

Ice Sheet System model

Application to Pine Island Glacier: Uncertainty Quantification (ice thickness, ice rigidity and basal drag)

Chris BORSTAD¹, Bao DUONG^{5,1}, Feras HABBAL^{2,1}, Daria HALKIDES^{1,3},
Michiel HELSEN², **Eric LAROUR**¹, Mathieu MORLIGHEM², Lan NGUYEN^{5,1},
Gilberto PÉREZ^{4,1}, Eric RIGNOT^{2,1}, John SCHIERMEIER¹,
Nicole SCHLEGEL¹, Hélène SEROUSSI¹

¹Jet Propulsion Laboratory - California Institute of Technology

²University of California, Irvine

³Joint Institute for Regional Earth System Science & Engineering, UCLA

⁴University of Southern California

⁵Cal Poly Pomona

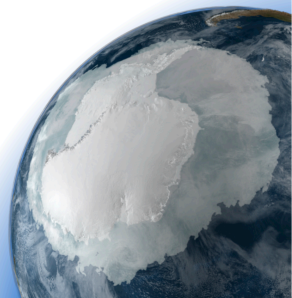


FIG: UQ Application

Overview

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

- 1 Introduction
- 2 Model setup
- 3 Sensitivity Analysis
- 4 Plot results
- 5 Conclusion

Introduction

Follow on study on 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 09 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

For more details on Uncertainty Quantification, check on [Larour et al, 2012]

FIG: UQ Application

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

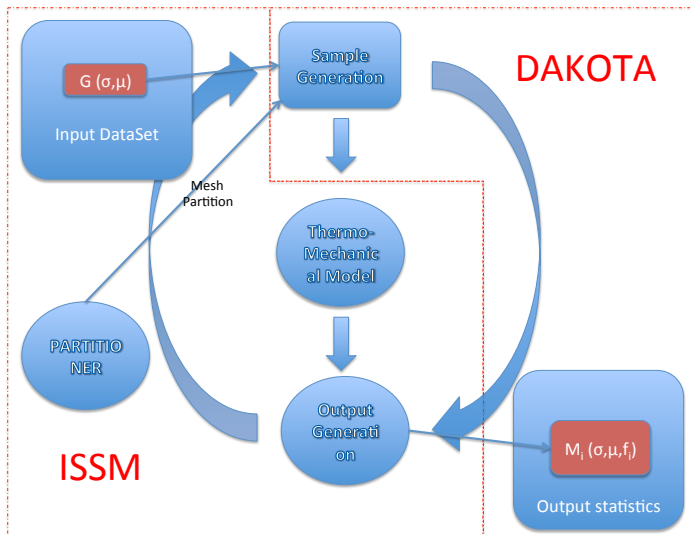


FIG: UQ Application

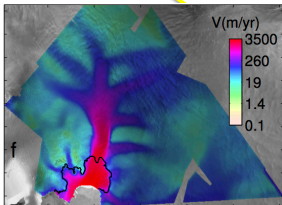
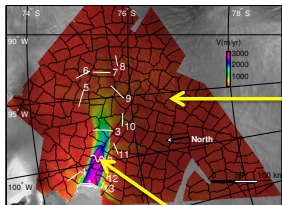
Introduction

Model setup

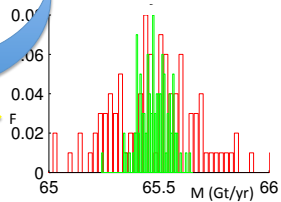
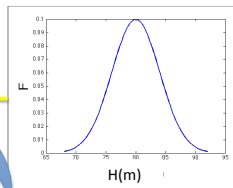
Sensitivity Analysis

Plot results

Conclusion



2D SSA



First Step: flux gates

Flux gates are exp files found in **Exp_Par/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.DiagnosticSolution.Vel, 'log', 10, ...
18 'expdisp', {'Exp_Par/MassFluxes/MassFlux1.exp', ...
19 'Exp_Par/MassFluxes/MassFlux2.exp', ...
20 'Exp_Par/MassFluxes/MassFlux3.exp', ...
21 'Exp_Par/MassFluxes/MassFlux4.exp', ...
22 'Exp_Par/MassFluxes/MassFlux5.exp', ...
23 'Exp_Par/MassFluxes/MassFlux6.exp', ...
24 'Exp_Par/MassFluxes/MassFlux7.exp', ...
25 'Exp_Par/MassFluxes/MassFlux8.exp', ...
26 'Exp_Par/MassFluxes/MassFlux9.exp', ...
27 'Exp_Par/MassFluxes/MassFlux10.exp', ...
28 'Exp_Par/MassFluxes/MassFlux11.exp', ...
29 'Exp_Par/MassFluxes/MassFlux12.exp', ...
30 'Exp_Par/MassFluxes/MassFlux13.exp'}, ...
31 'expstyle', {'k-', 'k-', 'k-', 'k-', 'k-', 'k-', 'k-', 'k-', 'k-', 'k-', 'k-', 'k-'},
32 'text', {'1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13'}, 'textposition')
```

FIG: UQ Application

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

Second step: load Cross-Over errors for ice thickness

What are the error margins on our ice thickness? — > McCords cross-over errors from CReSIS (courtesy of John Paden, pers. comm.). First load up the errors (step2):

```
40  %load cross overs from CRESIS McCord Antarctica 2009 season (courtesy ...  
    of John Paden)  
41  load('./Data/CrossOvers2009.mat');  
42  
43  %interpolate cross over errors over our mesh vertices  
44  DeltaHH=InterpFromMeshToMesh2d(index,x,y,dhh,md.mesh.x,md.mesh.y);
```

Then filter out, as some of these errors are too large, too small, or need to be interpolated onto a larger domain:

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

Third step: setup the sampling analysis

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

```
75 md.qmu.numberofpartitions=50;  
76 md=partitioner(md, 'package', 'chaco', 'npart', md.qmu.numberofpartitions, ...  
77 'weighting', 'on');  
78 md.qmu.partition=md.qmu.partition-1; %switch partition to c-indexing
```

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

To plot the corresponding partition on top of the mesh:

```
221 md = loadmodel('./Models/Pig.Sampling');  
222  
223 plotmodel(md, 'data', 'mesh', 'partitionedges', 'on', 'meshlinewidth', 1.5, ...
```


FIG: UQ Application

Examples of partitions-1:

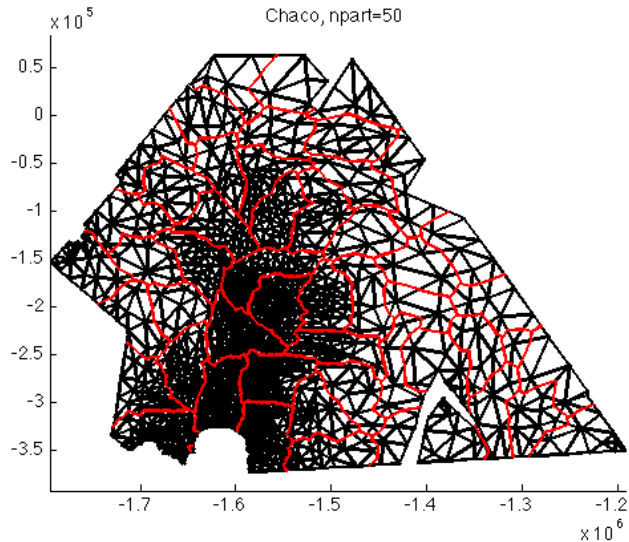
Introduction

Model setup

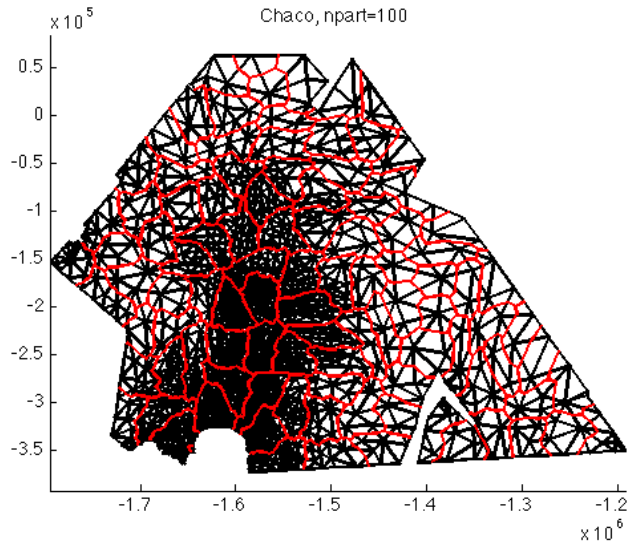
Sensitivity Analysis

Plot results

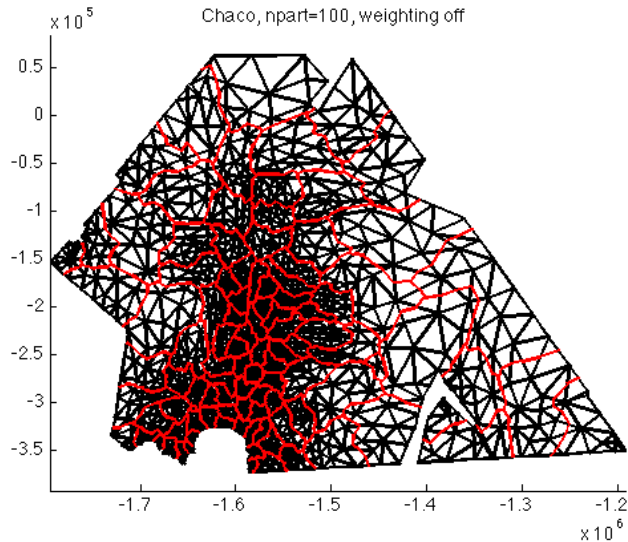
Conclusion



Examples of partitions-2:



Examples of partitions-3:



Setup PDF Ice Thickness:

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

```
81 DeltaHH=DeltaHH/6; %2 (to transform DeltaHH into a radius) x 3 (for 3 sigma)
82 DeltaHH_on_partition=AreaAverageOntoPartition(md,DeltaHH);
83 DeltaHH_on_grids=DeltaHH_on_partition(md.qmu.partition+1); %just to ...
    check in case
84
85 md.qmu.variables.thickness=normal_uncertain('scaled_Thickness',1,1);
86 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).

PIG: UQ Application

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

qmu class:

Anything UQ related will be in the **md.qmu** structure (qmu 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
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_directory : 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
>>

```

FIG: UQ Application

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

Setup output diagnostics-1:

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

```
89  %responses
90  md.qmu.responses.MassFlux1=response_function('indexed_MassFlux_1',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
91  md.qmu.responses.MassFlux2=response_function('indexed_MassFlux_2',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]); %grounding line
92  md.qmu.responses.MassFlux3=response_function('indexed_MassFlux_3',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
93  md.qmu.responses.MassFlux4=response_function('indexed_MassFlux_4',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
94  md.qmu.responses.MassFlux5=response_function('indexed_MassFlux_5',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
95  md.qmu.responses.MassFlux6=response_function('indexed_MassFlux_6',[],[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',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
97  md.qmu.responses.MassFlux8=response_function('indexed_MassFlux_8',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
98  md.qmu.responses.MassFlux9=response_function('indexed_MassFlux_9',[],[0.0001 .
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
99  md.qmu.responses.MassFlux10=response_function('indexed_MassFlux_10',[],[0.0001
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
100 md.qmu.responses.MassFlux11=response_function('indexed_MassFlux_11',[],[0.0001
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
101 md.qmu.responses.MassFlux12=response_function('indexed_MassFlux_12',[],[0.0001
    0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999]);
102 md.qmu.responses.MassFlux13=response_function('indexed_MassFlux_13',[],[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 identifier '*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.

```
105 md.qmu.mass_flux_profiles={'MassFlux1.exp',...
106                             'MassFlux2.exp',...
107                             'MassFlux3.exp',...
108                             'MassFlux4.exp',...
109                             'MassFlux5.exp',...
110                             'MassFlux6.exp',...
111                             'MassFlux7.exp',...
112                             'MassFlux8.exp',...
113                             'MassFlux9.exp',...
114                             'MassFlux10.exp',...
115                             'MassFlux11.exp',...
116                             'MassFlux12.exp',...
117                             'MassFlux13.exp'};
118 md.qmu.mass_flux_profile_directory='../Exp_Par/MassFluxes/';
```

Setup sampling engine:

Fourth, we need to decide on a sampling strategy:

```
120  %%  sampling analysis
121  md.qmu.method      =dakota_method('nond_samp');
122  md.qmu.method(end)=dmeth_params_set(md.qmu.method(end),...
123  'seed',1234,...
124  'samples',20,...
125  'sample_type','lhs'); %random or lhs
```

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

We determine the number of samples (20 for now) and also the type of sampling algorithm (*'lhs'* or *'sampling'*).

Some parameters:

We setup persistent parameters:

```
127  %% a variety of parameters
128  md.qmu.params.evaluation_concurrency=1;
129  md.qmu.params.analysis_driver='';
130  md.qmu.params.analysis_components='';
131  md.qmu.params.tabular_graphics_data=true;
132  md.verbose=verbose('qmu',true);
133
134  md.diagnostic.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).

FIG: UQ Application

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

Solve:

We solve:

```
136 %solve
137 md.qmu.isdakota=1; md.inversion.iscontrol=0;
138 md=solve(md,DiagnosticSolutionEnum,'overwrite','y');
139 md=tres(md,'dakota');
```

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

Here, we try and quantify the importance factors (sensitivities scaled by error margins) for each model input (ice thickness H , basal friction α and ice rigidity B) given a 5% error margin on all inputs.

Partitioning is identical. Model inputs need to be added:

```
157 md.qmu.variables.Thickness=normal_uncertain('scaled_Thickness',1,0.05);
158 md.qmu.variables.DragCoefficient=normal_uncertain(...
159 'scaled_FrictionCoefficient',1,0.05);
160 md.qmu.variables.rheology_B=normal_uncertain(...
161 'scaled_MaterialsRheologyB',1,0.05);
```

Model outputs are identical. Algorithm is different: 'nond_l':

```
195 md.qmu.method      =dakota_method('nond_l');
196 md.qmu.method(end)=dmeth_params_set(md.qmu.method(end),...
197 'output','quiet');
198
199 %parameters
200 md.qmu.params.direct=true;
201 md.qmu.params.analysis_driver='diagnostic';
202 md.qmu.params.evaluation_concurrency=1;
203 md.qmu.params.interval_type='forward';
204 md.qmu.params.tabular_graphics_data=false;
205 md.diagnostic.restol=10^-5; %tighten for qmu analyses
```

FIG: UQ Application

Model Setup-2

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

We solve the same way:

```
209 md.qmu.isdakota=1;  
210 md.inversion.iscontrol=0;  
211 md.verbose=verbose('qmu',true);  
212 md=solve(md,DiagnosticSolutionEnum,'overwrite','y');  
213 md=tres(md,'dakota');
```

FIG: UQ Application

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

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.

```
235 index=1;
236
237 %retrieve results for this specific profile
238 result=md.results.dakota.dresp_dat(md.qmu.numberofpartitions+index);
239
240 %plot histogram
241 plot_hist_norm(result,'cdfleg','off','cdfplt','off','nrmlplt','off',...
```

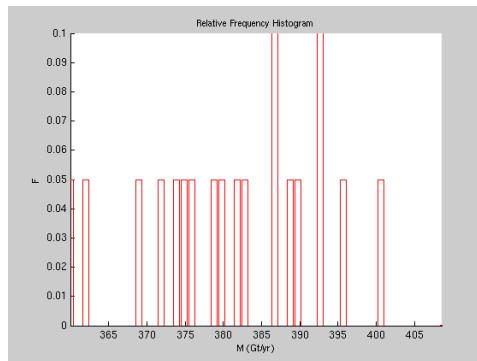


FIG: UQ Application

Introduction

Model setup

Sensitivity Analysis

Plot results

Conclusion

Plot sensitivity results

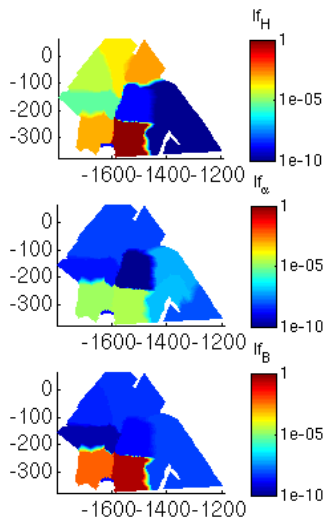
To retrieve importance factors for each model input:

```
253 index=1;
254
255 ifh=importancefactors(md,'scaled_Thickness',[ 'indexed_MassFlux_' ...
        num2str(index)]);
256 ifa=importancefactors(md,'scaled_FrictionCoefficient',[ 'indexed_MassFlux_' ...
        num2str(index)]);
257 ifb=importancefactors(md,'scaled_MaterialsRheologyB',[ 'indexed_MassFlux_' ...
        num2str(index)]);
```

To plot, as for any other field:

```
260 'expdisp#all',[ 'Exp_Par/MassFluxes/MassFlux' num2str(index) ...
        '.exp'], 'expstyle#all', 'b-', ...
261 'linewidth#all', 2, ...
```

FIG: UQ Application

[Introduction](#)[Model setup](#)[Sensitivity Analysis](#)[Plot results](#)[Conclusion](#)

Conclusion

UQ allows the following:

- assess propagation of errors from model inputs to model diagnostics
- assess sensitivities 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.

Thanks!

