Outline

* **Problem and Impact**
* **Background**
* **Technical Approach**
* **Implementation with Different Programming Languages**
* **Methodology to Test and Compare Performance**
* **Results**
* **Key Lessons Learned**
* **Future Work**

1. **Problem and Impact**

* Context: Simulation metamodels approximate simulation models and enable rapid systems analysis
* Problem: Evaluating the accuracy of mapping a simulation model by metamodels is important, and this forms the problem for this work

Diagram

Description automatically generated

* Impact: this work addresses computational issues in fitting and generating error measures of simulation metamodels

Approach for Validating Simulation Metamodels

* We use two metrics to compute the accuracy of candidate metamodels:
  1. PRESS – to compare the bias for the two candidate classes of metamodels
     + An estimate of the bias plus the inherent variability in the model
     + Predictive residual or accuracy
  2. Bootstrap Standard Error – To characterize stability of the estimate of the metamodel through computing standard error
     + Measurement of the standard error of the actual model
     + Model stability measure

Approach – Standardized Predicted Residual Sum of Squares (PRESS) Statistic

Text, letter

Description automatically generated

* Where f(hat) is the estimate of the true model within the metamodel family that does not use the points associated with the design point xi,
  + m is the number of design points
  + n is the number of replications
  + MSE – mean squared error
  + PRESS – Predicted Residual Sum of Squares

This measures the estimate percent increase in error due to model bias, this is how much we lose by having bias or approximate solution in the metamodel.

Approach – PRESS Statistic

Diagram

Description automatically generated

This measures how stable the model is. This is a flow chart of how to calculate the standardized PRESS statistic. We first calculate our mean squared error through standard regression model but treating the inputs as straight categorical responses. We don’t want to use actual measurements or observations. For example, yes if there’s a mean response for design point 1, no if there’s no mean response for design point 2.

Approach – Bootstrap Statistic

* The Bootstrap statistic recalibrates the metamodel at each bootstrap sample
* This provides insight into the sensitivity of the goodness-of-fit of the metamodel

Diagram

Description automatically generated

This is measuring the stability of our actual model fit. It doesn’t care whether the model is biased or not. It only gives the standard error of the models. To do this we take the full dataset and draw samples from it. We draw two samples from two different Bootstrap samples and fit our metamodel and calculate the difference between them.

Implementations

* Error statistics were implemented in:
  + MATLAB
  + R
  + Python
  + Python with Message Passing Interface (MPI)
* All scripts read in the same standard CSV files
* Recommended implementation by language documentation was used
  + MATLAB

Graphical user interface, text, application

Description automatically generated

* R

Graphical user interface, text, application

Description automatically generated

* Python

A close up of a person

Description automatically generated

Experimental Implementation

* Timing results are the mean of three runs of the analysis scripts
* Language-specific profiling functions were used
* MATLAB – MATLAB profile tool
* R – Rprof
* Python – cProfile
* Each of the profiling functions gives detailed information on time spent inside child functions
* Windows 64-bit laptop with intel i7 Processor
* 4 Cores and 8GB of RAM
* Experiments are from a data set generated by a Border crossing Scenario Model
* Three independent predictor variables
* Single response variable
* 200 design points, 10 replications
* An experiment consists of fitting on five different metamodels
  + - is the set of values representing the model’s best fit
    - p is the number of predictor variables, x1, x2,…xp
    - f0 is the metamodel
* Second Order with interactions
* Third Order
  + f0(x) = 1x1 + 2x2 + 3x21 + 4x22 + 5x31 + 6x32
* Third Order with interactions
  + f0(x) = 1x1 + 2x2 + 3x21 + 4x22 + 5x31 + 6x32  + 7x1x2  + 8x21x2 + 9x1x22
* Fourth Order
* Fourth Order with interactions
* values that need to be computed depend on the order of the model, k, and the number of predictor values, p

Table

Description automatically generated

Results

* Data Set 1: Border Crossing Scenario

Chart, scatter chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Key Lessons Learned

* Smaller data sets:
  + - Metamodel fitting performs better in R and Python
    - Subsequent error statistic computations are better performed in R or Python
  + Larger data sets
    - The merit of High-Performance Python is visible and shown to be valuable compared to MATLAB and R
  + The speedup of High-Performance Python is best recognized with complex metamodels and large datasets

Future Work

* Further investigation of different statistical computation implementations
* Develop a complete pipeline for simulation runs, metamodel fitting, and error computations

Diagram

Description automatically generated