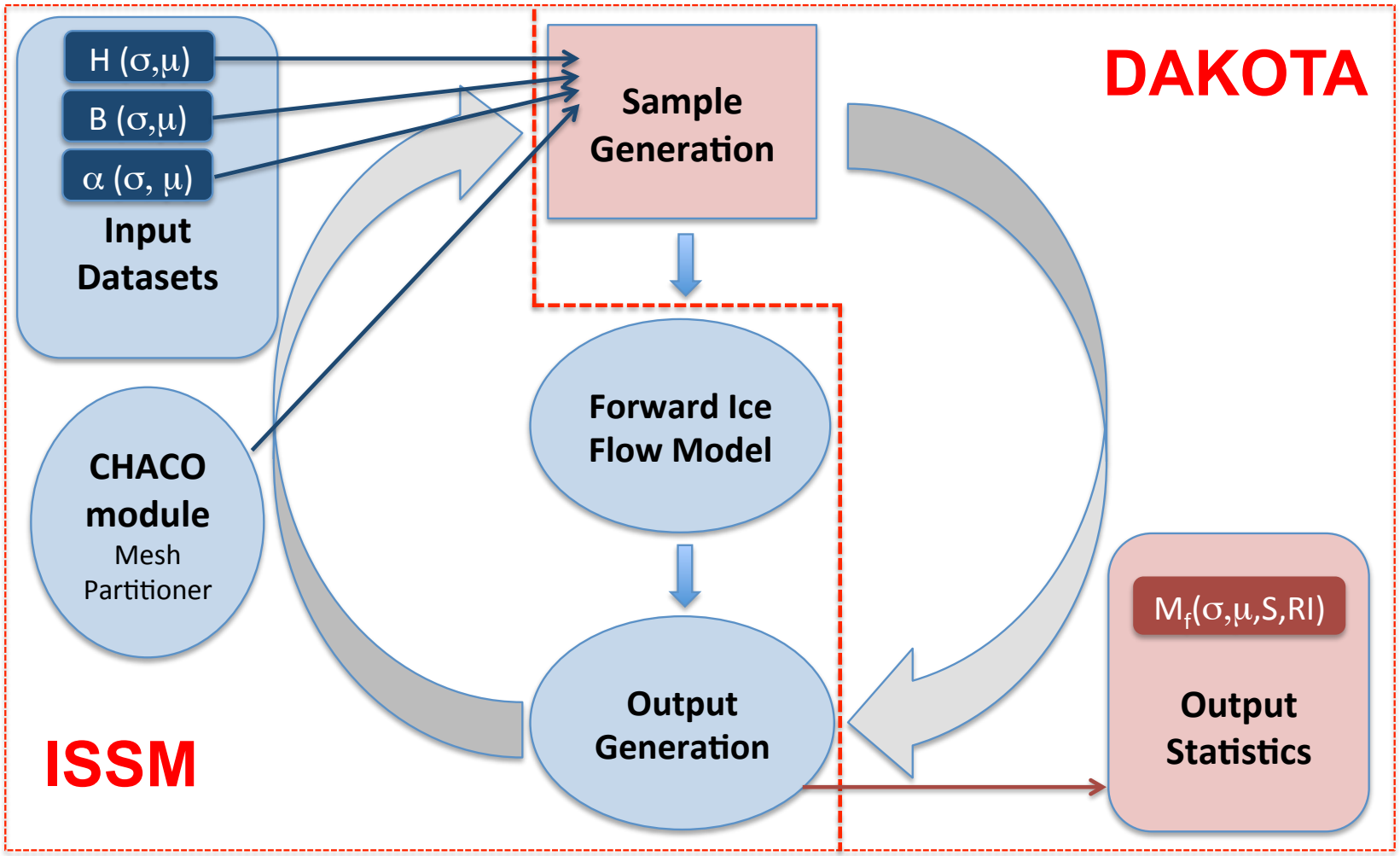
Our model inputs: ice thickness, ice rigidity and basal friction.
Our model outputs: mass flux at 13 flux gates across PIG.

Steps:

- Load PIG Application results (starting from the end of the basal friction inversion)
- Load ice thickness cross-over errors from IceBridge 2009 WAIS campaign
- Run sampling analysis using ice thickness cross-over and mass flux diagnostics.
- Run sensitivity analysis using ice thickness, ice rigidity and basal friction as inputs and mass flux diagnostics
- Plot results: partition
- Plot results: sampling
- Plot results: sensitivities

See [Larour et al, 2012 b and c] for more details on ISSM QMU capabilities

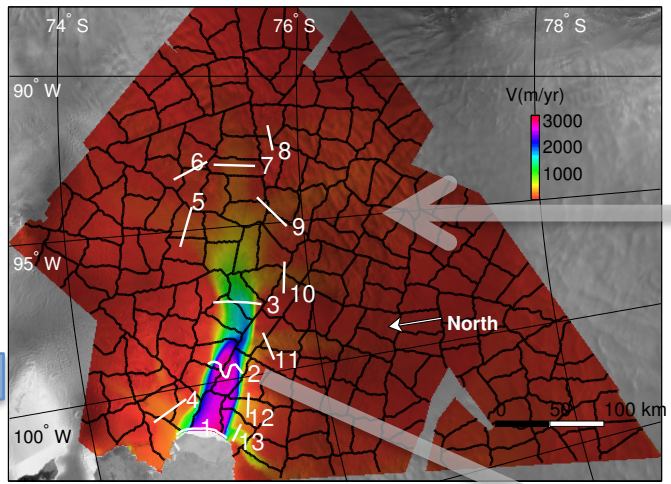
ISSM-DAKOTA

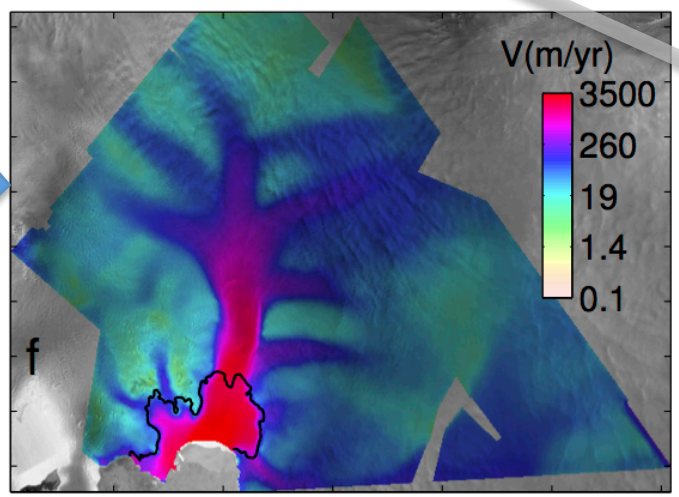

DAKOTA
H (σ,μ)
B (σ,μ)
α (σ, μ)
Input Datasets
Sample Generation
Forward Ice Flow Model
CHACO module
Mesh Partitioner
Output Generation
$M_f(\sigma,\mu,S,RI)$
Output Statistics
ISSM

Sampling

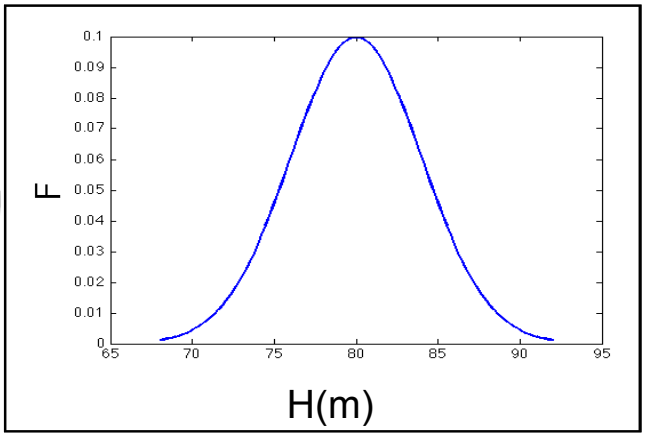
Repeated Samples

2D SSA

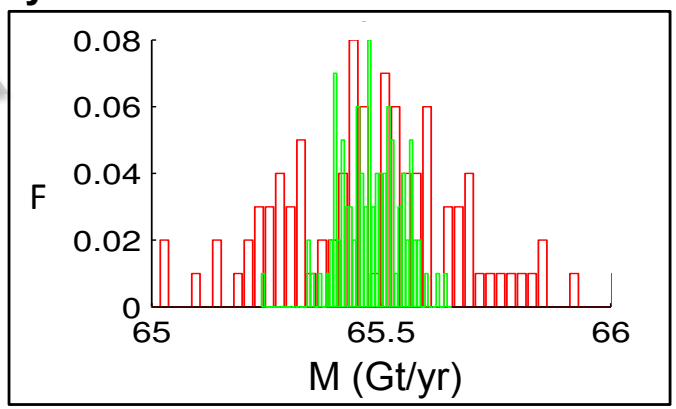




Thickness Errors

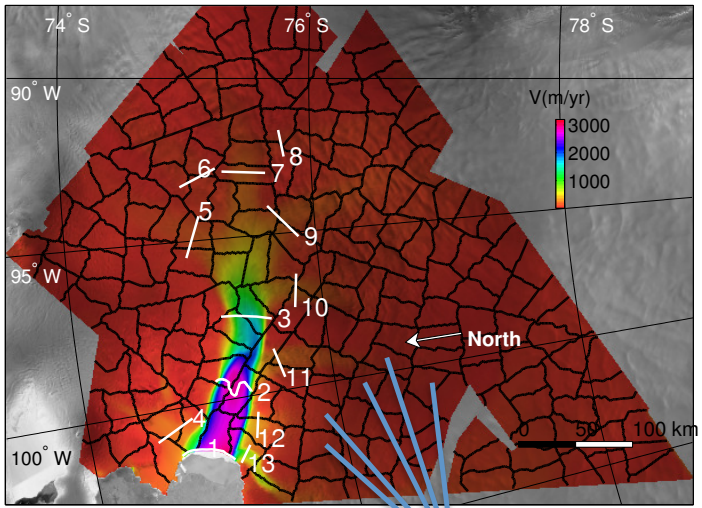


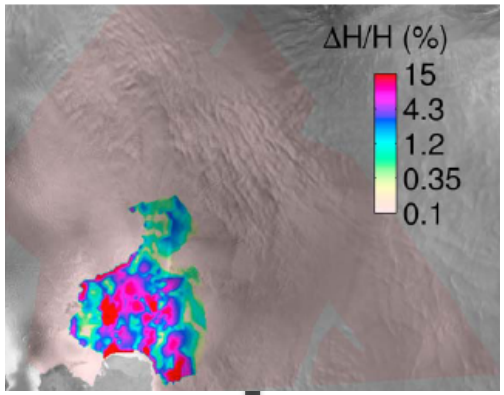
Mass Flux Uncertainty



Sensitivity

Small perturbation in thickness, one partition at a time





Thickness Errors

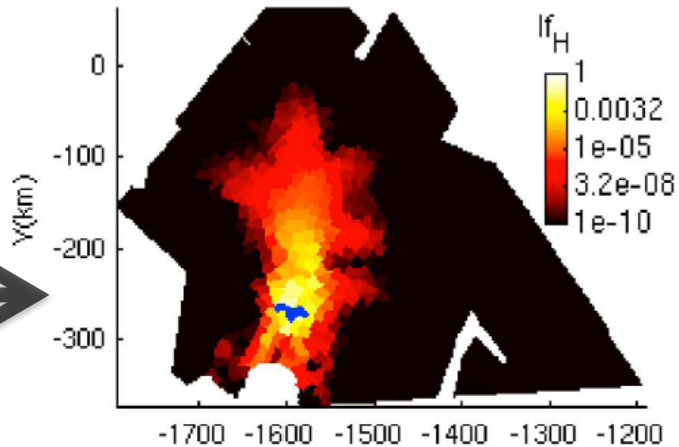
One 2D- SSA solve per partition

r = grounding line mass flux
x_i = ice thickness

$$\theta_i = \frac{\delta r}{\delta x_i}$$

Spatial sensitivities of response *r* to change in variable *x*

Sensitivities



Importance Factors

First Step: flux gates

Flux gates are exp files found in MassFluxes. The gates are positioned across PIG at the inset of tributary glaciers.

Mass fluxes will be computed (in Gt/yr) for all of these gates (using the depth-averaged ice velocity, ice thickness and ice density).

Run step 1 of the runme.m file to plot the gates overlayed over PIG surface velocities.

```
17  plotmodel(md,'data',md.results.StressbalanceSolution.Vel,'log',10,'expdisp',...
18   {'MassFluxes/MassFlux1.exp','MassFluxes/MassFlux2.exp',...
19   'MassFluxes/MassFlux3.exp','MassFluxes/MassFlux4.exp',...
20   'MassFluxes/MassFlux5.exp','MassFluxes/MassFlux6.exp',...
21   'MassFluxes/MassFlux7.exp','MassFluxes/MassFlux8.exp',...
22   'MassFluxes/MassFlux9.exp','MassFluxes/MassFlux10.exp',...
23   'MassFluxes/MassFlux11.exp','MassFluxes/MassFlux12.exp',...
24   'MassFluxes/MassFlux13.exp'},...
25   'expstyle',{'k-','k-','k-','k-','k-','k-','k-',...
26   'k-','k-','k-','k-','k-','k-'},'linewidth',2,...
27   'text',{'1','2','3','4','5','6','7',...
28   '8','9','10','11','12','13'},...
29   'textposition',textpositions);
```

Second step: load Cross-Over errors for ice thickness

For ice thickness errors, we will use McCords cross-over errors from CReSIS (courtesy of John Paden, pers. comm.). First load the errors (step2):

```
37  %load cross overs: CRESIS McCord Antarctica, 2009 (courtesy of John Paden)
38  load('../Data/CrossOvers2009.mat');
39
40  %interpolate cross over errors over our mesh vertices
41  DeltaHH=InterpFromMeshToMesh2d(index,x,y,dhh,md.mesh.x,md.mesh.y);
```

Some of these errors are too large, too small, or need to be interpolated onto a larger domain. Filter these out:

```
46  %filter out unrealistic error ranges
47  flags=ContourToNodes(md.mesh.x,md.mesh.y,'ErrorContour.exp',1);
48  pos=find(¬flags); DeltaHH(pos)=0;
49
50  %avoid large unrealistic values
51  pos=find(DeltaHH>1); DeltaHH(pos)=1;
52  pos=find(DeltaHH<-1); DeltaHH(pos)=-1;
53
54  %transform into absolute errors and setup a minimum error everywhere.
55  DeltaHH=abs(DeltaHH);
56  pos=find(DeltaHH==0); pos2=find(DeltaHH≠0);
57  DeltaHH(pos)=min(DeltaHH(pos2));
```

Third step: setup the sampling analysis

First, partition the mesh into equal area partitions. We'll start with 50.

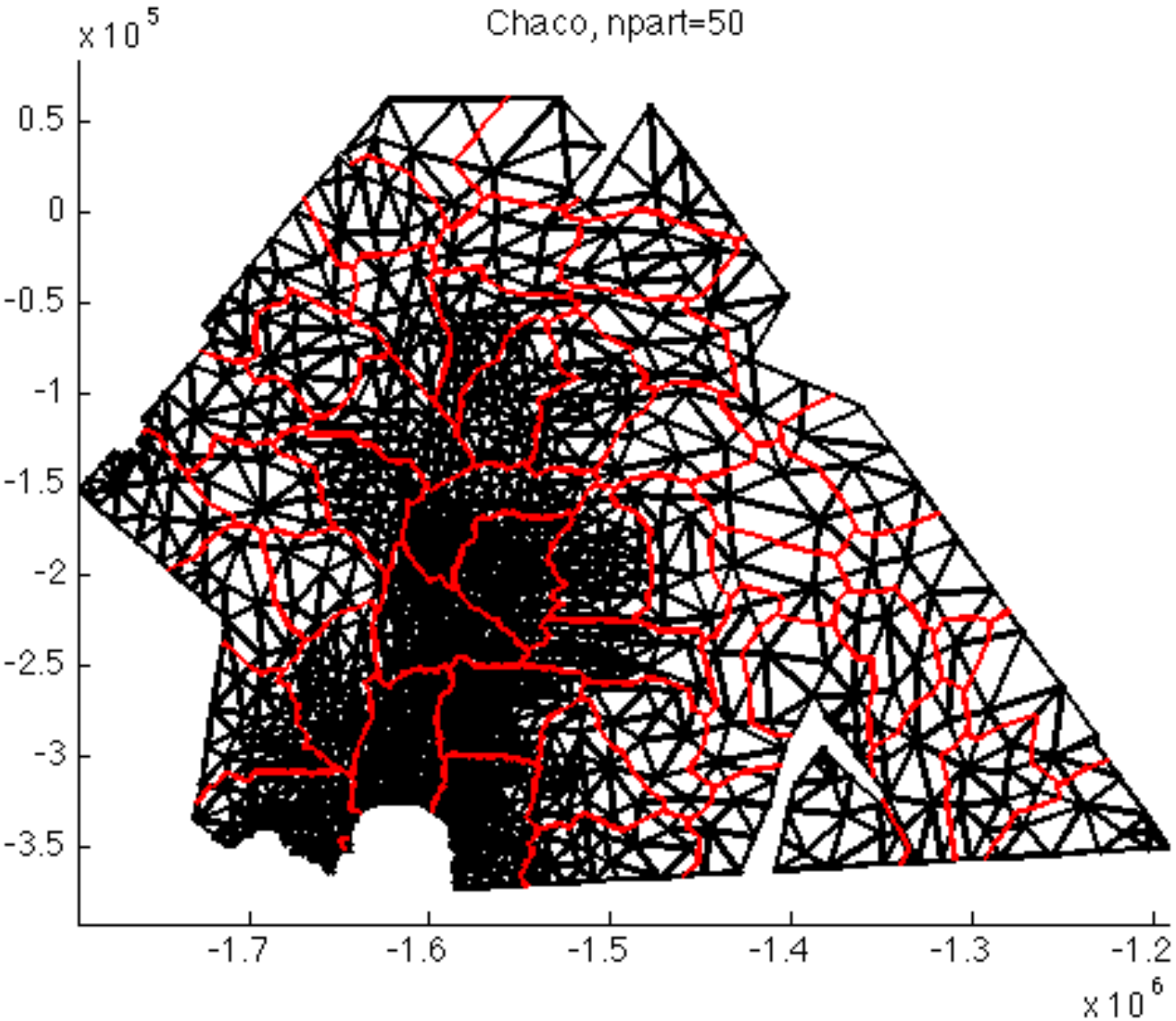
```
69   %partition the mesh
70   md.qmu.numberofpartitions=50;
71   md=partitioner(md,'package','chaco','npart',md.qmu.numberofpartitions,...
72   'weighting','on');
73   md.qmu.partition=md.qmu.partition-1; %switch partition to c-indexing
```

You can try and play with the package for partitioning ('chaco' or 'linear'), the number of partitions, and the weighting ('on' or 'off')

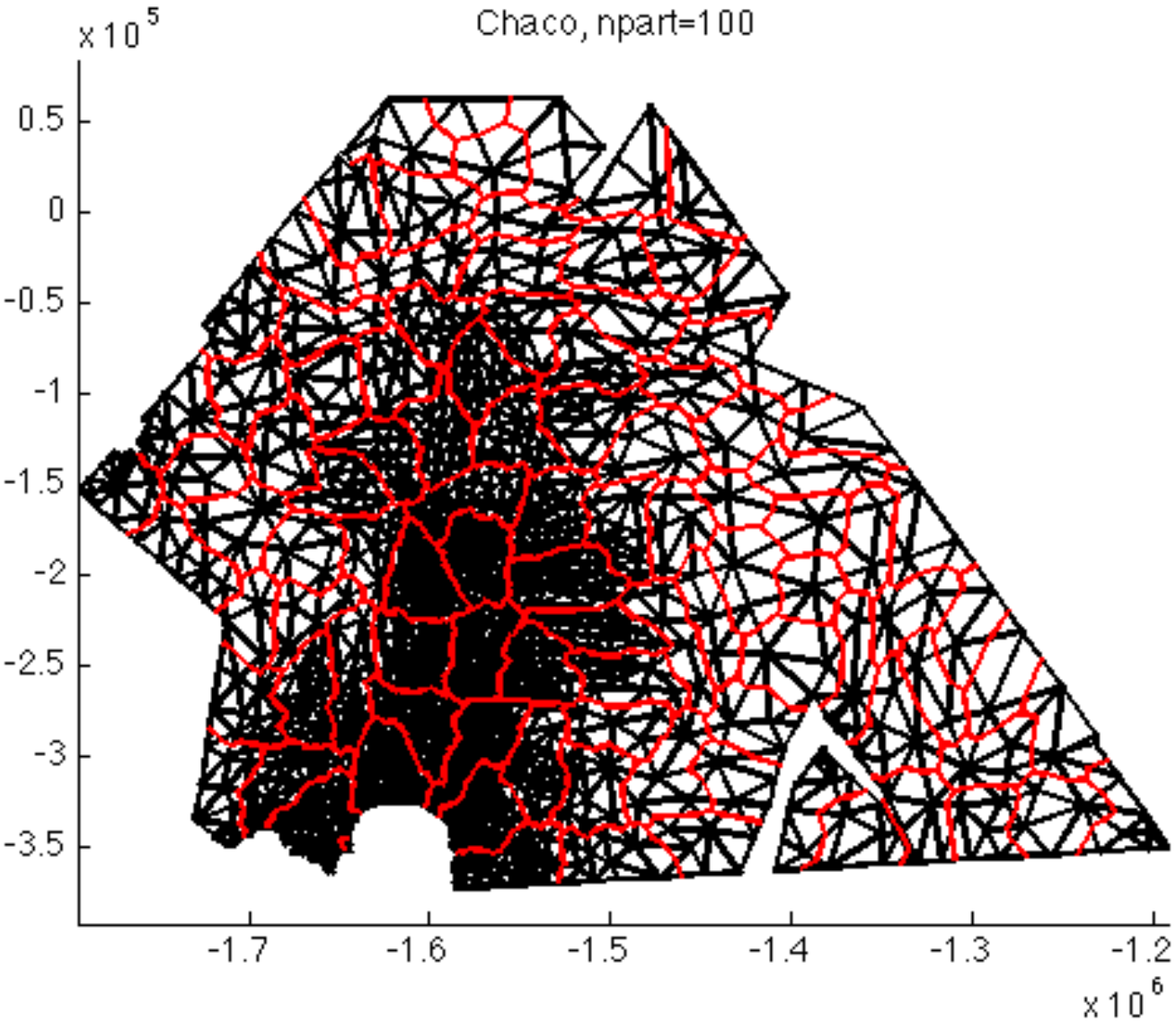
To plot the corresponding partition over a plot of the mesh:

```
249  plotmodel(md,'data','mesh','partitionedges','on',...
250  'linewidth',2, 'axis#all','image','unit','km','colorbar','off',...
251  'title','Partition Edges on ISSM mesh','grid','on');
```

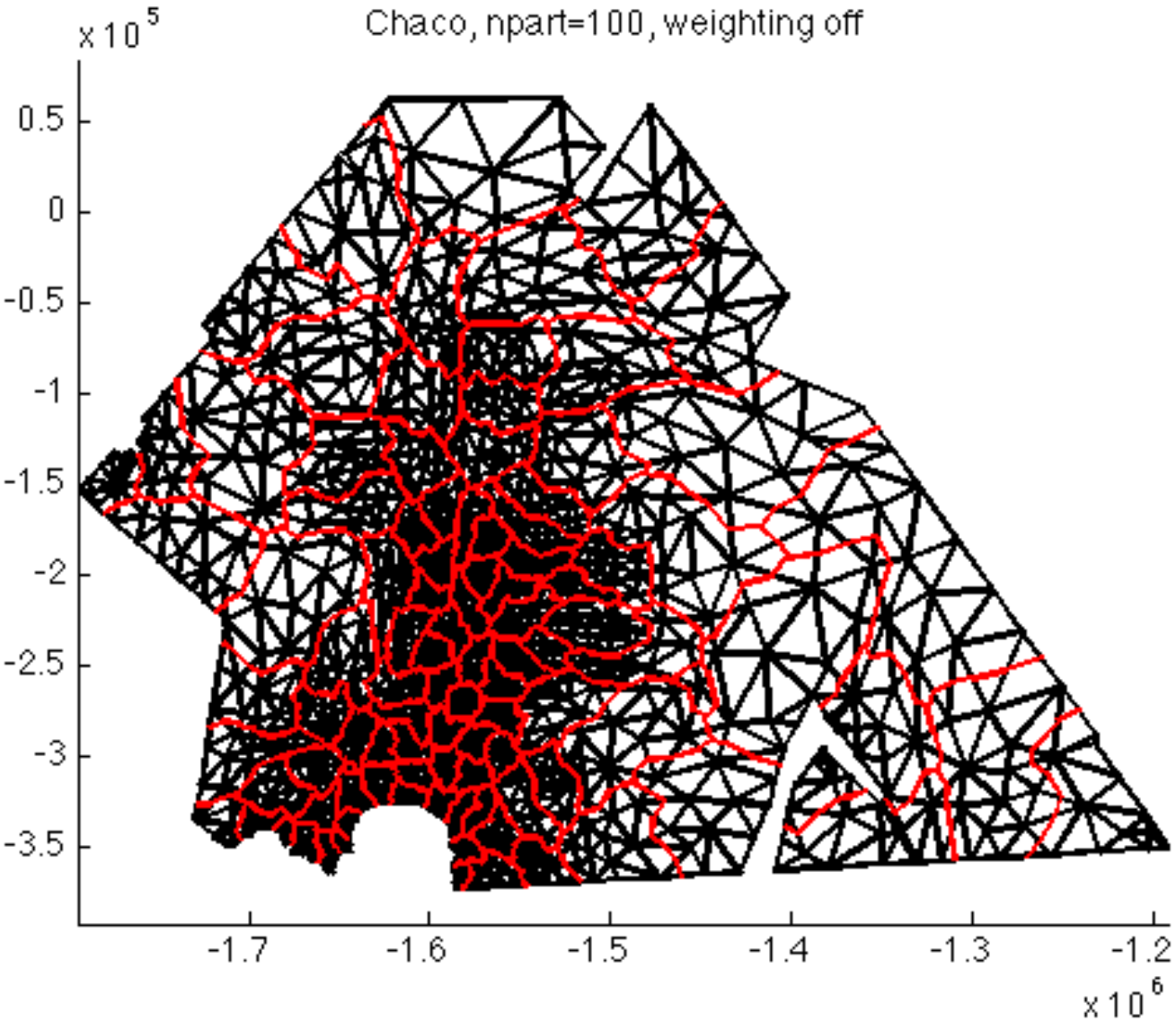
Examples of partitions-1:



Examples of partitions-2:



Examples of partitions-3:



Setup PDF Ice Thickness:

Second, setup PDF (Probability Density Function) for model input H:

```
75    %make DeltaHH into our 3 sigma deviation
76    DeltaHH=DeltaHH/6; %2 (to transform DeltaHH into a radius) x 3 (for 3 sigma)
77    DeltaHH_on_partition=AreaAverageOntoPartition(md,DeltaHH);
78    DeltaHH_on_grids=DeltaHH_on_partition(md.qmu.partition+1); %just to check ...
          in case
79
80    md.qmu.variables.thickness=normal_uncertain('scaled_Thickness',1,1);
81    md.qmu.variables.thickness.stddev=DeltaHH_on_partition;
```

Here, we need to provide the average μ and 3σ standard deviation (from $\Delta H/H$) for the PDF to be defined. Then we interpolate on the partition (not the mesh!).

Anything UQ related will be in the md.qmu structure (qmu for Quantification of Margins and Uncertainties).

qmu class:

Anything UQ related will be in the md.qmu structure (QMU stands for Quantification of Margins and Uncertainties).

```
>> md.qmu

ans =

   qmu parameters:
         isdakota                : 0                -- is qmu analysis activated?
         variables:  (arrays of each variable class)
         responses:  (arrays of each response class)
         numberofresponses       : 0                -- number of responses
         params:  (array of method-independent parameters)
         results:  (information from dakota files)
         partition               : N/A              -- user provided mesh partitioning, defaults to metis if not specifi
ed
         numberofpartitions      : 0                -- number of partitions for semi-discrete qmu
         variabledescriptors     : (0x0)
         responsedescriptors     : (0x0)
         method                  : N/A              -- array of dakota_method class
         mass_flux_profile_di...: N/A               -- directory for mass flux profiles
         mass_flux_profiles      : N/A              -- list of mass_flux profiles
         mass_flux_segments      : (0x0)
         adjacency               : N/A
         vertex_weight           : N/A              -- weight applied to each mesh vertex
>>
```

Setup output diagnostics-1:

Third, setup output responses (mass flux at 13 flux gates):

```
83   %responses
84   md.qmu.responses.MassFlux1=response_function('indexed_MassFlux_1',[],...
85    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
86   md.qmu.responses.MassFlux2=response_function('indexed_MassFlux_2',[],...
87    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]); %grounding line
88   md.qmu.responses.MassFlux3=response_function('indexed_MassFlux_3',[],...
89    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
90   md.qmu.responses.MassFlux4=response_function('indexed_MassFlux_4',[],...
91    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
92   md.qmu.responses.MassFlux5=response_function('indexed_MassFlux_5',[],...
93    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
94   md.qmu.responses.MassFlux6=response_function('indexed_MassFlux_6',[],...
95    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
96   md.qmu.responses.MassFlux7=response_function('indexed_MassFlux_7',[],...
97    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
98   md.qmu.responses.MassFlux8=response_function('indexed_MassFlux_8',[],...
99    [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
100  md.qmu.responses.MassFlux9=response_function('indexed_MassFlux_9',[],...
101   [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
102  md.qmu.responses.MassFlux10=response_function('indexed_MassFlux_10',[],...
103   [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
104  md.qmu.responses.MassFlux11=response_function('indexed_MassFlux_11',[],...
105   [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
106  md.qmu.responses.MassFlux12=response_function('indexed_MassFlux_12',[],...
107   [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
108  md.qmu.responses.MassFlux13=response_function('indexed_MassFlux_13',[],...
109   [0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
```

Setup output diagnostics-2:

For all responses, we specify a string identificator *'indexed_MassFlux_1'*, and confidence intervals. We also need to specify an exp file to describe each flux gate, and a directory where to find the latter.

```
111    %mass flux profiles
112    md.qmu.mass_flux_profiles={'MassFlux1.exp',...
113                               'MassFlux2.exp',...
114                               'MassFlux3.exp',...
115                               'MassFlux4.exp',...
116                               'MassFlux5.exp',...
117                               'MassFlux6.exp',...
118                               'MassFlux7.exp',...
119                               'MassFlux8.exp',...
120                               'MassFlux9.exp',...
121                               'MassFlux10.exp',...
122                               'MassFlux11.exp',...
123                               'MassFlux12.exp',...
124                               'MassFlux13.exp'};
125    md.qmu.mass_flux_profile_directory='../MassFluxes/';
```

Setup sampling engine:

Fourth, we need to decide on a sampling strategy:

```
127   %%  sampling analysis
128   md.qmu.method      =dakota_method('nond_samp');
129   md.qmu.method(end)=dmeth_params_set(md.qmu.method(end),...
130   'seed',1234,...
131   'samples',30,...
132   'sample_type','lhs'); %random or lhs
```

We specify the type of method ('*nond_samp*' for sampling or '*nond_l*' for local reliability method/sensitivity analysis), following Dakota guidelines.

We determine the number of samples (30 for now) and also the type of sampling algorithm ('lhs' or 'random').

Some DAKOTA parameters:

We setup persistent parameters:

```
134    %%  a variety of parameters
135    md.qmu.params.direct=true;
136    md.qmu.params.analysis_driver='';
137    md.qmu.params.analysis_components='';
138    md.qmu.params.evaluation_concurrency=1;
139    md.qmu.params.tabular_graphics_data=true;
140
141    md.stressbalance.restol=10^-5; %tighten tolerances for UQ analyses
```

This includes parallel concurrency, verbosity, and data backup ('tabular_graphics_data'); We also have to tighten the solver tolerance (in order to avoid spurious sensitivities to develop).

Setup Master/Slave CPU preferences:

We set up dakota to run in parallel:

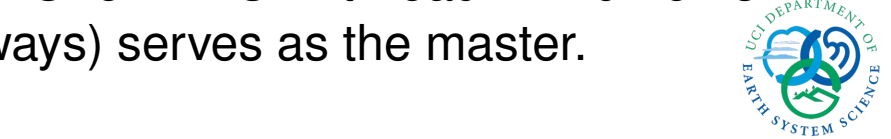
```
146    %Here, we choose to run with 4 processors, 3 for DAKOTA
147    % while one serves as the master
148    md.cluster=generic('name',oshostname,'np',4);
149
150    %Dakota runs in parallel with a master/slave configuration.
151    % At least 2 cpu's are needed to run the UQ
152    md.qmu.params.evaluation_scheduling='master';
153    md.qmu.params.processors_per_evaluation=md.cluster.np-1;
```

Note, the ISSM-Dakota implementation now requires that DAKOTA runs in parallel with a master/slave configuration. This means that at least 2 cpu's are needed to run the UQ. This means that:

$$md.cluster.np = md.qmu.params.processors_per_evaluation * N$$

where N is an integer which represents the number of parallel DAKOTA threads that will run at once.

In this example, we run with 4 processors. One DAKOTA thread will run on 3 processors (slave), while 1 processor (always) serves as the master.



Solve:

We solve:

```
155    %Turn dakota on
156    md.qmu.isdakota=1; md.inversion.iscontrol=0;
157    md=solve(md,StressbalanceSolutionEnum,'overwrite','y');
```

Don't forget to deactivate inversion (iscontrol=0), and to activate UQ run (isdakota=1).

Results will be in md.results.dakota and md.qmu.results

Model Setup-1

Next, we quantify *importance factors* (sensitivities scaled by error margins) for model inputs: ice thickness H, basal friction α, and ice rigidity B. We specify a 5% error margin on all inputs.

Partitioning is identical. Model outputs are identical.

To add model inputs:

```
174   %variables
175   md.qmu.variables.DragCoefficient=normal_uncertain(...
176   'scaled_FrictionCoefficient',1,0.05);
177   md.qmu.variables.rheology_B=normal_uncertain(...
178   'scaled_MaterialsRheologyB',1,0.05);
179   md.qmu.variables.Thickness=normal_uncertain('scaled_Thickness',1,0.05);
```

To specify new sensitivity method: 'nond_l':

```
212   %method: local reliability
213   md.qmu.method      =dakota_method('nond_l');
214   md.qmu.method(end)=dmeth_params_set(md.qmu.method(end),...
215   'output','quiet');
216
217   %parameters
218   md.qmu.params.direct=true;
219   md.qmu.params.analysis_driver='';
220   md.qmu.params.analysis_components='';
221   md.qmu.params.evaluation_concurrency=1;
222   md.qmu.params.tabular_graphics_data=false;
```

Model Setup-2

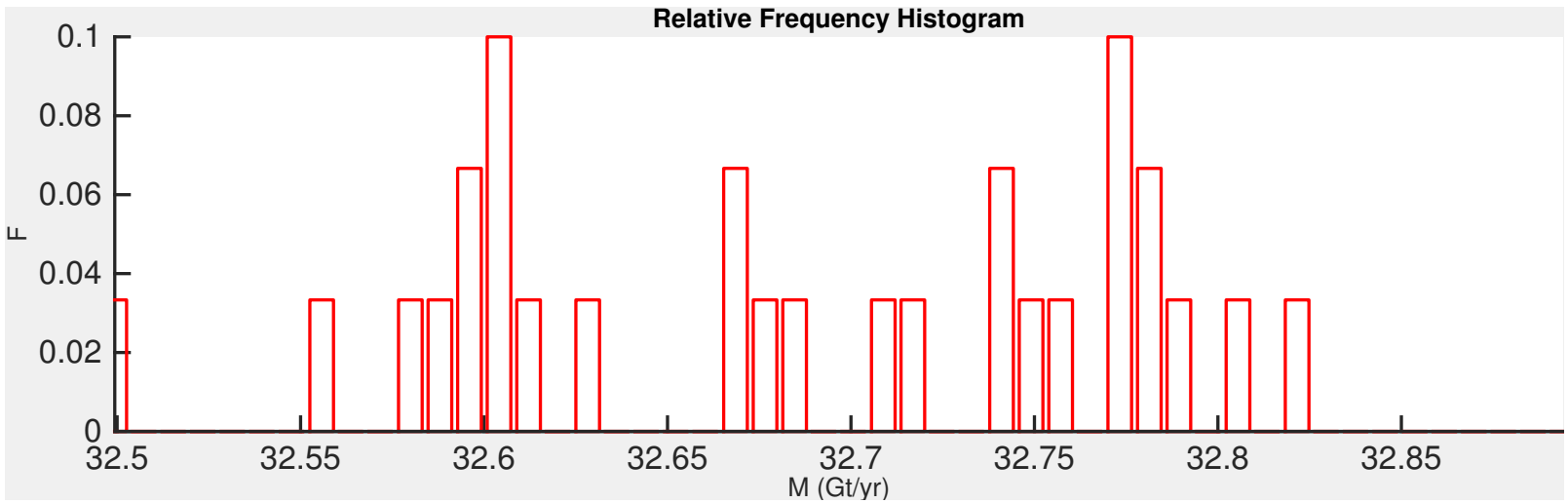
We solve the same way:

```
235    %Clear results and turn dakota on
236    md.results=[];
237    md.qmu.isdakota=1; md.inversion.iscontrol=0;
238    md.verbose=verbose('qmu',true);
239    md=solve(md,StressbalanceSolutionEnum,'overwrite','y');
```

Plot sampling results

In order to plot the results, we pick-up one of the flux gates, and display a histogram of the sampling results.

```
260    %which profile are we looking at?
261    index=1;
262
263    %retrieve results for the specific profile, mass flux in m^3 water equiv/s
264    result=md.results.dakota.dresp_dat(md.qmu.numberofpartitions+index);
265    result.sample=result.sample/1e12*60*60*24*365;
266
267    %plot histogram
268    plot_hist_norm(result,'cdfleg','off','cdfplt','off','nrmplt','off',...
269    'xlabelplt','M (Gt/yr)','ylabelplt','F','FontSize',8,'FaceColor',...
270    'none','EdgeColor','red');
```



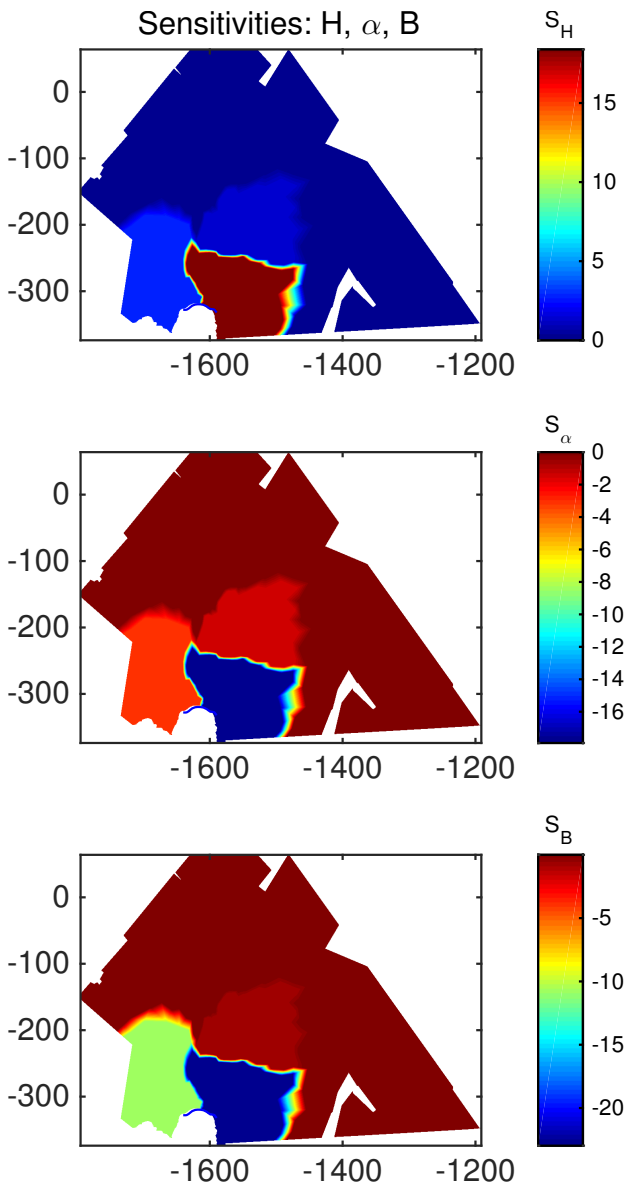
Plot sensitivity results

To retrieve sensitivities for each model input:

```
281    %which profile are we looking at?
282    index=1;
283
284    %To plot sensitivities
285    sa=md.results.dakota.dresp_out(index).sens(1:10); ...
           sa=sa(md.qmu.partition+1)/1e12*60*60*24*365;
286    sb=md.results.dakota.dresp_out(index).sens(11:20); ...
           sb=sb(md.qmu.partition+1)/1e12*60*60*24*365;
287    sh=md.results.dakota.dresp_out(index).sens(21:30); ...
           sh=sh(md.qmu.partition+1)/1e12*60*60*24*365;
```

To plot, as for any other field:

```
289    plotmodel(md,'data',sh,'data',sa,'data',sb,'expdisp#all',...
290     ['MassFluxes/MassFlux' num2str(index) '.exp'],...
291     'expstyle#all','b-','linewidth#all',2,...
292     'nlines',3,'ncols',1, 'axis#all','image',...
293     'colorbar#all','on','colorbarfontsize#all',10,...
294     'colorbartitle#1','S_{H}', 'colorbartitle#2','S_{\alpha}',...
295     'colorbartitle#3','S_{B}','unit#all','km','figure',1,...
296     'title','Sensitivities: H, \alpha, B');
```


Sensitivities: H, α, B
S_H
S_α
S_B
0
-100
-200
-300
-1600
-1400
-1200
15
10
5
0
-2
-4
-6
-8
-10
-12
-14
-16
-5
-10
-15
-20

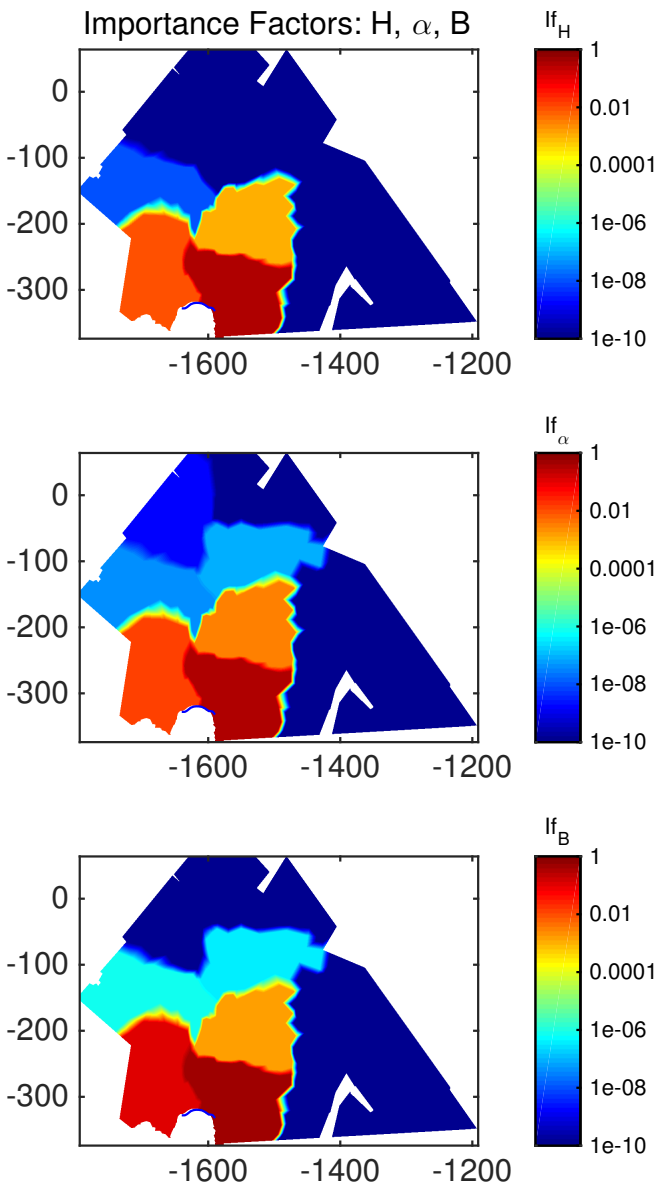
Plot Importance Factors

To retrieve importance factors for each model input:

```
298    %To plot importance factors
299    ifa=importancefactors(md,'scaled_FrictionCoefficient',['indexed_MassFlux_' ...
             num2str(index)]);
300    ifb=importancefactors(md,'scaled_MaterialsRheologyB',['indexed_MassFlux_' ...
             num2str(index)]);
301    ifh=importancefactors(md,'scaled_Thickness',['indexed_MassFlux_' ...
             num2str(index)]);
```

To plot importance factors:

```
303    plotmodel(md,'data',ifh,'data',ifa,'data',ifb,'expdisp#all',...
304     ['MassFluxes/MassFlux' num2str(index) '.exp'],...
305     'expstyle#all','b-','linewidth#all',2,'log#all',10,...
306     'nlines',3,'ncols',1, 'axis#all','image','caxis#all',[1e-10 1],...
307     'colorbar#all','on','colorbarfontsize#all',10,...
308     'colorbartitle#1','If_{H}', 'colorbartitle#2','If_{\alpha}',...
309     'colorbartitle#3','If_{B}','unit#all','km','figure',2,...
310     'title','Importance Factors: H, \alpha, B');
```


Importance Factors: H, α, B
If_H
If_α
If_B
1
0.01
0.0001
1e-06
1e-08
1e-10
0
-100
-200
-300
-1600
-1400
-1200

Conclusion

UQ allows the following:

- assess propagation of errors from model inputs to model diagnostics
- assess sensitivites of model diagnostics with respect to model inputs
- quantify error margins for projections

Our UQ capabilities apply to any model input, any model output, any solution, irrespective of the type of sampling or sensitivity analysis being carried out.

QMU

Exercises

Extra Time? Try to something more complicated.

Addditional Suggested Exercises:

- Add diagnostics IceVolume or MaxVelocity
- Sample with a uniform distribution (See help uniform_uncertain)
- Sample additional variables (i.e. friction coefficient, ice rheology)
- Try qmu on a different solution type
- Change number of partitions, NB: for sensitivity this could take a while!

Thanks!