Ice Sheet System model

Sensitivity Analysis: user guide (Dakota)

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

ISSM

December 2011

Outline

Motivation

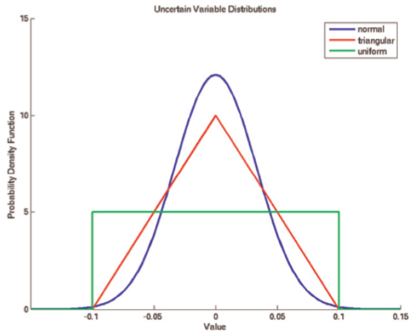
- Assessing output errors of ice flow models is a major challenge.
  - Constraints in ice flow models include:
    - Geometry (thickness and surface elevation)
    - Boundary conditions (geothermal flux, basal drag, surface temperature)
  - Errors in input data come from:
    - Measurement (instruments)
    - Calculation (inverse methods)
  - Input errors result in uncertainties that propagate across the model and influence output results.

DAKOTA

- The DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) toolkit provides:
  - Interface between analysis codes and iterative systems analysis methods.
  - Iterative methods:
    - Uncertainty quantification
    - Sampling
    - Reliability
    - Sensitivity analysis
    - Parameter estimation
    - Design optimization
  - `http://dakota.sandia.gov/index.html`

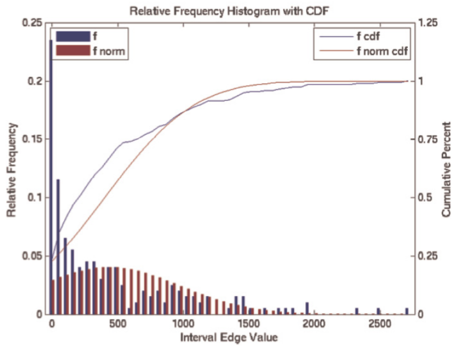
Sampling Analysis (1/4)

- In order to perform a sampling analysis for uncertainty quantification:
  - Uncertain input variables are defined with a statistical distribution:
    - Normal
    - Uniform
    - Triangular
    - Etc.



Sampling Analysis (2/4)

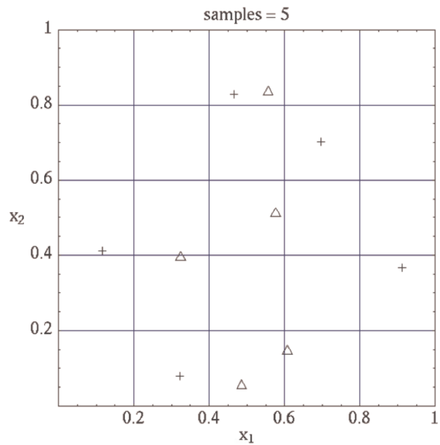
- Repeated analyses are run with values of the input variables generated from the distributions.
  - Statistics are calculated on the output responses:
    - Means
    - Standard deviations
    - Cumulative distribution functions (CDFs)



Sampling Analysis (3/4)

Generating the values of the input variables can be done in a number of ways:

- Monte Carlo (MC):
  - Generated randomly.
  - Tails, which are often critical for UQ, may be neglected.
- Latin Hypercube Sampling (LHS):
  - n-Dimensional variable space is divided into equal-probability bins.
  - One and only one sample occurs per bin.
  - Forces samples into tails.
  - More efficient method of sampling.
- At right, random (Δ) and LHS (+) points are shown.
  - From DAKOTA Users5.1.pdf



Sampling Analysis (4/4)

For a large number of input variables, the cost of sampling analysis to decrease the confidence intervals of the output responses to desired levels may be prohibitive.

- At a minimum must use at least two samples per variable to have any hope of attributing changes in responses to changes in variables.
- 95% confidence intervals are calculated for the mean and standard deviation of each response.
- Size of confidence intervals decreases with 1/sqrt(n).
- Other methods may be used to decrease the number of input variables.

Local Reliability Analysis (1/3)

A local reliability analysis may be performed to determine the input variables that have the most significant effects on the output responses.

- This method calculates a finite difference partial derivative for each output response with respect to each input variable at their baseline values.
- Requires only n+1 solutions.
- Variables that have the largest effects can be studied further.
- Sampling or parameter methods may be used.
- Those with little or no effect might not be of interest.

Local Reliability Analysis (2/3)

Given a response r that is a function of n multiple input variables X_i:

$$r = r(X_1, X_2, ..., X_n)$$

The sensitivities θ_i are defined as:

$$\theta_i = \frac{\partial r}{\partial X_i}$$

The finite-difference step size is typically defined by the user, so if the function is not linear in the neighborhood of the baseline solution, the value of the secant will change.

- One-dimensional parameter studies (or different step sizes) can be used to ascertain the behavior.

Local Reliability Analysis (3/3)

- The mean of the output responses are assumed from the baseline value.
- If each of the input variables is independent, the variance σ_r^2 of the output response can be computed from the well-known error propagation equation, where the σ_i^2 are the specified variances of each input variable:

$$\sigma_r^2 = \sum_{i=1}^{n} \theta_i^2 \cdot \sigma_i^2$$

- Importance factors IF_i for each input variable may be calculated by dividing each right-side term by σ_r^2:

$$IF_i = \frac{\theta_i^2 \cdot \sigma_i^2}{\sigma_r^2}$$

- These importance factors provide non-dimensional quantities:
  - They add up to unity.
  - Therefore they can be used the rank the contributions of the input variables.

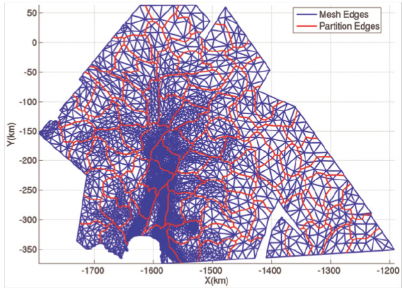
Mesh Partitioning (1/5)

- Both the sampling and local reliability methods are based on updates of input variables.
- For a spatially distributed input variable which covers the entire domain, such as thickness or basal drag, the domains must be partitioned into a number of discrete regions to be updated.
  - The finite element mesh provides a convenient discretization of the domain.
  - However, varying the input variable for each finite-element node or element would be prohibitive for very large problems.
  - In addition, there is the problem of physical size.
    - For anisotropic meshes with differently sized elements, some variables would have an inordinate contribution to the response given the physical areas over which they extend.

Mesh Partitioning (2/5)

An example of equal-sized partitions is shown on the right:

- Pine Island Glacier
- 100 partitions
- Mesh edges in blue, partition edges in red.
- Carried out using the Chaco package with nodal weighting.



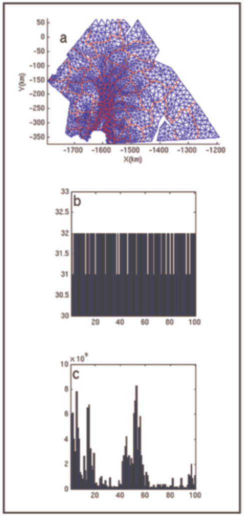
Mesh Partitioning (3/5)

- For each partition surface, a statistical distribution is specified for the field being sampled.
  - For example, if thickness is being considered, error margins on measurements from GPR can be used to specify the 3σ (99%) standard deviation.
  - The average value of the thickness can be used to specify the mean value of the field.
- Each node that belongs to the partition area will behave accordingly.
- Since thicknesses are specified at nodes, not elements, they will be linear over the elements between the partitions with no discontinuities.
- Same as in a customary finite element analysis.
- Sampling will be carried out, not over the entire field, but one field partition at a time.

Mesh Partitioning (4/5)

- Three software packages were considered:
  - MeTiS: a Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices
  - Chaco: Software for Partitioning Graphs
  - SCOTCH: Software package and libraries for sequential and parallel graph partitioning, static mapping, and sparse matrix block ordering, and sequential mesh and hypergraph partitioning
- All three have the goal of reducing parallel computing time for matrix solutions.
  - Methods are based on a nodal graph of finite element connectivity.
  - Each element area was divided by the number of nodes, and that area was assigned to the weighting of each node.
- Chaco was chosen for best continuous partitions.
  - Since source code was available, tight interfaces were written to pass data back and forth within memory.
  - For current sampling and local reliability analyses, only one partitioning needs to occur for the entire analysis.

Mesh Partitioning (5/5)

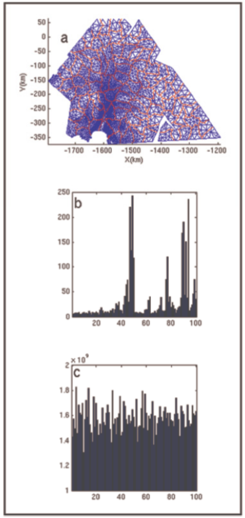


On the left, the mesh has been partitioned by number of nodes.

- 31 or 32 nodes in each partition (b).
- Areas vary by two orders of magnitude (c).

On the right, the mesh has been partitioned by area assigned to nodes.

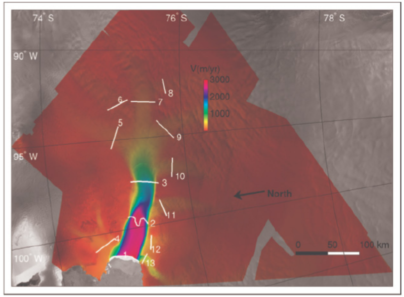
- Number of nodes varies from <10 to nearly 250 (b).
- Areas are much more consistent (c).



Application to ISSM

Responses

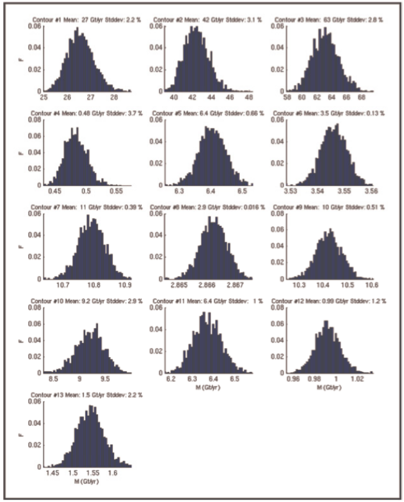
- On the right, 13 mass flux gates (in white) are shown.
  - Gate 1 is the ice front.
  - Gate 2 is the 1996 grounding line.
  - Others are tributaries.
- The mass fluxes through these gates are the output responses for the UQ analyses.
- Background is InSAR surface velocity map.



Application to ISSM

Sampling

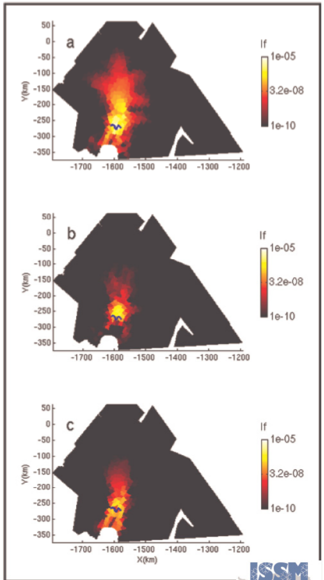
- On the right, the histograms for the mass fluxes at the 13 gates are shown.
  - The input variables were normal distributions of thickness in each of 200 partitions.
  - Mean and standard deviations based on measured data.
  - Uniform distributions used in a separate run.
  - 2000 LHS samples were run.
- Mean and standard deviation were calculated for each output response (and are displayed in each title).



Application to ISSM

Local Reliability

- On the right, the importance factors for the mass flux at Gate 2 (1996 grounding line, in blue) are shown for:
  - Thickness (a)
  - Basal drag coefficient (b)
  - Ice rigidity (c)
- These provide insight as to which parts of the model are most important, relative to the particular output response.
  - Provide a sanity check.
  - May allow other areas to be neglected.



Setup a sampling analysis

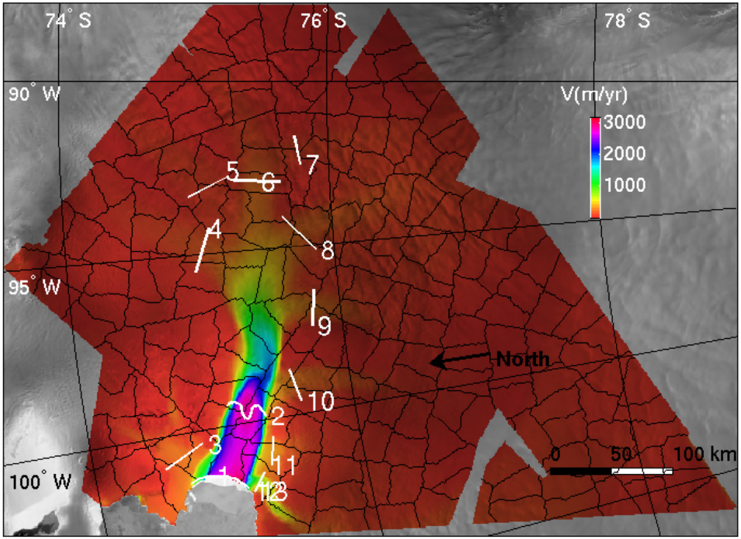
Sampling analysis of thickness and impact on maximum velocity on Pine Island Glacier. Thickness is sampled assuming a gaussian distribution centered around the average thickness measurement, and a 5% uncertainty range. We rely on the Matlab Dakota of ISSM, so that Dakota input files can be pre-processed by ISSM.
First step is to partition the mesh:

```
md.npart=200;
md=partitioner(md,'package','chaco','npart',md.npart,'weighting','on');

%md.npart=md.numberofnodes;
%md=partitioner(md,'package','linear');

md.part=md.part-1;
```

To plot the partition:

```
plotmodel(md,'data','mesh','partitionedges','on');
```

Setup a sampling analysis (2)



Setup a sampling analysis (3)

Second step is to setup the variable inputs, i.e. the variable being sampled for:

```
md.qmu.variables.thickness=normal_uncertain('scaled_Thickness',1,.05);
```

This assumes a scaled average thickness of 1xthickness and a σ standard deviation of 5%, for a gaussian distribution ('normal uncertain variables in the Dakota user guide').
One can also setup a uniform distribution:

```
md.qmu.variables.thickness=uniform_uncertain('scaled_Thickness',.95,1.05);
```

One can add several variables if needed:

```
md.qmu.variables.surface=uniform_uncertain('scaled_Surface',.95,1.05);
```

All variables will be sampled by the Dakota engine and handed directly to your solution (diagnostic_core, transient_core, etc.) no matter what the solution, no matter what the model being run. I.e: you will have access to the new variable for each sample run, after the variable has been sampled and scaled accordingly.

Setup a sampling analysis (4)

Third step is to setup the output response computed after each sample model is run:

```
md.responses.MaxVel=response_function('MaxVel',[],[0.0001 0.001 0.01 0.25 0.5 0.75 0.99 0.999 0.9999])
```

Each output response will be computed for every run, as well as output statistics such as average and standard deviation (assuming a gaussian distribution of the results).

Setup a sampling analysis (5)

Fourth step is to setup the engine driving the sampling analysis:

```
md.qmu_method     =dakota_method('nond_samp');
md.qmu_method(end)=dmeth_params_set(md.qmu_method(end),...
'seed',1234,...
'samples',2000,...
'sample_type','random',...
'output','debug');
```

A lot of the parameters can be found in the Dakota user guide, and map directly into the Matlab interface implemented in ISSM.
Sampling can be switched between 'random' (for Monte-Carlo) and 'lhs' among others. Number of samples can also be controled. Choose 20-30 samples per partition of your mesh?

Running a sampling analysis

Settings can be found in md.qmu:
To run qmu (Quantification of Margins and Uncertainties) analysis, just activate

```
mq.qmu.analysis=1;
```

and run your solution as usual:

```
md=solve(md,DiagnosticSolutionEnum);
```

or any other solution:

```
md=solve(md,ThermalSolutionEnum);
md=solve(md,HydrologySolutionEnum);
```

Try and run these in parallel, as Matlab and mex API tend to conflict with Dakota + it's slow.

Outputing sampling analysis results

All results are in results.dakota, for each variable and each output variable:

```
md.results.dakota

ans =

dresp_out: [1x13 struct]
scm: [1x1 struct]
pcm: [1x1 struct]
srcm: [1x1 struct]
prcm: [1x1 struct]
dresp_dat: [1x213 struct]
```

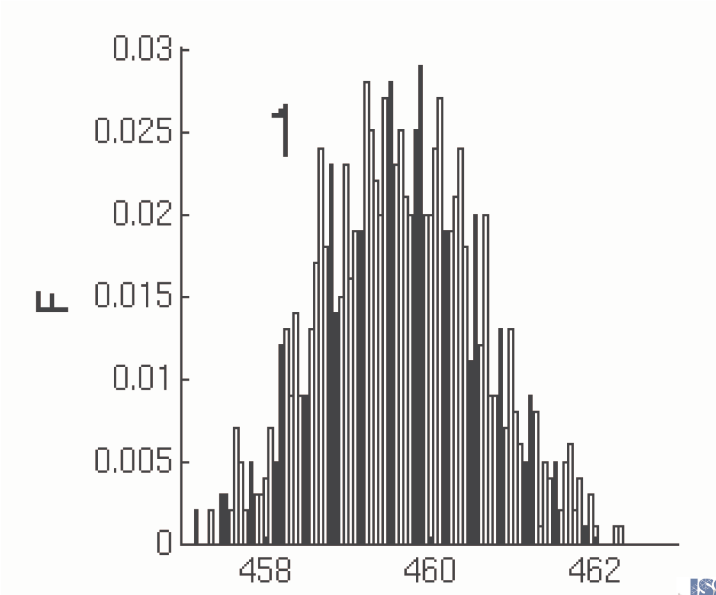
Get mean and stddev of your result number j (if you have several responses)

```
mean=md.results.dakota.dresp_out(j).mean;
stddev=md.results.dakota.dresp_out(j).stddev;
```

Plot a histogram of your results:

```
plot_hist_norm(md.results.dakota.dresp_dat(md.npart+j),'cdfleg','off','cdfplt','off','nrmplt','off',..

'xlabelplt',xlabelplt,'ylabelplt',ylabelplt,'FontSize',8,'FaceColor','none','EdgeColor','red');
```

Outputing sampling analysis results (2)



Conclusions

- 1 Many other types of analyses can be run: parameter space studies, local reliability analyses, optimization analyses.
- 2 Every analysis, parameter, variable input and output has been mapped from Dakota into ISSM. Follow the Dakota user guide and you will have a good idea of what is implemented in ISSM.
- 3 This is still a prototype interface. It's getting more stable, but this is "dangerous" piece of code.


Thanks!