SANDIA REPORT
SAND2024-15407
Printed November 2024

Sandia National Laboratories

Sensitivity Analysis for the Component Design App: Analysis of Success Assured Data

Talia G. Duffy

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

A new tool has been developed to perform variance-based global sensitivity analysis (VBGSA) on data from a set-based concurrent engineering software called Success Assured (SA). The tool is part of a digital component design app, which is currently in production as an Accelerated Digital Engineering Pathfinder at Sandia National Laboratories.

When working with complex digital models, it is important to understand relationships between inputs and outputs, i.e., how "sensitive" model outputs are to changes in model inputs. After extensive research and trials of various sensitivity analysis methods, it was determined that estimation of Sobol' indices for VBGSA with Monte Carlo simulation, paired with simple surrogate models, produces the best results for SA data. This tool increases understanding of SA models and streamlines the creation of SA datasets.

This report details the methodology and implementation of this sensitivity analysis tool so others can understand it and implement it.

This page left blank

ACKNOWLEDGEMENTS

Many thanks to all who helped me with this project along the way, including, but not limited to, Kathryn Hoffmeister, Steve Crowder, Josh Allers, Taylor Hall, Brendon Mizener, Andrew Clark and Nancy Lies.

I was lucky to work on a super cool project with a super cool team.

This page left blank

CONTENTS

FIGURES

TABLES

This page left blank

EXECUTIVE SUMMARY

Sandia National Laboratories is developing a new component design app to work with data from Success Assured, a set-based concurrent engineering application. However, the digital models in SA and their resulting datasets are extremely complex, so not every model input and output can be included in the dataset that will populate the new design app.

Engineers working with the design app want to be sure they have generated the best SA dataset possible, in which the most important inputs (design parameters) and outputs (design requirements) have been selected to answer the design question at hand. A set of statistical methods called "sensitivity analysis" can be leveraged to solve this problem. This report focuses on a new sensitivity analysis tool developed for SA data.

The new tool, which currently has versions in R and MATLAB, implements the Sobol' method of variance-based global sensitivity analysis, which calculates sensitivity "indices" for each input that represent the proportion of output variance attributable to changes in each input. This report details the math behind Sobol' indices, the Monte Carlo process and surrogate models used to estimate them, and how everything was altered to work with SA data. The tool calculates the sensitivity indices and presents them in tables and bar charts for easy interpretation.

With this information, engineers can assess the relative importance of each design parameter and decide which to keep and which to leave out of the design app's dataset. The tool also performs related analysis that locates highly sensitive areas of the input space and indicates whether to add any hidden design parameters. Overall, the tool increases understanding of the complex digital models in SA and creates a faster, guided process for creating SA datasets instead of leaving it up to frustrating trial-and-error.

The tool is featured as its own sensitivity analysis tab in the MATLAB-based digital component design app.

This page left blank

ACRONYMS AND TERMS

Acronym/Term	Definition
ADE	accelerated digital engineering
ASC	Advanced Simulation and Computing
csv	comma-separated values (file type)
MBSE	model-based systems engineering
RMSE	root mean squared error
SA	Success Assured (set-based concurrent engineering software)
SBCE	set-based concurrent engineering
SME	subject matter expert
S_j	main effect sensitivity index for input X_j
S_{Tj}	total effect sensitivity index for input X_j
VBGSA	variance-based global sensitivity analysis

This page left blank

1. INTRODUCTION

Inefficient communication between engineers can unnecessarily delay the design process. These delays often result from shortcomings in engineering software: certain apps can be slow, uninterpretable, or difficult to integrate with other software used for the same project. One such software is a set-based concurrent engineering (SBCE) app called Success Assured (SA).

Ideally, questions such as the following should be answered quickly and with sufficient evidence so the design process can move forward:

- Are there any viable designs that operate with this voltage?
- Can we double the radius here and keep our mass low enough to meet requirements?
- If we compare these two similar designs side-by-side, which is more reliable?

This Accelerated Digital Engineering (ADE) Pathfinder project transforms data from SA into an intuitive digital component design app that's useful in these situations.

But before the app runs, the team must know whether they exported the right data from the digital models in SA. Are the inputs relevant to the outputs, and is this dataset useful to answer the design question at hand? A branch of statistics called "sensitivity analysis" can help answer this question. This report details different sensitivity analysis methods that were explored for component design data, how the chosen method was altered to work best with the data, and how the final analysis was implemented as a sensitivity analysis tool for and within the design app.

This page left blank

2. SENSITIVITY ANALYSIS

Suppose we are presented with a model in which an output or response, y, is determined by some function of n input variables, x_j.

$$y = f(x_1, x_2, \ldots x_n)$$

Uncertainty is inherent to all models. We don't have an exact value for our output; it varies with different combinations of the inputs and their transformation through the model function.

Sensitivity analysis is a set of statistical methods that describes, quantitatively or qualitatively, how uncertainty in our model output is impacted by uncertainty in our model inputs (Saltelli 2004). In other words, sensitivity analysis can determine how "sensitive" our model output is to changes in each of the model inputs.

For example, if changing x_1 results in significant changes in y, we can say that y is highly sensitive to x_1. But if we change x_2 and see comparatively little change in our response y, we can say that y is not sensitive to x_2.

Sensitivity analysis has wide applications in science and engineering, and it is now considered an essential step in the modeling process (Saltelli 2002).

2.1. Methods of Sensitivity Analysis

There are many viable methods for performing sensitivity analysis, and the choice depends on the model, the variables, and the overall question at hand. Sensitivity analysis can be divided into two subfields: local sensitivity analysis and global sensitivity analysis.

2.1.1. Local vs. Global Sensitivity Analysis

Local sensitivity analysis focuses on how the output is impacted by isolated changes in one input. These methods assume all other inputs are "fixed." Local sensitivity analysis measures include correlation coefficients and evaluations of partial derivatives, and they're not very effective for complex or nonlinear models (Tang 2006).

Global sensitivity analysis allows for investigation of the entire input space, including interactions between inputs. When quantifying the sensitivity of the output to one of its inputs, we assume all other inputs are free to vary as well, which more accurately reflects the true behavior of the model. The most popular global methods focus on input and output variance as a representation of uncertainty (Saltelli 2002).

Datasets pulled from Success Assured (SA) often are high-dimensional. Additionally, the digital component models that generate SA data are complicated and design parameters should be assumed to interact with each other.

Since our goal is to understand the sensitivity relationships of the dataset as a whole and ensure engineers have created a robust design space for the app, global sensitivity analysis methods are most appropriate for this project.

2.1.2. Methods Explored for this Project

Several methods of sensitivity analysis were explored for implementation with the SA data—with varying levels of success.

First, a non-standard approach to variance-based global sensitivity analysis (VBGSA) was attempted. This approach provided realistic importance rankings of the input variables, but the sensitivity measures did not adhere to expected mathematical properties.

To test the suitability of different sensitivity analysis subfields on SA data, a local sensitivity measure involving partial derivatives was also explored in the initial stages of the project. Likely due to the reasons explained in Section 2.1.1, the results were not reliable. This attempt was not relevant to the final tool, so it is not described in this report.

Results indicated that VBGSA approaches reflect component model behavior more accurately than local methods. Consequently, a more documented and robust approach to VBGSA was attempted. This method leverages Monte Carlo simulation to estimate Sobol' sensitivity indices and is explained in detail in Section 6. To effectively run Monte Carlo simulations, simple surrogate models are created based on the SA dataset to obtain reasonable model outputs at simulated sets of inputs.

This produced the best and most consistent results on the test dataset, so it became the focus of the sensitivity analysis tool. Information about how different methods were validated and compared can be found in Appendix D.

2.2. Applications of Sensitivity Analysis

What happens after sensitivity analysis is performed on a model? It depends on the needs of the project and the reason a sensitivity study was requested in the first place.

Sometimes the goal of sensitivity analysis is to generally improve understanding of relationships between different variables. Other times, it can be used to drop or fix input variables considered unimportant in a model. Conversely, sensitivity analysis can provide direction for future experiments and data collection, which should focus on the more important variables.

It's a broad field, but the general idea is to narrow down which variables are most important or influential and use that information to benefit the project (Razavi 2021).

3. COMPONENT DESIGN APP

The model-based systems engineering (MBSE) portion of this ADE Pathfinder project is creating an application (a "component design app") that gives insight into the design space of individual components in a weapons system. This includes design parameters as well as requirements the components must meet.

Parameters are controllable characteristics of the design such as size dimensions or amount of power supply.

Requirements can be thought of as "standards" that the part—or the system as a whole—must live up to once it is built. For example, a part must survive for at least 10 years, or the system must perform an operation successfully 98% of the time. Whether or not a design meets requirements is theoretically determined by the chosen combination of design parameters. It is also possible that controllable design parameters are subject to requirements, which restrict the range of design choices.

The goal of the new design app is to promote dynamic and interpretable analysis of how design parameters and requirements interact with each other. Users can change inputs, view sets of designs that match their choices, and determine whether the designs are viable based on requirements. The sensitivity analysis tool ensures the design app meaningfully and successfully achieves this goal.

With this app, engineers can answer design questions faster and with evidence to back it up: "yes, that's fine," or "no, absolutely not," or "maybe, as long as you're okay with losing a year of the part's lifetime."

3.1. App User Interface

On the app's interface, users can change the values of design parameters with sliders. Within this SAND report, parameters chosen for the sliders are considered "inputs."

The center of the app's main tab is a three-dimensional model of the component that changes based on the current design settings. A table of requirements sits to the right of the visualization and uses color-coding to indicate whether the design meets each requirement. Within this SAND report, design elements that are the focus of the design question are considered "outputs," which usually end up being requirements.

The app, written and designed by Joshua Allers (Virtual Technologies & Engineering, 0291) and Brendon Mizener (Computer Simulation Infrastructure, 1545), is based in MATLAB and runs with MATLAB's interactive application functionality. Figure 3-A represents a recent version of the app, but the project is ongoing, and the app will continue to change after publication of this report.

Figure 3-A. A recent version of the user interface of the main tab in the component design app.

3.2. Choosing Data for the App

The app is populated with data from Success Assured (SA), an SBCE software. As part of generating and exporting data from SA for use in the app, engineers must make a series of subjective decisions, such as which design parameters to display or leave out, whether a design element is considered an input or an output, and the range of possible values for each input. Section 4 of this report covers SA in more detail, including how it generates data based on digital models of components.

While the engineers generate the best datasets they can, based on their knowledge of the component, some sort of statistical assessment of the dataset is essential before populating the design app and presenting results.

3.3. Benefits of Sensitivity Analysis for the Design App

Recall the purpose of the component design app: in the context of some design question or proposed change, users can control inputs and consider the effects on key requirements and other model outputs. But if the dataset generated from SA consists of inputs that have no influence on the outputs, the app will not effectively answer the design question, providing misleading answers or no answers at all.

After running the dataset through the sensitivity analysis tool, engineers will receive statistics and visualizations that indicate whether each input is "important." Based on that information, they can fix or drop unimportant variables. This frees up space in the dataset for more detailed values in important design parameters.

The sensitivity analysis tool includes additional analysis that indicates where the input space might need more detailed data and whether the outputs may be sensitive to an omitted design parameter.

Datasets from SA are large and take time to generate and export. Sensitivity analysis will provide better guidance for variable selection, shortening the trial-and-error process for creating an appropriate dataset.

Ideally, the dataset will include all variables that play a significant role in the design question, with the most detailed data going to the most important inputs.

4. SUCCESS ASSURED DATA AND ITS CHALLENGES

Success Assured (SA), a set-based concurrent engineering (SBCE) software, is used to explore the design space of complex component models. It is used in early decision-making stages.

4.1. Set-Based Concurrent Engineering

In contrast with traditional engineering approaches in which engineers design and test one model at a time and start over if a failure occurs, in SBCE, groups of designs are explored together and narrowed down until the most successful design is chosen.

Proponents of SBCE maintain that it improves efficiency, especially when combined with digital engineering tools (Raudberget 2010).

In SA, engineers can analyze the trade-offs and effects of different decisions within a high-dimensional model. As understanding of the model increases, engineers reduce the full set of models to a select few and analyze those in more detail.

The design app replicates some of this functionality.

4.2. Generation of Data

Success Assured is the home of all design information for digital component models: parameters, requirements, the relationships between them, etc. Users customize the interface to display certain features as inputs and outputs, and the results can be exported as a comma-separated values (.csv) file.

Certain characteristics of SA data caused challenges throughout the project. This section outlines those challenges and provides the corresponding solutions that were implemented to prepare the data for sensitivity analysis.

4.2.1. Choice of Inputs and Outputs

In SA, engineers select a subset of parameters and requirements from the design space to display as sliders and curves on a trade-off graph. The sliders are considered "inputs," as their settings determine the shape of the trade-off curves. As a result, the two features that make up the curve are considered "outputs." Any feature in the model can be chosen as either an input or an output, depending on the design question.

For simpler models, it is common to include all design features; in this case, the outputs are no longer represented as curves, but as tight ranges that can be viewed as a single point in the output space. The component that generated the test dataset is too complicated for total parameter inclusion. This is why sensitivity analysis is needed: to assess whether the subjectively chosen subset of features is meaningful.

Each controllable slider in SA consists of a discrete set of values—not a continuous range. The ranges and spacing of these discrete values are also determined by the dataset creator. The test dataset was highly discrete, as inputs were given only two to five different values. This ensured the size of the test dataset was manageable for tool development.

During Monte Carlo sampling processes (see Section 6), the inputs were sampled from a continuous uniform distribution, from the input minimum to the input maximum.

Success Assured does not include experimental design functionality, so the dataset consists of all viable combinations of designs.

4.2.2. The "Web"

The core of each digital component model in SA is a set of equations, lookup tables, and other calculation methods that define the relationships between design features. These relationships are organized in a "web," and when a user chooses a set of input values, those numbers are fed through the web (Figure 4 A).

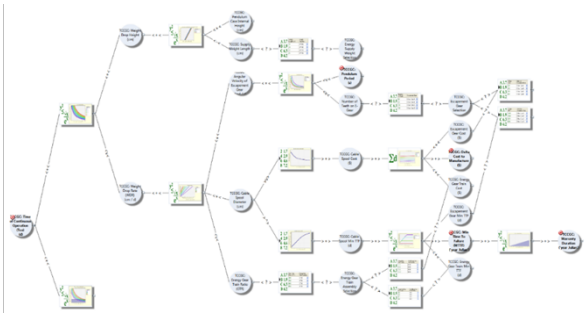

Figure 4-A. An example of a decision map, or "web," from Success Assured (Targeted Convergence Corporation 2024).

Because the "web" of relationships is large and difficult to extract for use outside of SA—especially for this project's test component—we treat the SA model as a "black box." In black box models, the decisions that transform input data into output data are unclear or hidden. The model itself cannot be effectively analyzed or understood.

Leading sensitivity analysis methods require some function to evaluate the output at simulated sets of inputs. However, we cannot realistically use the complicated SA webs for these functions. Instead, for this sensitivity analysis tool, individual model output evaluations were obtained by running simulated inputs through surrogate models (discussed in more detail in Section 6.3).

4.2.3. Outputs as a Curve

Another challenge of SA data is how the output values are generated and appear in the dataset. When a set of inputs is fed through the "web," a curve of the two outputs appears on the SA interface so engineers can analyze trade-offs that might occur for a set of designs.

Recall the description of how SA generates datasets: engineers select which inputs to include and which to leave out. When the user chooses a set of values on the input sliders, all other "hidden" variables are still free to vary.

The output curve that appears on the SA interface represents the output ranges that remain possible for the chosen set of inputs. It is important to note that the curve does not represent a functional relationship between the two outputs; it is simply a plot meant to show potential trade-offs.

For example, in in Figure 4-B, a clock design with an energy gear train cost of $102.50 and an escapement gear cost of $125 might have a cable spool cost anywhere from $0 to $35 and a delta

cost to manufacture anywhere from $225 to $260, depending on the values of the hidden inputs. The dataset exported by SA translates these curves into columns and rows.

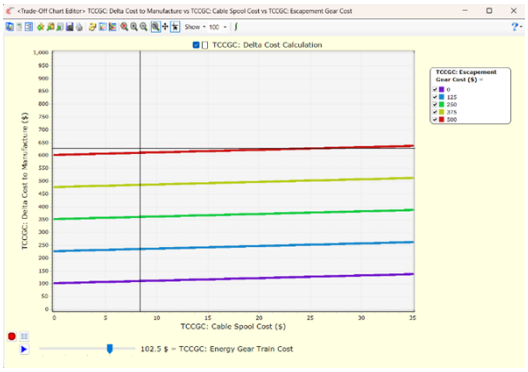

Figure 4-B. An example of a trade-off chart in Success Assured (Targeted Convergence Corporation 2024).

Commented [LNK1]: Ref 9

As an example, Figure 4-C shows how this is reflected in the test dataset used to generate the app: multiple rows are generated for one combination of inputs.

	output_2_min	output_2_max	output_1_min	output_1_max	input_1	input_2	input_3	input_4
1	40.5273	40.6250	1.3631	1.4604	0.2500	-1	0.2500	0.5000
2	40.5273	40.6250	1.4604	1.5578	0.2500	-1	0.2500	0.5000
3	40.5273	40.6250	1.5578	1.6552	0.2500	-1	0.2500	0.5000
4	40.5273	40.6250	1.6552	1.7525	0.2500	-1	0.2500	0.5000
5	40.5273	40.6250	1.7525	1.8499	0.2500	-1	0.2500	0.5000
6	40.5273	40.6250	1.8499	1.9473	0.2500	-1	0.2500	0.5000
7	40.5273	40.6250	1.9473	2.0446	0.2500	-1	0.2500	0.5000

Figure 4-C. A subset of the test dataset demonstrating how output curves are represented as columns and rows.

Notice that the curves generated by SA are made of small rectangles (Figure 4-D) that create the appearance of a continuous curve, and that each output is represented by two columns in the raw dataset.

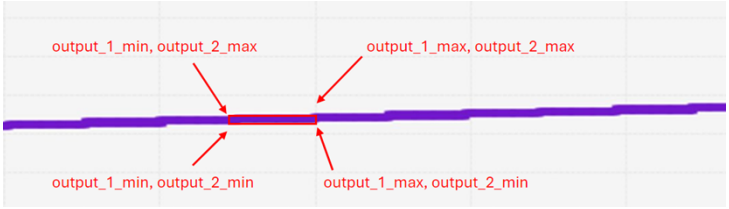

Figure 4-D. The same trade-off chart zoomed in to show the rectangular components of each curve.

Each row in the dataset corresponds to one of these rectangles. Output_1_min is the minimum value of the first output within a rectangle, and Output_1_max is its maximum value within that rectangle. The same is true for Output 2.

Together, sets of rows that have the same values for all inputs draw out curves like those seen in Figure 4-B.

With SA data, every row is part of a curve, which means that each row cannot be treated as an independent observation. This makes calculations like mean and variance across the dataset misleading.

To address this, the tool approximates the output curves with point measurements at their minimums, medians, and maximums, each stored as its own column. Each output variable is represented by these three columns in the dataset, resulting in six total output columns (Figure 4-E).

	output_2_min	output_2_max	output_1_min	output_1_max	output_2_med	output_1_med
1	40.3320	40.4297	2.7262	2.8235	40.3809	2.7749
2	40.3320	40.4297	2.8235	2.9209	40.3809	2.8722
3	40.4297	40.5273	2.7262	2.8235	40.4785	2.7749
4	40.3320	40.4297	2.9209	3.0183	40.3809	2.9696
5	40.3320	40.4297	3.0183	3.1156	40.3809	3.0669
6	40.7227	40.8203	0.0974	0.1947	40.7715	0.1460

Figure 4-E. A subset of the six columns used as outputs for the sensitivity analysis.

The sensitivity analysis tool calculates sensitivity measures for each input's effect on all six output columns separately, giving the user a concise yet informed assessment of how the output space varies with the inputs.

4.3. The Final Dataset

The raw test dataset used to develop the sensitivity analysis tool originally consisted of 4,304,150 rows with eight inputs and two outputs. The final, aggregated dataset was 7,130 rows, with eight input columns and six output columns to represent the minimum, median, and maximum of each output. Throughout this report, the inputs and outputs will be referenced by number (i.e., Input 1 or Output 2.)

5. VARIANCE-BASED GLOBAL SENSITIVITY ANALYSIS AND THE SOBOL' METHOD

5.1. VBGSA

Variance-based global sensitivity analysis (VBGSA) is considered the leading method in quantitative sensitivity analysis. Initial tests of VBGSA for this project produced interpretable and consistently accurate results, so it became the primary focus of the tool.

VBGSA methods, sometimes called "decomposition of variance," methods, are based on the idea that variance is a natural measure of uncertainty in model inputs and outputs. The goal is to quantitatively apportion output variance to each of the inputs, often called a sensitivity index or sensitivity score.

Even within the subfield of VBGSA, there are different methods of calculating sensitivity scores. The most popular approach is called the Sobol' method, first developed by Russian mathematician Ilya M. Sobol' and researched further by Andrea Saltelli. This approach as described in Sobol' and Saltelli's work will be recounted in detail in this report to provide an understanding of the sensitivity analysis tool (Sobol' 1993 and 2001).

5.2. Sobol's Method

The Sobol' method of VBGSA involves calculating sensitivity indices that can be used to rank inputs in terms of importance. Two types of indices can be calculated for each input X_j: the main effect indices, S_j, and the total effect indices, S_{Tj}.

These indices can be interpreted as proportions of output variance. For example, if the main effect sensitivity index for Input 1 is $S_1 = 0.33$, it can be said that 33% of all output variance is attributable to variance in Input 1. Higher values for a sensitivity index mean the output is more sensitive to changes in the corresponding input.

5.2.1. Main Effect Indices

Main effect, or first order, sensitivity indices S_j quantify the amount of output variance that remains if input X_j is fixed. In other words, if we remove variation in X_j and consequently see little variation in the output Y, we can assume that the value of Y heavily relies on changes in X_j. This indicates high sensitivity.

Mathematically, this is represented by

$$\frac{E\left(V\left(Y \middle| X_j\right)\right)}{V(Y)}$$

where the average output variance that remains after fixing X_j is taken as a proportion of the total output variance, $V(Y)$. Smaller values here represent higher sensitivity, based on the ideas described above. The main effect index accounts for the effect of that variable alone.

However, using the following property

$$V(Y) = E\left(V\left(Y \middle| X_j\right)\right) + V\left(E\left(Y \middle| X_j\right)\right)$$

we know if we can identify high-sensitivity inputs with the lowest values of $E(V(Y \mid X_j))$, we can do the same with the highest values of $V(E(Y \mid X_j))$.

Thus, we can rewrite the formula for main effect sensitivity indices as

$$S_j = \frac{V(E(Y \mid X_j))}{V(Y)}$$

This is more intuitive, as higher values for the index S_j would indicate that the output is more sensitive to that variable.

The sum of the main effect indices for each input is

$$\sum_{j=1}^{k} S_j \leq 1$$

This is because the main effect indices do not encapsulate the effects of interactions between inputs. Separate indices can be calculated for these interactions. Sensitivity indices for two-way interactions are calculated as follows

$$S_{i,j} = \frac{V(E(Y \mid X_i,X_j)) - V(E(Y \mid X_i)) - V(E(Y \mid X_j))}{V(Y)}$$

The independent effects of the inputs are subtracted, so the index represents the proportion of variance attributable to the inputs' interaction alone.

In theory, if all main effect indices are calculated, their sum is exactly 1. However, with high dimensional datasets, it is impractical to obtain estimates for all inputs and all interactions. This is known as the "curse of dimensionality." A total of $2^k - 1$ main effect indices must be calculated to obtain the complete set; for the test dataset consisting of eight inputs, this would amount to 255 different sensitivity indices for each output.

5.2.2. Total Effect Indices

Because of the curse of dimensionality, it's not common to calculate the full set of main effect indices unless the dataset is small. Instead, non-additive information about the model can be described with total effect, or total order, indices, which account for the effects of an input and its interactions with all other inputs (Homma and Saltelli 1996).

A total effect index S_{Tj} quantifies the amount of output variance that remains if everything is fixed except X_j. In other words, if X_j is the only input left to vary, and we still see lots of variation in the output Y, we can assume that the value of Y heavily relies on changes in X_j and the resulting interactions. This indicates high sensitivity.

Mathematically, this is represented by

$$S_{Tj} = \frac{E(V(Y \mid X_{-j}))}{V(Y)}$$

where the subscript $-j$ indicates every input variable except X_j. Each total effect index is the sum of all main effect indices that include X_j.

The sum of the total effect indices for each input is:

$$\sum_{j=1}^{k} S_{Tj} \geq 1$$

This is because some interactions are double counted within each index. For example, in a model with $k \geq 2$ input variables, S_{T1} includes the effect of X_1 interacting with X_2, and S_{T2} includes the effect of X_2 interacting with X_1, which is the same effect.

5.2.3. Calculation

Sobol' sensitivity indices can be calculated analytically or estimated numerically. The analytical process with integrals is often too computationally expensive for practical use, so numerical estimation using Monte Carlo sampling is preferred.

The standard for VBGSA is to estimate the main effect indices and the total effect indices for each input, for a total of $2k$ sensitivity indices. This provides a comprehensive picture of sensitivity relationships within the dataset.

This page left blank

6. MONTE CARLO SIMULATION FOR ESTIMATION OF SOBOL' SENSITIVITY INDICES

The sensitivity analysis tool uses the estimation process outlined in Saltelli's *Sensitivity Analysis in Practice* (Saltelli 2004). The process is outlined here with additional explanations and context for implementation with SA data.

Recall the definition of Sobol' sensitivity indices from Section 5. For the main effect index associated with X_j, we have:

$$S_j = \frac{V(E(Y \mid X_j))}{V(Y)}$$

and for the total effect index associated with X_j, we have:

$$S_{Tj} = \frac{E(V(Y \mid X_{-j}))}{V(Y)}$$

One method of calculating these indices involves Monte Carlo sampling of the input space to obtain estimates for the components these formulas (Saltelli 2004).

6.1. Sampling Procedure

To create a Monte Carlo sample of our input space, we first must define input probability distributions to sample from. However, such probability distributions do not truly exist. SA data is not experimental; instead, it is based on unique combinations of values pulled from user-controllable sliders.

In this context, it can be assumed that, for any input, no slider value is more or less likely to be chosen than another. Therefore, the only rational choice for a probability distribution is the uniform distribution.

$$X_j \sim Unif(min(X_j), max(X_j))$$

Minimum and maximum parameters are defined by the minimum and maximum slider values for each input.

An appropriate sample size N should be determined by the user. A common place to start is 5,000, which should be increased gradually if signs of nonconvergence appear, such as inconsistent results, negative indices, or sets of indices that do not add to one.

For the test dataset, a sample size of 10,000 was found to be sufficient, but computation time was low enough that higher sample sizes would also work if desired.

Once the input distributions and the sample size N are defined, we create two separate sample matrices with N rows and one column for each input (Figure 6-A). We will refer to these as M1, representing the "original sample," and M2, representing a "resample."

6.2. Estimates

6.2.1. Construction of Estimates

Let us assume we have N samples for each of the k input variables. From either of the sample matrices, the following estimators are used for the variance and expectation of the output:

$$\hat{E}(Y) = \frac{1}{N}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right)$$

$$\hat{V}(Y) = \frac{1}{N-1}\sum_{r=1}^{N} f^2\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) - \hat{E}^2(Y)$$

The estimate for variance is used as the denominator in the calculation of each sensitivity index.

The estimate for expectation, along with another estimate called the "U-value" (described later on this page) are used to calculate our numerators. For S_j, this is rather simple:

$$\mathrm{V}\left(E\left(Y \middle| X_j\right)\right) = U_j - \mathrm{E}^2(\mathrm{Y})$$

A similar result for S_{Tj} requires a few more steps.

Recall the identity

$$\mathrm{V}(Y) = \mathrm{V}\left(E\left(Y \middle| X\right)\right) + \mathrm{E}\left(V\left(Y \middle| X\right)\right)$$

Therefore, we have for the numerator of our total effect indices:

$$\mathrm{E}\left(V\left(Y \middle| X_{-j}\right)\right) = \mathrm{V}(Y) - \mathrm{V}\left(E\left(Y \middle| X_{-j}\right)\right)$$

$$\mathrm{E}\left(V\left(Y \middle| X_{-j}\right)\right) = \mathrm{V}(Y) - U_{-j} + E^2(Y)$$

These U-values are estimated as follows, with the symbol $\odot$ indicating scalar multiplication of the model outputs:

$$\hat{U}_j = \frac{1}{N-1}\sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) \odot f\left(x_1^{(r')}, x_2^{(r')}, \ldots, x_j^{(r)}, \ldots, x_k^{(r')}\right)$$

$$\hat{U}_{-j} = \frac{1}{\mathrm{N}-1}\sum_{\mathrm{r}=1}^{\mathrm{N}} f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) \odot f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_j^{(r')}, \ldots, x_k^{(r)}\right)$$

where r indicates that the variable comes from the original sample matrix M1, and r' indicates that the variable is "resampled," or switched with the column from resample matrix M2. This simulates changing or varying those inputs. The input matrix where X_j is taken from M2 is named N_j, and the input matrix where all except X_j is taken from M2 is named N_{-j}. An example of this is shown in Figure 6-A.

More information about these U-values and how to interpret them can be found in Appendix B.

- $X_1 \sim Unif(0, 1)$
- $X_2 \sim Unif(1.5, 2)$
- $X_3 \sim Unif(0.5, 0.75)$
- N = 5

- $X_1 \sim Unif(0, 1)$
- $X_2 \sim Unif(1.5, 2)$
- $X_3 \sim Unif(0.5, 0.75)$

For X_2:

$M1$

0.022	1.639	0.747
0.507	1.556	0.711
0.966	1.941	0.656
0.498	1.543	0.708
0.988	1.711	0.529

$M2$

0.725	1.607	0.548
0.154	1.957	0.739
0.528	1.513	0.732
0.028	1.581	0.552
0.157	1.789	0.668

N_j

0.022	1.607	0.747
0.507	1.957	0.711
0.966	1.513	0.656
0.498	1.581	0.708
0.988	1.789	0.529

N_{-j}

0.725	1.639	0.548
0.154	1.556	0.739
0.528	1.941	0.732
0.028	1.543	0.552
0.157	1.711	0.668

Figure 6-A. An example of the construction of sample matrices M1, M2, Nj, and N-j.

6.2.2. Final Calculations

Thus, we can construct estimators for our main effect indices and total effect indices for each variable:

$$\hat{S}_j = \frac{\hat{U}_j - \hat{E}^2(Y)}{\hat{V}(Y)}$$

$$\hat{S}_{Tj} = 1 - \frac{\hat{U}_{-j} - \hat{E}^2(Y)}{\hat{V}(Y)}$$

6.3. Surrogate Models

The estimates above require some function, f, that represents the model so output values can be obtained at different sample observations. However, as outlined in Section 4.2.2, a function is not attainable for SA component models. Without a function, estimation of Sobol' indices via Monte Carlo simulation is impossible.

In the initial stages of this project, another method of sensitivity analysis was explored for the SA data. This local sensitivity analysis approach involved partial derivatives of a "measurement function" of output with respect to each of the inputs. Models were fitted to the SA data to represent the "measurement functions" in place of the SA web. The partial derivative approach was determined unreliable for the test dataset.

However, while the measurement functions' rates of change were not useful, would their predictions give us accurate enough approximations of output values for the Monte Carlo estimation process?

Models created for this purpose are called "surrogate models" and are often used when calculation of outputs is required, but the original model is inaccessible or too computationally expensive to use.

6.3.1. Details of the Models

The goal of the surrogate models created for this project is to create approximations of the black box SA models that are simple and inexpensive, yet still make accurate predictions. These "predictions" or evaluations are then used in the iterative Monte Carlo process for estimating sensitivity indices.

Six models were fit in total, with the minimum, median, and maximum of the two outputs as the response variables. All eight inputs were included with all quadratic, linear, and two-way interaction terms. An example model is shown below, where $Y_{1,min}$ represents the minimum of output 1 and X_1,X_2,...X_8 represent the inputs, and a colon represents an interaction:

$$Y_{1,min} = X_1 + X_2 + ... + X_8 + X_1^2 + X_2^2 + ... + X_8^2 + X_1{:}X_2 + X_1{:}X_3 + ... + X_7{:}X_8$$

Each model has a total of 45 coefficients: one intercept, eight linear, eight quadratic, and $\binom{8}{2}$ = (28) interactions.

Insignificant terms were not removed from the model; all inputs were included so their sensitivity indices could be calculated. Even if certain variables seemed "unimportant" through small or insignificant coefficients in the model, this was something we wanted to know and present with the tool.

6.3.2. Judging Model Predictions

These models were evaluated in a few different ways to determine if they were adequate surrogates.

The models created for Output 2 had more obviously successful predictions, so those models are discussed first. It was determined, despite initial concerns, that the predictions for Output 1 were also accurate enough for our purposes.

6.3.2.1. Output 2

All three models for Output 2 had coefficients of determination (R^2) between 0.9 and 0.92, which is very high. The coefficient of determination describes the percentage of output variance accounted for in the model. Since our sensitivity analysis techniques focus on decomposition of output variance, higher R^2 values are especially encouraging.

Several residual plots were produced during the model evaluation process. (A more complete diagnostic assessment can be found in Appendix F.) Plots of actual vs. fitted values were used as the main indicator of success, because the purpose here is to obtain accurate model outputs. The plot for the Output 2 Maximum model is shown in Figure 6-B.

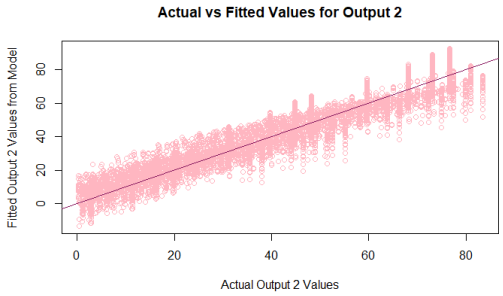

Figure 6-B. Scatter plot of actual vs. fitted values for model of Output 2 maximum against the line y = x.

The plot indicates good predictions. If the predictions were perfect, all points on the plot would fall exactly on the line y = x, shown in dark pink. Here, the fitted values are centered around the line, meaning the model predictions closely follow actual response values.

Quantitatively, the correlation coefficient between actual and fitted values can indicate this prediction strength. For this model, the correlation is 0.9509, which is very high.

Cross-validation techniques were also implemented to obtain correlations and root mean squared errors (RMSEs) on test-set predictions. For this model, we used 50-fold cross validation with 10% of the data reserved as the test set in each iteration. The 50 models' predictions for the test sets produced an average correlation of 0.9511 and an average RMSE of 6.96, subject to slight random variation. This boosts confidence that the surrogate model will perform well with unseen sample observations in the Monte Carlo process.

Minimum and median values for Output 2 performed similarly.

6.3.2.2. Output 1

Before discussing the Output 1 surrogate models, we must address a quirk of the test dataset. The minimum value of Output 1 on every output curve is zero (which, incidentally, means some value of an unseen design parameter drives Output 1 to zero).

But, in this context, this means the model for the minimum of Output 1 is a null model—there is no variance in the response to capture, every regression coefficient is zero, and key evaluation metrics, like R^2, are undefined. However, this does not detract from our analysis. If Output 1's minimum is always zero, it has no variance to decompose for VBGSA, and it's not sensitive to changes in the inputs anyway. If, upon implementation of the sensitivity analysis tool, there is a column for which all sensitivity coefficients are `NaN`, check to see whether the corresponding model is null.

The models for the median and maximum of Output 1 were non-null, but diagnostic plots presented some initial concerns. The plot in Figure 6-C seems to indicate that the model is severely overpredicting. A group of points sits far above the line y = x. Predictions were more accurate at either extreme in the data, but in general, this is not an ideal actual vs. fitted values plot.

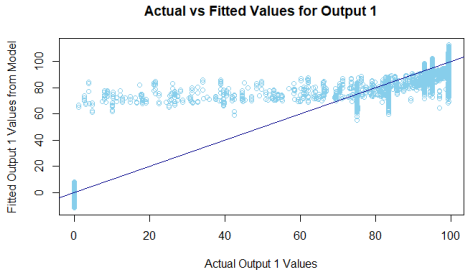

Figure 6-C. Scatter plot of actual vs. fitted values for model of Output 1 maximum against the line y = x.

However, the correlation coefficient between actual and fitted values for this model is 0.9720—higher than the correlation for Output 2. What's going on here?

Further investigation of the data found that the data is highly inflated at the extremes, meaning far more data is clustered near 0 and 100. As a result, the model will favor predictions in those spaces. In fact, only 5% of observations in the test dataset fall in the range between 5 and 70 for this response (Figure 6-).

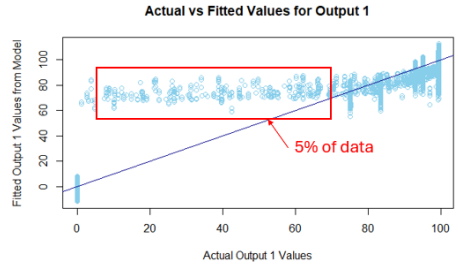

Figure 6-D. Only 5% of Output 1 values were between 5 and 70.

While the model sometimes overpredicts, most observations are appropriately fitted. The R^2 for this model is also sufficiently high at 0.9447. Cross validation techniques identical to those described for Output 2 resulted in an average correlation of 0.9718 and an average RMSE of 10.535, subject to slight random variation.

Models with transformations on the response were tested, but they did not result in significant improvements. More details on this can be found in Appendix G. Additionally, refer to Appendix F to view more diagnostic plots, evaluation metrics, and accompanying commentary for these surrogate models.

Based on these metrics, it is reasonable to proceed with these surrogate models and use them to estimate Sobol' sensitivity indices.

6.4. Results

Displaying all sensitivity indices in tables, as seen in Figure 6-D and Figure 6-E, allows for quick identification of the most important design parameters. For example, we can see in Figure 6-D that changes in Input 3 are responsible for over 79% of the variance in the minimum value of Output 2. Intuitively, results are similar for Output 2's median and maximum.

Input 1 and 2 are each responsible for around half of the variance in Output 1, so they're equally important.

Input	Out 1: Min	Out 1: Med	Out 1: Max	Out 2: Min	Out 2: Med	Out 2: Max
Input 1	NaN	0.4866	0.4786	0.0045	-0.0057	0.0022
Input 2	NaN	0.4725	0.4744	-0.0042	-0.0025	0.0131
Input 3	NaN	0.0104	0.0104	0.7925	0.8451	0.8329
Input 4	NaN	0.0100	0.0099	0.0261	0.0264	0.0258
Input 5	NaN	0.0101	0.0107	0.1123	0.0671	0.0548
Input 6	NaN	0.0104	0.0102	0.0065	0.0081	0.0083
Input 7	NaN	0.0098	0.0099	0.0213	0.0210	0.0212
Input 8	NaN	0.0100	0.0099	-0.0038	-0.0040	-0.0044
Sum	NaN	1.020	1.014	0.9552	0.9554	0.9539

Figure 6-D. One instance of main sensitivity indices calculated for the test dataset.

Input	Out 1: Min	Out 1: Med	Out 1: Max	Out 2: Min	Out 2: Med	Out 2: Max
Input 1	NaN	0.5212	0.5192	0.0133	0.0012	.0075
Input 2	NaN	0.5174	0.5265	0.0012	0.0063	0.0217
Input 3	NaN	0.0023	0.0014	0.8088	0.8607	0.8505
Input 4	NaN	-0.00003	-0.00002	0.0023	0.0024	0.0022
Input 5	NaN	0.00009	0.0002	0.1135	0.0681	0.0563
Input 6	NaN	0.0004	0.0003	0.0313	0.0308	0.0313
Input 7	NaN	0.0002	0.0002	0.0297	0.0290	0.0284
Input 8	NaN	0.0001	-0.000009	0.0118	0.0123	0.0123
Sum	NaN	1.0418	1.0476	1.0119	1.0108	1.0101

Figure 6-E. One instance of total sensitivity indices calculated for the test dataset.

Successive runs of the Monte Carlo simulation produced stable and consistent estimates for each input's sensitivity indices. Additionally, subject matter experts confirmed that the sensitivity rankings

are in line with the design features they would expect to be most and least impactful for each output. These rankings are nearly identical to those obtained with the nonstandard method (see Appendix A).

Further, these sensitivity indices followed the expected properties. No one index was estimated to be greater than 1. The sum of the main effect indices was around 1, and the sum of the total effect indices was slightly greater than 1. The total effect indices were typically greater than or equal to the corresponding main effect indices. There are slight deviations from these properties, but this is inherent to the numerical estimation process, especially since many of the sensitivity indices are so close to zero. These deviations can be mitigated by increasing the number of simulations in the Monte Carlo process.

Most important is that the tool can successfully identify major sensitivity relationships in the dataset, which it does. Minor inconsistencies in the exact index values for non-important inputs do not affect understanding of the sensitivity.

Occasionally, a sensitivity index was estimated to be negative, as seen in Figure 6-D and Figure 6-E. Again, the only negative indices were those that were extremely close to zero. This is something else that may occur during numerical estimation, and these values should be considered zero as long as zero is reasonably within the confidence interval of the estimate.

After initial success for these results and satisfaction from the rest of the design app team, estimation of Sobol' sensitivity indices with Monte Carlo simulations moved forward as the main feature of the sensitivity analysis tool.

7. IMPLEMENTATION

The product of these efforts is a sensitivity analysis tool to be used in conjunction with the component design app. After a potential dataset is curated and exported from SA, the user will run it through the sensitivity analysis tool and use the results to inform decisions about the dataset.

7.1. Main Analysis: Estimation and Visualization of Sensitivity Indices

The tool was initially created as an .RMarkdown file, which allows for combining code blocks to run the analysis and text blocks to provide instructions and explanations. Users can copy and edit the file to work with their dataset.

Additionally, the code has been translated into MATLAB by Joshua Allers (Virtual Technologies & Engineering, 0291), and it is featured as its own section of the component design app. The MATLAB version of the tool allows for certain procedures to be easily automated, including the creation of surrogate models.

7.1.1. Data Preprocessing

At the beginning of the .RMarkdown file, users load the raw dataset from SA. The code is written to work with the default structure of SA datasets, so it selects the first four columns as the output curves and selects the remaining columns as the inputs.

Recall that several rows in the dataset have the same input values, but different output values that draw out the curve. To approximate the output curve, the dataset is grouped by unique combinations of inputs. It then calculates the absolute minimum, median, and maximum of both output ranges for each input combination and stores the results as new columns. Refer to Section 4.3 for an example of output columns in the dataset.

The resulting dataset contains one row for each input setting with columns for the minimum, median, and maximum of each output for that set of inputs. The test dataset used for this project went from over 4,000,000 rows to just over 7,000 rows with eight input columns and six output columns (Figure 7-A).

input_7	input_8	output_1_min	output_1_med	output_1_max	output_2_min
100	25	0	51.11572317	92.88457124	29.10156
1000	100	0	0.04868164	0.09736328	33.00781
100	75	0	0.04868164	0.09736328	73.04688
10000	25	0	9.59028330	19.18056660	0.00000
10000	125	0	10.70996104	21.51728537	0.00000
100	75	0	0.04868164	0.09736328	25.19531
1000	25	0	60.60864319	99.21318459	0.00000
1000	50	0	0.04868164	0.09736328	33.00781

Figure 7-A. A random sample of rows from the final test dataset.

7.1.2. Estimating Indices

Next, the user defines the number of samples to be drawn for the Monte Carlo sample of the input space. Some other aspects of the code should be edited as well. For example, the tool uses R's

`lm()` function to build the surrogate models, and the formulas in the model must be changed to reflect input and output names in the user's dataset.

Efforts to fully automate model construction and other aspects of the .RMarkdown tool are currently in progress. The version of the tool within the component design app uses MATLAB's functionality to automate model construction and fixes the number of Monte Carlo simulations behind the scenes.

Functions that estimate the main effect indices and total effect indices appear next. All code is visible in the. RMarkdown tool and paired with explanations, so users can understand what's happening at every step.

The code uses the functions in loops to estimate sensitivity indices for each input related to each output.

7.1.3. Visualization of Results

Once all results are obtained, they are stored and displayed as tables presented within the .RMarkdown file (Figure 7-B). These tables are also presented with bar charts displaying the main effect and total effect indices side-by-side (Figure 7-C, Figure 7-D). The code displays 6 six bar charts total, one for the minimum, median, and maximum of each output.

Input <chr>	output_1_min <dbl>	output_1_med <dbl>	output_1_max <dbl>	output_2_min <dbl>	output_2_med <dbl>	output_2_max <dbl>
Input 1	NaN	0.01785934	0.01762375	-0.001496854	0.000476554	0.001957372
Input 2	NaN	0.01773306	0.01778187	0.113291463	0.067985520	0.057018724
Input 3	NaN	0.01791712	0.01771848	0.024047230	0.025941154	0.028014122
Input 4	NaN	0.49387814	0.48594612	0.006251230	-0.001246737	0.009027670
Input 5	NaN	0.01800773	0.01774147	0.025871083	0.028160987	0.029227981
Input 6	NaN	0.47240859	0.47466458	-0.001977594	0.001785237	0.016148529
Input 7	NaN	0.01915806	0.01880176	0.796226129	0.851672777	0.843152454
Input 8	NaN	0.01825349	0.01797048	0.008288614	0.011736643	0.014081006

Figure 7-B. A table organizing the main effect sensitivity coefficients estimated with the sensitivity analysis tool.

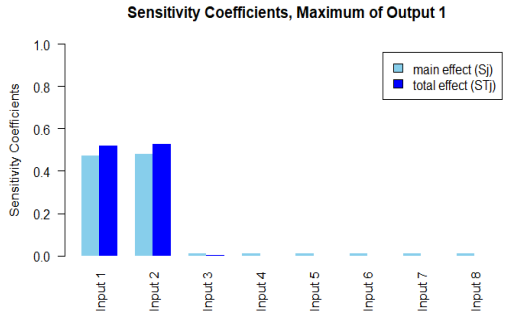

Figure 7-C. An example bar chart produced by the sensitivity analysis tool.

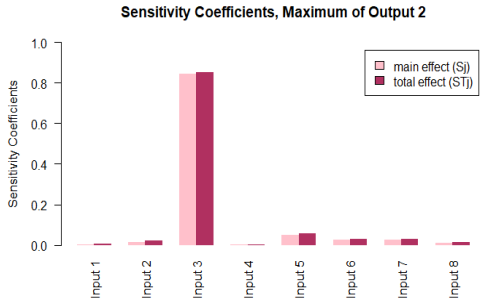

Figure 7-D. Another example bar chart produced by the sensitivity analysis tool.

7.2. Other Features of the Script

While the main feature of the tool is to calculate sensitivity indices, the script performs related analysis to further explore input-output relationships and inform choices for a better dataset.

7.2.1. Hidden Parameters

While the calculated sensitivity indices can indicate which of the selected inputs should be kept or dropped, they can't tell us whether any unselected parameters should be added as inputs.

Since the hidden parameters were not chosen for input sliders, they do not exist as variables in the dataset exported from SA. However, we can derive some indirect information about these inputs from the output columns.

Recall that the outputs appear in SA as curves representing a range of output values because of the undetermined values of the unincluded design parameters. This range is restricted based on the input settings.

Logically, if the remaining range is quite small, it can be assumed that the values of the hidden parameters don't affect the output very much. However, if the remaining range is large, it's possible that one of the hidden parameters could drive significant changes in the output.

The script uses this concept of ranges to suggest the existence of an influential hidden parameter.

Recall the structure of the grouped dataset. Each combination of input values is represented by one row in the dataset, along with the minimum, median, and maximum of each output for that setting. From this, output ranges can be calculated simply by subtracting the minimum from the maximum.

For example, say for one input setting, the maximum of Output 1 is 95 and the minimum is zero. That means there is some set of choices for the hidden parameters that gives us an Output 1 value of 95, and some other set of choices that drives it all the way to zero. This indicates that Output 1 is sensitive to hidden inputs, at least for this row (input setting).

The script finds the range for every row and then calculates the average range across all rows. Then, this average range is taken as a proportion of the total range of the output in the dataset, which is the highest maximum minus the lowest minimum. For a dataset with n rows (n different input combinations), this can be written for Output 1 as:

$$p = \frac{\frac{1}{n}\sum_{i=1}^{n} max_{Out1,i} - min_{Out1,i}}{max(max_{Out1,i}) - min(min_{Out1,i})}$$

This proportion is then judged against cutoffs that categorically indicate the likelihood that the output is sensitive to a hidden design parameter. These cutoffs are arbitrarily chosen and can be changed by the user in the MATLAB version of the component design app, or by entering different numbers in the .RMarkdown script. The results are displayed in a table (see Figure 7-E). For example, with cutoffs of 0.1 and 0.5, the tool suggests the following:

- If the output's average range is less than 10% of the total range, it is "unlikely" the output is sensitive to a hidden parameter.
- If the output's average range is between 10% and 49% of the total range, the output "may be" sensitive to a hidden parameter.
- If the output's average range is 50% or higher of the total range, the output is "highly likely" to be sensitive to a hidden parameter.

outputs <chr>	hidden_sensitivity <chr>	total_range <dbl>	average_range <dbl>
Output 1	very likely	99.40791	50.831731
Output 2	unlikely	83.39844	8.064886

Figure 7-E. An example of the table produced by the range-based hidden parameter sensitivity indicator.

7.2.2. Drop-off Tables

Now that the most important variables are identified—now that we know there are changes in certain variables that cause large changes in the output—we might want to know where in the input space those large changes are happening. This can help guide decisions for the next iteration of the dataset.

For example, if we know that changing Input 1 from 0.25 to 0.5 causes a massive change in Output 1, we might want to include more precise values on the Input 1 slider between 0.25 and 0.5.

A section of the code explores these "drop off" situations. It simply groups the dataset by different values of an input and evaluates the outputs' averages for each group. This section of the code in the .RMarkdown file leverages R's aggregate function, and the results are displayed in a table (see Figure 7-F).

input <dbl>	Output 1 Min <dbl>	Output 1 Med <dbl>	Output 1 Max <dbl>	Output 2 Min <dbl>	Output 2 Med <dbl>	Output 2 Max <dbl>
0.25	0	0.04868164	0.09736328	36.59334	36.64386	36.69438
0.50	0	28.57422534	53.12453242	28.74087	34.68668	36.59647
0.75	0	35.20844236	64.68933705	26.00736	34.18932	36.62649

3 rows

Figure 7-F. An example of a drop-off table produced by the tool.

Above, the first column represents the different values that Input 1 can take. The rest of the columns represent the average output value for all rows with each value of Input 1. Here, we can see a large drop-off in the average value of Output 1 between designs with an Input 1 of 0.25 and 0.5.

7.3. Integration with the Design App

The tool successfully completes a comprehensive sensitivity study of SA datasets. It has been tested on multiple other test datasets, with promising results so far.

Initially, the tool was developed with R, but it has since been translated to MATLAB by Joshua Allers and is featured as its own tab labeled "Sensitivity Analysis" (Figure 7-G). This will make curating new datasets more seamless, and it will also give users of the app a better idea of the relationships among parameters of the model.

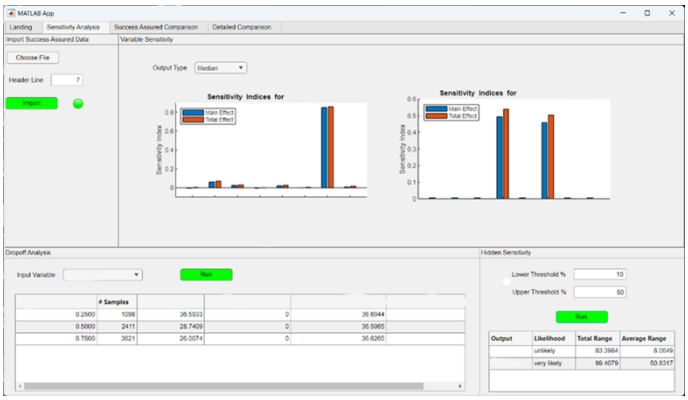

Figure 7-G. The Sensitivity Analysis tab of the component design app.

7.4. Future Work

Within the R version of the app, further automation of the script, including building surrogate models, is a priority. General improvements to the surrogate models to predict even more accurate output values is also a focus for the future. While the quirks of the surrogates, such as Output 1 overprediction, turned out not to affect the sensitivity estimates, ideally the models should behave as expected. Transformations are being explored to bind model predictions to physical realities and mitigate issues like overprediction. The hope is to improve all models while also keeping their construction as automatic as possible within the sensitivity analysis tool.

Another goal is to calculate and display confidence intervals for the sensitivity coefficients. This will make results more robust and reduce user concerns about numerical issues, like slightly negative indices or sums that deviate from Sobol' properties.

While the sensitivity analysis tool was written with the component design app in mind, it can be altered to work with other projects and other types of data. Discussions with other project groups are in the works to see how the tool can be adapted and expanded.

This page left blank

8. CONCLUSION

This new tool successfully performs sensitivity analysis with methods tailored for implementation with set-based concurrent engineering data from SA. The tool will be used to validate the dataset that was chosen for the component design app, significantly shorten the trial-and-error process for curating SA datasets, and give users of the app insight into the component design space.

As the field develops, sensitivity analysis is becoming an essential part of the modeling process (Razavi 2021). This tool provides results that are interpretable and widely applicable for model-related decision making. The code may be adapted for use with various engineering projects across Sandia.

This project is ongoing. The tool is ready for use in tandem with the design app, but it will continually be updated to improve performance and expand the analysis.

This page left blank

REFERENCES

Homma, Toshimitsu, and Andrea Saltelli. 1996. "Importance measures in global sensitivity analysis of nonlinear models." *Reliability Engineering and System Safety.*

Raudberget, Dag. 2010. "Practical Applications of Set-Based Concurrent Engineering ." *Strojniški vestnik - Journal of Mechanical Engineering.*

Razavi. 2021. "The Future of Sensitivity Analysis: An essential discipline for systems modeling and policy support." *Environmental Modelling and Software.*

Saltelli, Andrea. 2002. "Sensitivity Analysis for Importance Assessment." *Risk Analysis.*

—. 2004. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models.* John Wiley & Sons Ltd.

Sobol', I.M. 2001. "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates." *Mathematics and Computers in Simulation.*

Sobol', I.M. 1993. "Sensitivity Estimates for Nonlinear Mathematical Models." *Mathematical Modelling and Computational Experiments.*

Tang. 2006. "Comparing Sensitivity Analysis Methods to Advance Lumped Watershed Model Identification and Evaluation." *Hydrology and Earth System Sciences Discussions.*

Targeted Convergence Corporation. 2024. "Success Assured."

This page left blank

APPENDIX A. NONSTANDARD APPROACH TO VARIANCE-BASED GLOBAL SENSITIVITY ANALYSIS

While VBGSA was being explored as an option for this project, we did not immediately jump into the full Monte Carlo simulation process for estimation. Instead, a nonstandard approach was taken to calculate sensitivity indices based on the formulas described by Sobol' and Saltelli.

This method of calculating Sobol' indices was not included in the final sensitivity analysis tool. However, it was an essential steppingstone toward the best results and solidified VBGSA as the appropriate approach, so it is described here.

A.1. General Idea of the Nonstandard Approach

Recall the formulas for the main effect and total effect sensitivity indices:

$$S_j = \frac{V\left(E\left(Y \mid X_j\right)\right)}{V(Y)}$$

$$S_{Tj} = \frac{E\left(V\left(Y \mid X_{-j}\right)\right)}{V(Y)}$$

The non-standard approach literally interprets these formulas and loops through the entire dataset to calculate them. This test code was written in MATLAB. Section A.2. shows an example of the process.

For S_j, we first fix X_j at one of its values. Next, the average value for the output Y is calculated for all rows with that value of X_j; this average is stored in the first element of a vector. This is repeated for each unique value of X_j, creating a vector of average outputs. Finally, the code calculates the variance of the average output values in the vector and takes this variance as a proportion of the total output variance.

For S_{Tj}, everything is fixed except X_j. The variance of the output is taken for all rows where everything is the same except X_j, which is left to vary. This variance is stored in a vector. This is repeated for all unique combinations of the other inputs, with only X_j left to vary each time, creating a large vector of output variances across X_j. Finally, the code calculates the mean of the vector, the "average variance," and takes this as a proportion of total output variance.

This is repeated for each input variable.

This highly iterative approach is computationally possible because of the discrete nature of the data. With more detailed datasets, calculations will take much longer, which is one reason why this was not the chosen method.

A.2. Example in MATLAB

Here, an example of the calculation process is shown visually.

A.2.1. Main Effect Example – Input 4 and Output 1

Let's say that in our test dataset, Input 4, or X_4, can be either 0.25, 0.5, or 0.75. First, we fix X_4 at 0.25 by finding all rows that match that condition. All other inputs are left to vary freely.

Input 4	Input 6	Input 3	Input 5	Input 8	Output
0.2500	25	0.2500	2.5000	50	0.0974
0.2500	25	0.2500	2.5000	75	0.0974
0.2500	25	0.2500	2.5000	100	0.0974
0.2500	25	0.2500	2.5000	125	0.0974
0.2500	25	0.2500	5	50	0.0974
0.2500	25	0.2500	5	75	0.0974

Figure A-1. An example of some rows that are "fixed" at one value of Input 4. (Not all rows that match the condition are shown here.)

Next, we find the average of the output across these rows. This represents $E(Y \mid X_4 = 0.25)$. This average is stored in a vector.

We repeat the process for the next value of X_4 in the dataset, which is 0.5.

Input 4	Input 6	Input 3	Input 5	Input 8	Output
0.5000	25	0.2500	2.5000	25	92.8846
0.5000	25	0.2500	2.5000	50	95.0266
0.5000	25	0.2500	2.5000	75	95.0266
0.5000	25	0.2500	2.5000	100	95.0266
0.5000	25	0.2500	2.5000	125	95.0266
0.5000	25	0.2500	5	25	92.8846

Figure A-2. Rows that are "fixed" at the next value of Input 4.

The average output is taken again here, representing $E(Y \mid X_4 = 0.5)$. We repeat this for every value of X_4 until a complete vector of conditional expected values is formed. Here, there are only three entries in the vector because X_4 only takes three distinct values.

```
0.0974
53.1245
64.6893
```

Finally, we take the variance of this vector to obtain $V(E(Y \mid X_4))$, which is the numerator for main effect index S_4. When we divide by the total variance of output Y, the calculation is complete. Here, the total variance of the output is quite high at 2,008, and the sensitivity index is 0.5907.

Again, this same process can be repeated for all input variables.

A.2.2. Total Effect Example – Input 4 and Output 1

The process for calculating the total effect indices is based on similar logic. Consider the idea that there are multiple rows in which every design parameter is the same except Input 4. With eight inputs in the test dataset, there are many such combinations where only X_4 is left to vary.

The code first finds all unique combinations of the other seven inputs. It then takes the first combination of values and finds all rows in the dataset that match those values. These rows have everything in common except the value of X_4, and by isolating these rows, we are essentially "fixing" everything but X_4.

Input 4	Input 6	Input 8	Input 3	Output
0.2500	25	50	0.2500	0.0974
0.5000	25	50	0.2500	95.0266
0.7500	25	50	0.2500	99.2132

Figure A-3. An example set of rows where everything is "fixed" except Input 4.

Next, we find the variance of the output across the different values of X_4. This represents one instance of $V(Y \mid X_{-4})$, which is then stored in a vector.

We repeat the process for the next unique setting of all inputs except Input 4.

Input 4	Input 6	Input 8	Input 3	Output
0.2500	25	75	0.2500	0.0974
0.5000	25	75	0.2500	95.0266
0.7500	25	75	0.2500	99.4079

Figure A-4. Rows that are "fixed" at the next setting of all inputs except Input 4.

Here, the variance of the output is again calculated and stored in a vector. The code uses a loop to cycle through the dataset and calculate the variance at each setting. Because there are many different combinations of the other inputs, this vector of variances will be quite long.

```
10^3 ×
    0.0080
    3.1422
    3.1489
    3.1489
    3.1489
    0.0200
    3.1489
    3.1489
    3.1489
    3.1489
      ⋮
```

Finally, we take the average of this vector to obtain $E(V(Y \mid X_{-4}))$, which is the numerator for total effect index S_{T4}. Again, we divide by the total variance of the output Y to obtain the proportion and finish calculating the total sensitivity index. For the test dataset, this code calculated 0.9119 as the total effect index for X_4.

This same process can be repeated for all variables.

A.3. Results

Below are the results generated by this method in MATLAB for Output 1:

	variables	first_sensitivity_indices
1	Input 4	0.5907
2	Input 6	0.5629
3	Input 8	0.1307
4	Input 7	0.0001
5	Input 3	0
6	Input 5	0
7	Input 2	0
8	Input 1	0

Figure A-5. Main effect sensitivity indices for Output 1.

	variables	total_sensitivity_indices
1	Input 4	0.9119
2	Input 6	0.8949
3	Input 7	0.0753
4	Input 2	0.0663
5	Input 3	0.0519
6	Input 5	0.0322
7	Input 8	0.0314
8	Input 1	0.0145

Figure A-6. Total effect sensitivity indices for Output 1.

Below are the results for Output 2:

	variables	first_sensitivity_indices
1	Input 7	0.9209
2	Input 2	0.0692
3	Input 5	0.0471
4	Input 3	0.0218
5	Input 6	0.0052
6	Input 8	0.0033
7	Input 1	0.0004
8	Input 4	0

Figure A-7. Main effect sensitivity indices for Output 2.

	variables	total_sensitivity_indices
1	Input 7	1.1506
2	Input 2	0.3257
3	Input 3	0.1474
4	Input 6	0.1305
5	Input 5	0.0952
6	Input 8	0.0389
7	Input 1	0.0316
8	Input 4	0.0171

Figure A-8. Total effect sensitivity indices for Output 2.

At a glance, it is obvious that these results violate some properties of Sobol' sensitivity indices. For example, the sum of the main effect sensitivity indices for Output 1 is greater than 1. And while the sum of the total effect sensitivity indices can be greater than 1, it should not be the case that any single index is greater one, as we see between Input 7 and Output 2.

However, subject matter experts confirmed that the rankings of the parameters are in line with their expectations; they knew that Inputs 4 and 6 would be most important for Output 1, and that Input 7 would be most important for Output 2. Further checks with external sensitivity apps also supported the rankings.

Thus, while this calculation method had some numerical issues, the results made us confident that VBGSA methods can successfully identify the important parameters and unimportant parameters in SA datasets.

Based on these promising results, more standard and robust methods for calculating Sobol' sensitivity indices were explored. Monte Carlo simulations for estimation of these indices as formalized by Andrea Saltelli turned out to be highly successful, as described in Section 6 of this report.

This page left blank

APPENDIX B. UNDERSTANDING THE U-VALUES

B.1. U-values

The U-values are used to quantify how much our output might change if certain inputs are fixed or held steady.

Simple regression concepts provide an appropriate analogy for the U-values and how they fit into the rest of the formula. In this analogy, the numerator of each sensitivity index can be viewed as sort of "model sum of squares" or "fitted sum of squares."

$$\text{FSS} = \sum_{i=1}^{n} (\hat{y}_i - \overline{y})^2$$

Where $\hat{y}_i$ are the fitted values from the model, and $\overline{y}$ is the average of all actual responses. Here, the U-value stands as the fitted values and the estimated expectation of Y stands as $\overline{y}$.

We can also consider the overall variance in the output as the "total sum of squares":

$$\text{TSS} = \sum_{i=1}^{n} (y_i - \overline{y})^2$$

Where y_i are the actual response values.

Which makes our estimates for the sensitivity indices analogous to the coefficient of determination—also known as the proportion of variance explained by the model.

$$R^2 = \frac{FSS}{TSS}$$

Which makes sense in the context of the general idea of VBGSA—we are finding the "proportion of variance explained" by each input.

B.2. Another View

Recall the estimators for the variance of the output and the main-effect sensitivity indices:

$$\hat{V}(Y) = \frac{1}{N-1} \sum_{r=1}^{N} f^2\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) - \hat{E}^2(Y)$$

$$\hat{S}_j = \frac{\widehat{U}_j - \hat{E}^2(Y)}{\hat{V}(Y)}$$

And consider the formula for our U-value estimate:

$$\widehat{U}_j = \frac{1}{N-1} \sum_{r=1}^{N} f\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) \odot f\left(x_1^{(r')}, x_2^{(r')}, \ldots, x_j^{(r)}, \ldots, x_k^{(r')}\right)$$

The first term of the summand is a model evaluation at a sample observation from M1. Let us call this f_o. The second term is a model evaluation where all input values have been changed except the values of X_j. Let us call this f_j.

If f_j takes the same value as f_o, we know that changing the other variables had no effect on the model output; in other words, the output value is completely dominated by the value of X_j. This indicates high sensitivity—100% sensitivity, in fact.

In this case, we can say that $\widehat{U_j}$ is equivalent to:

$$\widehat{U_j} = \frac{1}{N-1}\sum_{r=1}^{N} f^2\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right)$$

And plugging this into the estimator for $\widehat{S_j}$, we have:

$$\hat{S}_j = \frac{\frac{1}{N-1}\sum_{r=1}^{N} f^2\left(x_1^{(r)}, x_2^{(r)}, \ldots, x_k^{(r)}\right) - \hat{E}^2(Y)}{\hat{V}(Y)}$$

Notice that now, the numerator for $\widehat{S_j}$ is identical to the estimator for $V(Y)$, so we have:

$$\hat{S}_j = \frac{\hat{V}(Y)}{\hat{V}(Y)} = 1$$

The sensitivity index becomes 1, indicating that changes in input X_j are responsible for 100% of the changes in Y. Input X_j gives Y all its variance. This is what we expected, because changing the other input variables had no impact on Y (as determined by f).

The same logic can be applied to the total effect index, which is constructed with:

$$\hat{S}_{Tj} = 1 - \frac{\hat{U}_{-j} - \hat{E}^2(Y)}{\hat{V}(Y)}$$

The U-value for the total effect indices is:

$$\hat{U}_{-j} = \frac{1}{\mathrm{N}-1}\sum_{\mathrm{r}=1}^{\mathrm{N}} \mathrm{f}\left(\mathrm{x}_1^{(\mathrm{r})}, \mathrm{x}_2^{(\mathrm{r})}, \ldots, \mathrm{x}_\mathrm{k}^{(\mathrm{r})}\right) \odot \mathrm{f}\left(\mathrm{x}_1^{(\mathrm{r})}, \mathrm{x}_2^{(\mathrm{r})}, \ldots, \mathrm{x}_\mathrm{j}^{(\mathrm{r}')}, \ldots, \mathrm{x}_\mathrm{k}^{(\mathrm{r})}\right)$$

Here, let us call the first model evaluation term f_o and the second f_{-j}. For f_{-j}, all variables have been fixed except X_j, which has been changed. If f_{-j} is equivalent to f_o, that means that changing X_j had no effect on the model output—there is zero sensitivity to that variable.

In this case, we can say that the U-value is equivalent to:

$$\hat{U}_{-j} = \frac{1}{\mathrm{N}-1}\sum_{\mathrm{r}=1}^{\mathrm{N}} \mathrm{f}^2\left(\mathrm{x}_1^{(\mathrm{r})}, \mathrm{x}_2^{(\mathrm{r})}, \ldots, \mathrm{x}_\mathrm{k}^{(\mathrm{r})}\right)$$

And plugging this into the estimator for $\hat{S}_{Tj}$, we have:

$$\hat{S}_{Tj} = 1 - \frac{\frac{1}{\mathrm{N}-1}\sum_{\mathrm{r}=1}^{\mathrm{N}} \mathrm{f}^2\left(\mathrm{x}_1^{(\mathrm{r})}, \mathrm{x}_2^{(\mathrm{r})}, \ldots, \mathrm{x}_\mathrm{k}^{(\mathrm{r})}\right) - \hat{E}^2(Y)}{\hat{V}(Y)}$$

Again, the numerator in the fractional component becomes identical to $\hat{V}(Y)$, so we have:

$$\hat{S}_{Tj} = 1 - \frac{\hat{V}(Y)}{\hat{V}(Y)} = 1 - 1 = 0$$

The sensitivity index becomes 0, meaning that changes in input X_j causes 0% of the changes in Y. Input X_j does not contribute to $V(Y)$, which is again what we expected.

This page left blank

APPENDIX C. EXISTING TOOLS FOR SENSITIVITY ANALYSIS AND WHY WE DEVELOPED OUR OWN

As sensitivity analysis becomes more common in modeling work, some tools have been created to make sensitivity analysis easier for users of popular software such as R, MATLAB, and Python.

However, these tools were not implemented in the script that serves as the main feature of this project. Trials found them to be inflexible and generally difficult to interpret. Success Assured (SA) data is uniquely structured and presents several challenges to analysis. The tool uses R and MATLAB code to calculate Sobol' indices for the data "by hand," without any pre-existing sensitivity analysis packages.

This allowed for full control over the process—basic VBGSA methods were necessarily altered to work with the peculiarity of SA data and produce mathematically sound results. Using pre-written apps or packages would have eliminated flexibility and potentially jeopardized the accuracy of results.

In turn, engineers using the tool can read the code in the .RMarkdown file and understand exactly what is happening as their dataset undergoes analysis. Interpretability is important at every step in this design app project, including the initial dataset selection stages. Additionally, members of the team can continue to tweak or add to the code as needs of the project evolve and new ideas form. These new ideas wouldn't be possible if the project was constrained to the functionality of someone else's sensitivity tool.

This page left blank

APPENDIX D. VALIDATION OF METHODS

Different sensitivity methods use vastly different metrics to compare variables. Sensitivity coefficients and importance rankings depend on choice of method. Because of this, there's not necessarily a "right" or "wrong" answer when determining which inputs are most important to the outputs. However, it's also true that certain methods will not provide helpful insight.

So how did we determine the relative success of different approaches attempted for this project? Several indicators solidified our confidence in Monte Carlo simulation for estimation of Sobol' sensitivity indices.

D.1. Subject Matter Experts

Subject matter experts (SMEs) were consulted throughout the project. Each test dataset represents the design space of a real component. Sandia SMEs with advanced knowledge of the component and its digital Success Assured (SA) model were able to judge the inputs' calculated sensitivity scores based on which features they knew to be most important in determining the output. For all test datasets, SMEs determined that the estimated Sobol' sensitivity indices matched their expectations.

D.2. Cross-Referencing with Existing Tools

Second, tools that already exist with popular software to perform sensitivity analysis were explored and used as reality checks against our own results. Sensitivity descriptions obtained with MATLAB's Sensitivity Analyzer app supported both SME testimony and the results produced by our tool. More information about existing sensitivity analysis tools and why they were not otherwise relied on for this project can be found in Appendix C.

D.3. Consistency

Third, consistency within and across methods boosted confidence that the method developed by Sobol' and Saltelli accurately reflected the model's input-output relationships. Separate runs of the Monte Carlo simulation and estimation produced sensitivity estimates that did not vary beyond the small amount expected when performing numerical estimation.

Additionally, while the numerical values for sensitivity indices differed depending on the method used, the ordering of the inputs from most to least sensitive was similar. Different methods identified the same "most important" inputs.

This page left blank

APPENDIX E. DEATH BY DIAGNOSTICS

This section contains more diagnostic plots and evaluation metrics for the surrogate models. Diagnostic plots focus mainly on the maximum of the outputs; diagnostic trends were similar across minimum, median, and maximum.

Again, diagnostics for Output 2 will be shown first, as interpretation of Output 1 diagnostics is easier with this context.

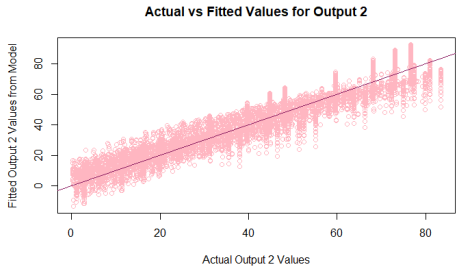

Figure B-1. Actual vs. fitted values for Output 2, shown again for reference.

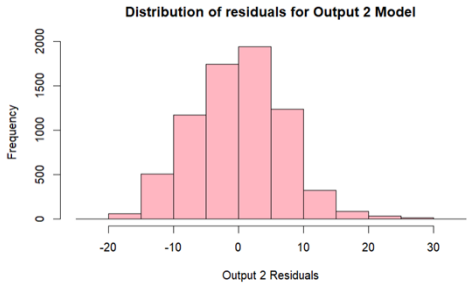

Figure B-2. Residuals are roughly normally distributed.

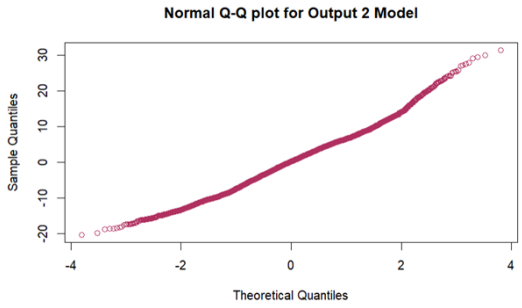

Figure B-3. Q-Q Plot shows a roughly straight line.

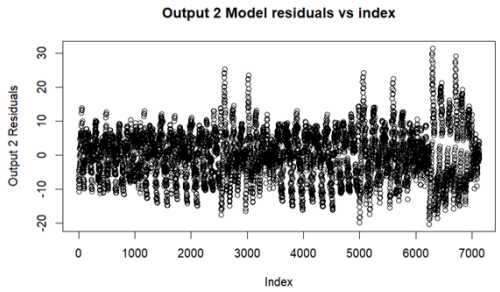

Figure B-4. Residuals are centered at zero. Most are quite small, with some larger values.

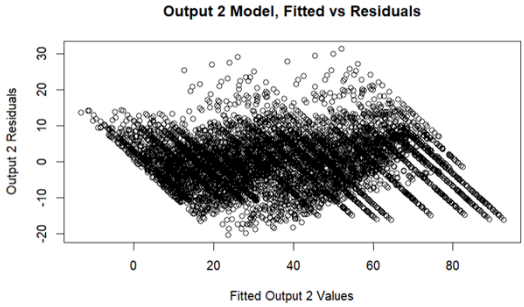

Figure B-5. No alarming trend appears in the fitted-residuals plot; residuals are centered at zero. Diagonal lines appear because simulated Output 2 values are semi-continuous.

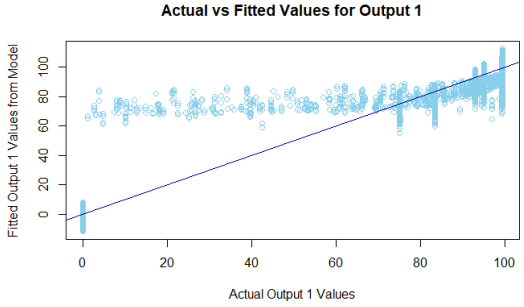

Figure B-6. Actual vs. fitted values for Output 1, shown again for reference. Recall that just 5% of Output 1 values fall between 5 and 70.

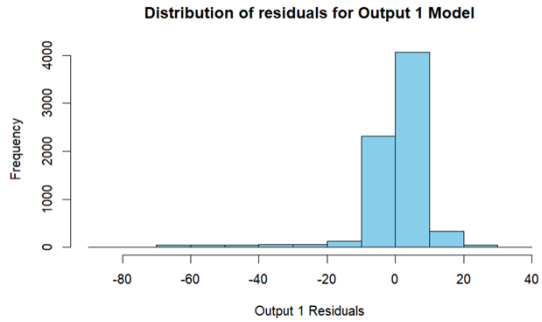

Figure B-7. Distribution of residuals is not immediately clear; skewed by overpredictions on the troublesome 5% of data.

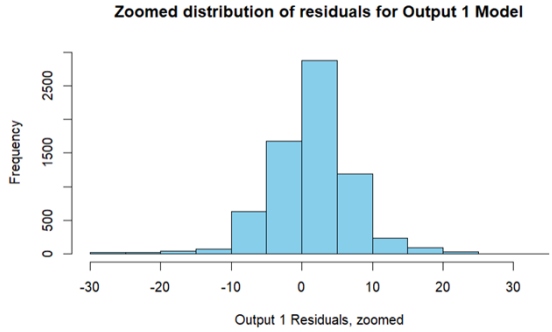

Figure B-8. However, if we get rid of residuals less than -30, we see most residuals are roughly normally distributed.

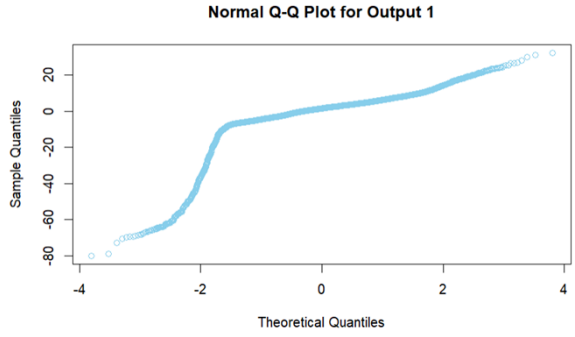

Figure B-9. The strange curve on the left side of this Q-Q plot is common with truncated data, and we have a minimum of 0.

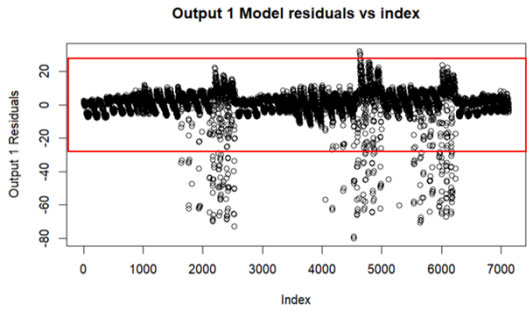

Figure B-10. Ignoring that troublesome 5% of data, we see the residuals are tightly centered at zero.

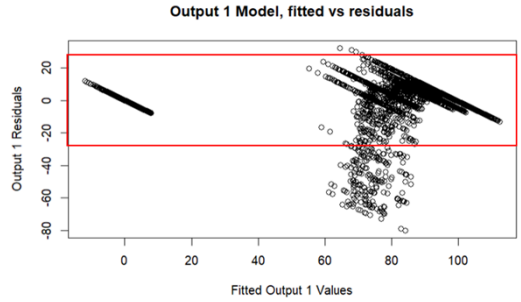

Figure B-11. Again, ignoring that 5%, we see a reasonable fitted vs. residual plot – centered at 0 with no clear trends. Diagonal lines appear because simulated Output 1 values are semi-continuous.

Recall that the model for the minimum of Output 1 is null. Correlations between actual and fitted values for all six models are shown below:

Table E-1. Correlations for all six surrogate models.

	Minimum	Median	Maximum
Output 1	N/A	0.9675	0.9720
Output 2	0.9554	0.9550	0.9509

And here are the R^2 values for all six surrogate models:

Table E-2. Coefficients of determination for all six surrogate models.

	Minimum	Median	Maximum
Output 1	N/A	0.9360	0.9447
Output 2	0.9128	0.9121	0.9041

APPENDIX F. TRANSFORMATIONS FOR SURROGATE MODELS

As seen above, diagnostic plots for the surrogate models weren't terrible, but there's room for improvement, especially with the Output 1 models. Various transformations and alternative models were tested on the outputs to see if this would fix the overprediction issue, obtain more accurate final sensitivity estimates, and generally improve prediction strength.

F.1. Arcsine Transformation on Output 1

Since the range of Output 1 falls between 0 and 100—in the test dataset, it is a proportion—the first alternative model included an arcsine transformation on the output. This is the typical transformation used for proportion data.

$$sin^{-1}(\sqrt{(Output_1/100)}) = X_1 + X_2 + \ldots + X_8 + X_1^2 + X_2^2 + \ldots + X_8^2 + X_1{:}X_2 + X_1{:}X_3 + \ldots + X_7{:}X_8$$

This successfully constrained the predicted values to the expected range between 0 and 100, but it did not solve the overprediction issue seen with the original model. Correlation between actual and fitted Output 1 (maximum) values only improved by around 0.001, to 0.973.

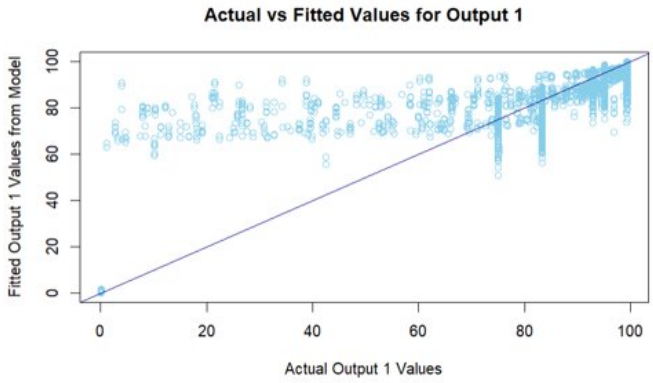

Figure F-1. Scatterplot of actual vs. fitted values for Output 1 with arcsine transformation.

Additionally, estimates for the sensitivity indices were not significantly different when we implemented the transformed surrogate model; any differences were not distinguishable from the slight variation inherent to the numerical estimation process.

F.2. Square root transformation on Output 2

A square root transformation was attempted on Output 2, which must be positive.

$$\sqrt{Output_2} = X_1 + X_2 + \ldots + X_8 + X_1^2 + X_2^2 + \ldots + X_8^2 + X_1{:}X_2 + X_1{:}X_3 + \ldots + X_7{:}X_8$$

As expected, this transformation kept Output 2 predictions above 0. However, this slightly decreased the correlation between actual and fitted Output 2 (maximum) values, to 0.9407.

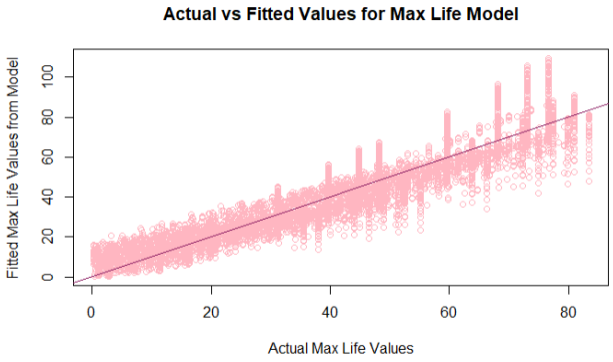

Figure F-2. Scatterplot of actual vs. fitted values for Output 2 with square root transformation.

Again, there were no noticeable differences between sensitivity estimates obtained with the original model and with the alternative model.

F.3. Results

Ultimately, the goal of this project is to understand which model inputs are most important in determining model outputs. The tool successfully achieves this goal by performing sensitivity analysis on Success Assured datasets, and this success was neither improved nor hindered by adding transformations to the surrogate models.

These transformations did not change sensitivity estimates or importance rankings, so they did nothing to improve our understanding of relationships within the dataset. There were additional concerns regarding the ability to automate the construction of alternative models based on diagnostic results and output properties.

For these reasons, the effort was ceased, and estimation continued with the original simple linear surrogate models.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Kate N. Gabet Hoffmeister	07585	kngabet@sandia.gov
Taylor Hall	07585	thhall@sandia.gov
Joshua Allers	00291	jpaller@sandia.gov
Brendon Mizener	01545	bjmizen@sandia.gov
Talia Duffy	07585	tgduffy@sandia.gov
Technical Library	1911	sanddocs@sandia.gov

This page left blank

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.