# A Hardware Model Validation Tool for Use in Complex Systems

Misty D. Davies* and Karen Gundy-Burlet†
*NASA Ames Research Center, Moffett Field, CA, 94035*

*and*

Greg Limes‡
*CraigTech Industries, Tallahassee, FL, 32311*

**One of the many technological hurdles that must be overcome in future missions is the challenge of validating as-built systems against the models used for design. We propose a technique composed of intelligent parameter exploration in concert with automated failure analysis as a scalable method for the validation of complex systems. The technique is impervious to discontinuities and linear dependencies in the data, and can handle dimensionalities consisting of hundreds of variables over tens of thousands of experiments.**

## I. Introduction

Increasing capabilities for both vehicle and ground systems comes at the cost of ever-growing complexity and highly interdependent subsystem components. Design of one subsystem component is often achieved by modeling the other components, with the final integration only happening on the doorstep of the final implementation. We handle much of the complexity by taking advantage of the flexibility inherent in software engineering. As a result, software is integrated with almost every other component of the system and depends on reliable models of sensors, actuators, processors, communications, and storage in order to be robust. Validating the models against the as-built hardware in the system is difficult, and mismatches have led to space system failures. As one example, the 1997 Lewis satellite failed because it could not maintain its Sun-pointing mode.[1] This mode had been tested using a thruster simulation that assumed the thrust vector would be perfect. It is likely that the final implementation of the thrust system was well within the tolerance specified by the requirements; however, because this tolerance was not accounted for in the thruster model the fault in the flight software was impossible to find using simulation testing.

In order to avoid underspecification in system models, Jaffe et. al. [2,3] have defined criteria for requirements that include assignments for many properties of system interactions, inputs, and outputs. The primary goal of these criteria is to formalize requirements generation and to define

* Research Computer Engineer, Intelligent Systems Division, Mail Stop 269-3, AIAA Member.
† Research Scientist, Intelligent Systems Division, Mail Stop 269-3, AIAA Associate Fellow.
‡ Senior Engineer, Intelligent Systems Division, 4836 Lake Park Dr., Talahasee, FL.

a measure of requirement completeness. However, in systems that have models built according to these criteria, we have an added benefit: these models allow us to determine the margins to failure in the interplay between the software and hardware components of the system.

The Margins Analysis[3-5], as shown in Figure 1, is a multi-step Monte Carlo Filtering technique[6,7] developed by the Robust Software Engineering (RSE) group within the Intelligent Systems Division at NASA Ames Research Center. The tool is used for beginning-to-end directed testing, with both the software and simulations of the hardware modeled together as the system-under-test. We then generate tests well into the off-nominal (but still possible) flight envelopes. By assuming that most hazards are triggered by a maximum combination of two or three input variables we limit our number of test vectors while still obtaining a parametric coverage guarantee.[8] This assumption allows us to analyze hundreds of input and output parameters over tens of thousands of experiments.

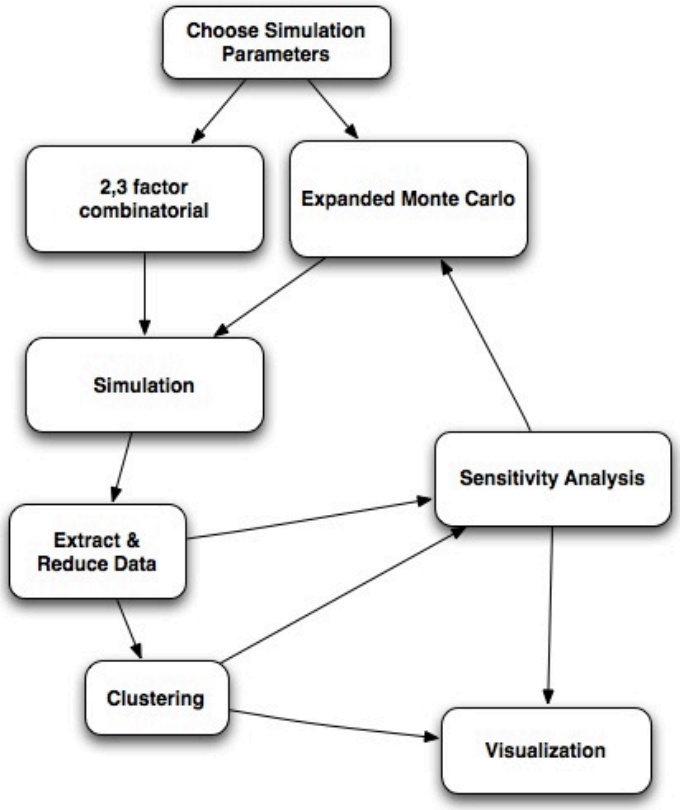


**Figure 1: A flowchart for the Margins Analysis process.**

The system requirements are used to create penalty functions. Each run is graded according to the penalty function, and new tests are generated in order to fully explore the failure boundaries. We use machine learning techniques[9-11] in order to both identify suspicious inputs (and their critical ranges) and to find unexpected structure. By driving the failure of requirements in an automated loop, we can collect enough examples to determine which inputs demonstrate the most sensitivity with respect to a particular undesired behavior. Defects within flight software or within the system model can usually be found on the narrowed path between the implicated inputs and the failed outputs.

In order to validate hardware models, we propose a multi-pronged approach. Prior to the delivery of hardware, we expand the hardware parameters in the model far beyond the nominal ranges of the system in order to find the margins to failure. This is possible (and simple to automate) when the hardware models include the details Jaffe et.al.[2,3] outline as being germane to requirements specification. Our analysis at this stage gives us a sense of how robust our system is within the assumptions inherent to the model. Assuming that our model is reasonably correct, it should also give us the input parameters that the hardware behavior is most sensitive to. We can use this information to plan tests on the final hardware.

After the hardware is delivered (or in the case that we have multiple fidelities of models below the actual hardware) we can use the analysis to validate that the abstraction assumptions used at the lower fidelities don't lead to behavior significantly different from the behavior of the final system as implemented. In this case, the penalty method for the analysis is based on differences between the lower-fidelity and higher-fidelity systems. The sensitivity analysis will then illuminate the key input parameters related to the mistaken assumptions.

## II. Methodology

The most important step required for the validation of models is an understanding of the limitations and assumptions inherent in those models. This step is non-trivial and requires human expertise. Unfortunately, a methodology for how to perform this step is beyond the scope of this paper. Interested readers should look to the work of Jaffe, et. al. [2,3], Nancy Leveson[12], and to the NASA Modeling and Simulation Standard.[13] For the purposes of this paper, what is important to note is that it is impossible to measure effects that have not been instrumented. In the case of clarifications like the expected (and possible) ranges on sensor inputs it is possible to design a system that will remind the engineer to complete specifications. However, other modeling assumptions (e.g. the linearity of springs), are harder to automatically detect. *This point cannot be stressed highly enough: if assumptions on the design are not documented, it is (at least) difficult or (likely) impossible to determine whether those assumptions will affect the behavior of the overall system.*

Instrumenting the model with the appropriate assumptions at design time allows for experiments that determine how appropriate those assumptions are. The first and most obvious step is to perform a sensitivity analysis of the hardware design parameters. Spring constants, friction, pointing errors, etc. will all have an expected set of values and a larger possible set of values. Running sensitivity analyses over the possible range of values allows the engineers and designers to know how close to failure they are, and which parameters are most critical to control.

Tests of complex systems provide high dimensionalities of data. This data is rarely independent, and is usually a mixture of continuous, periodic, and discrete distributions. As a result, sensitivity analysis methods based on derivative techniques become trapped in the wealth of local minima. Pattern-detecting algorithms like clustering assume that the data can be described as mixtures of Gaussians. Rule-based learners like the one we are using for our sensitivity analysis often assume linear independence of the outputs.

To combat these problems, we've implemented a number of data transformations within the Margins Analysis. We automatically sort the data to find delta functions and periodicities. A peak occurring at a period boundary is handled by moving the boundaries. The unsupervised clustering technique within the Margins Analysis utilizes an expectation-maximization algorithm that has numerical difficulties at high dimensionality. As a consequence, we use a principal component analysis to reduce the dataset. In practice, we've found that more than 99% of the information is contained within fewer than half of the dimensions.

We use a supervised treatment learning algorithm[14] for the sensitivity analysis within the Margins Analysis. Treatment learning is a sampling technique and is unfazed by high dimensionality or by discontinuities. The treatment learning algorithm's Achilles heel is the assumption that all of the parameters are independent. When this assumption is false the treatment learner misses important sensitivities. For this step in the technique, we perform an

analysis over the data simultaneously in the original space and in the rotated space of the principle component analysis. We then project the results back into the original variable space, so that they can be visualized and understood by domain experts[15].

Knowing the areas of greatest sensitivity in a lower-fidelity model allows a test designer to intelligently plan tests on higher-fidelity models (including the actual hardware itself). Making sure that ranges in which the greatest change is expected are adequately covered provides for stronger statistical confidence that the data obtained from low-fidelity simulation and higher-fidelity testing are actually from the same distribution.

The fact that the sensitivity analysis we are using here is impervious to discrete and continuous variables allows for another layer of validation when models of differing fidelity exist and the data obtained from bisimulation of those models is shown to differ. The sensitivity analysis can be tuned to discover the likely root causes of the differing data—giving the test engineer valuable information about which parts of the model or which hardware component are most in need of extra analysis and possible change.

## III. Test Examples

As a proof of concept, we are using three different test examples. All of the test examples have been coded in MATLAB's Simulink modeling language. The first example is of a simple spring-mass-damper system with a constant force applied, as shown in Figure 2. The Simulink model for the simple system is shown in Figure 3. All of the nominal values are shown in Figure 3. Integrations are performed using a continuous Runge-Kutta scheme with a variable timestep and the spring is assumed to be a linear spring. For our tests, we dispersed the mass of the vehicle (m1) between 1 and 3 kg, the gain on the damper (c) between 150 and 500 N*sec/meter, the spring equilibrium distance (L0) between 0.03 and 0.09 meters, the constant force (f) between 0 and 16 N, the linear spring constant (k) between 200 and 400 N/meter, the initial position of the vehicle (x1(0)) between -0.1 and 0.1 meters, and the initial velocity of the vehicle (x1_dot(0)) between -0.1 and 0.1 meters/sec.

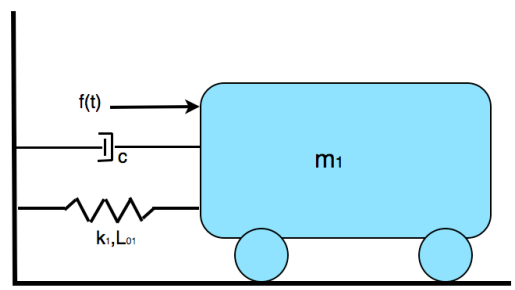


**Figure 2: A simple, second-order, spring-mass-damper system.**

As a slightly more complicated test example we also implemented the system shown in Figure 4. This system has two interacting masses, with three springs, two dampers and two forces. The Simulink model for the simple system is shown in Figure 5, along with all of the nominal values for the system. As in the simple, second-order system, integrations are performed using a continuous Runge-Kutta scheme with a variable timestep and all of the springs are assumed to be linear springs. For this test case, we dispersed the width of the vehicles (d1 and d2), the total distance (d3), the vehicle masses (m1 and m2), the equilibrium spring positions (L01, L02, and L03), and the initial vehicle positions (x1(0) and x2(0)) to +/- 10% of their nominal values. The rationale for the dispersion range for these variables is that these values were the most likely to be easily and accurately measured, so the dispersions could be narrower. The

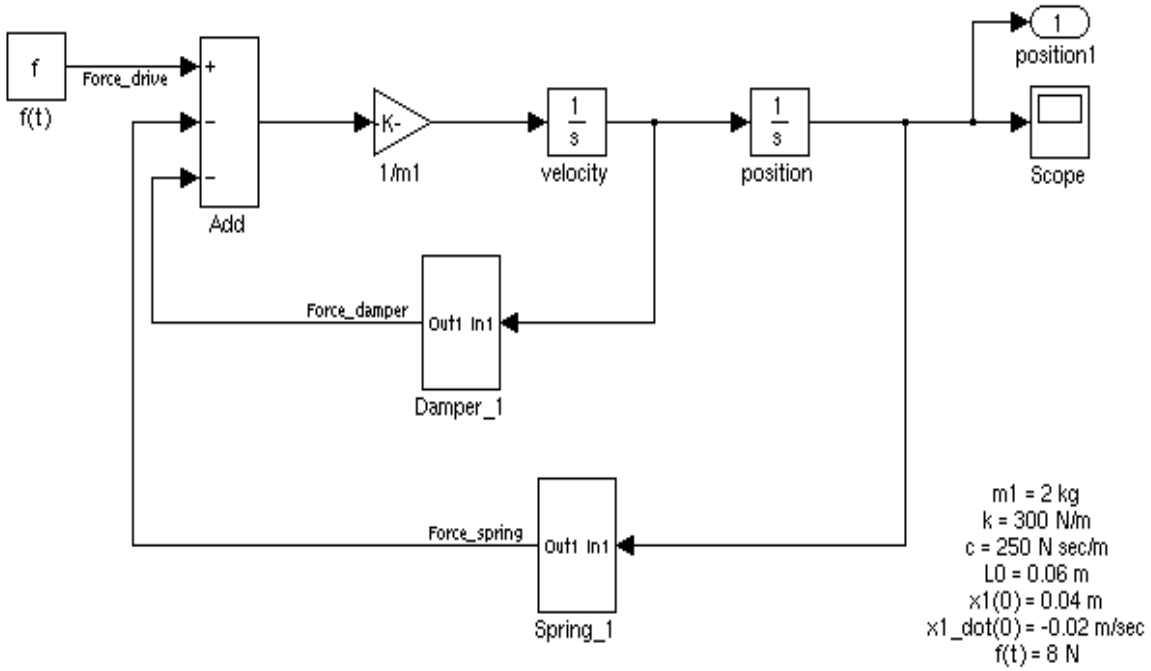


**Figure 3: The resulting Simulink model for the system shown in Figure 2.**

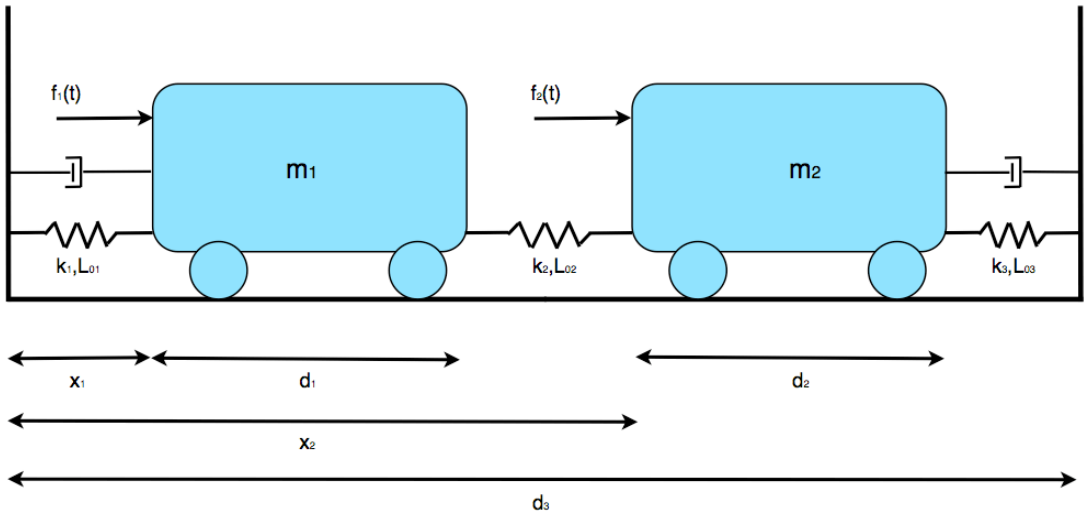


**Figure 4: A more complicated spring-mass-damper system. This system has three springs, two masses, two dampers, and two independent forces.**

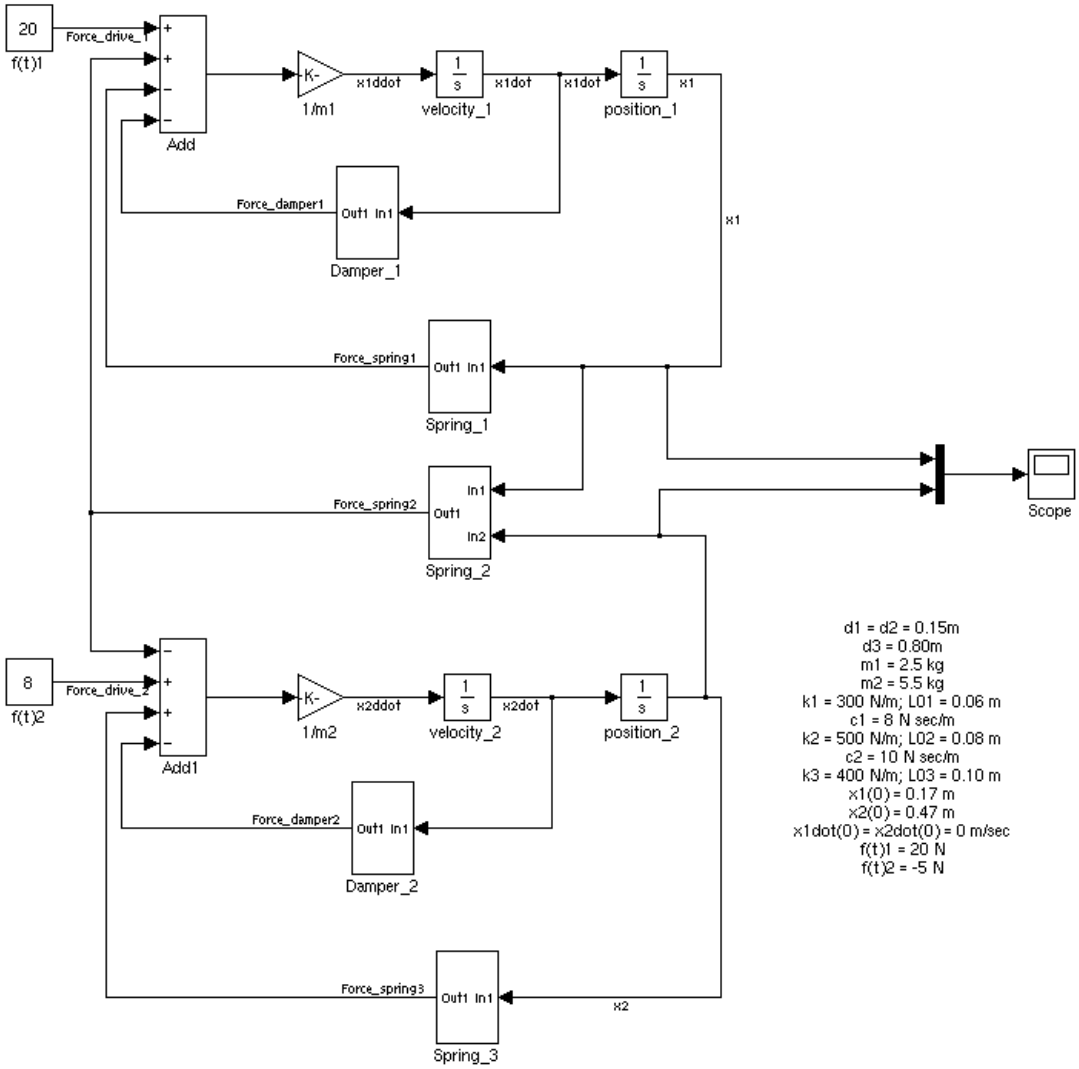


**Figure 5: The Simulink model for the system shown in Figure 4**

first spring constant (k1) was dispersed between 150 and 600 N/meter, the second spring constant (k2) was dispersed between 250 and 500 N/meter, the third spring constant (k3) was dispersed between 200 and 800 N/meter, the first damper constant (c1) was dispersed between 4 and 16 N*sec/meter, the second damper constant (c2) was dispersed between 5 and 20 N*sec/meter, the first force (f1) was dispersed between -20 and 40 N, and the second force (f2) was dispersed between -10 and 5 N. The initial velocities (x1_dot(0) and x2_dot(0)) were dispersed between -0.1 and 0.1 meters/sec. These wider dispersions were chosen to capture the greater amount of uncertainty on these values and the possible effects of unmodeled dynamics like unexpected friction forces. At this point, we note that there are still uninstrumented sources of error in the model—for example, the springs are still modeled as linear springs. Without

specifically instrumenting this assumption and providing a comparison (like a higher-fidelity spring model), it is impossible to be sure that this is a reasonable assumption for our system.

The final test example for this work is a gravity model based on JPL's Lunar Constants and Models document.[14] This model takes the gravitational "lumpiness" of the body into account by creating a model based on spherical harmonics. This particular application uses the gravitational constants for the Moon, Earth, and Mars, along with the diameter of the Moon, all of which have some uncertainty in measurement. For this paper, we've assumed that the last two significant digits for each of these parameters have some uncertainty. The gravity model can be tuned to fidelities between 1 (for the coarsest level model) to over 75. At the highest orders, computational errors remove any quantitative difference between the fidelities of adjacent orders, as shown in Figure 6. The recommended fidelity for this model is of the order of 50. However, increasing fidelities also greatly increase computational cost per point location. For a recent example, illustrated in Figure 7, the coarsest model took 10.19 ms per point location, order 40 took 28.7 ms per point location, and order 75 took 234 ms per point location. This model is intended to run as the environment simulator for processor-in-the-loop testing. In order to return results quickly enough for the processor to process them, it will likely be necessary to run

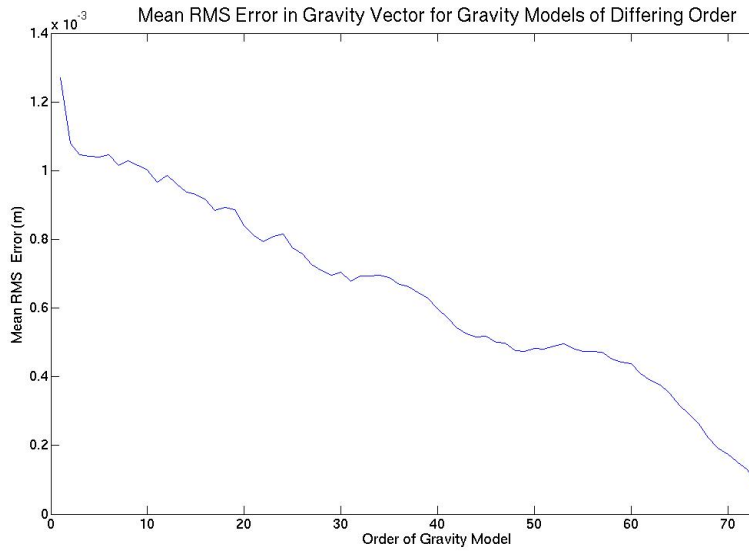


**Figure 6: Error in the gravity model for differing orders of fidelity.**

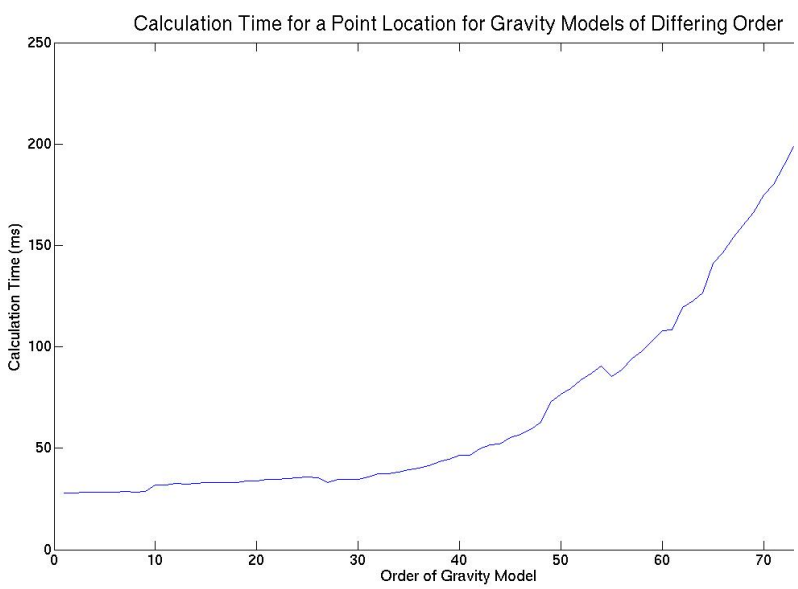


**Figure 7: Calculation time for a point location given differing orders of fidelity within the gravity model.**

at an order below the highest fidelity. An important question to ask is which parameters are most likely to be affected by this cost-performance trade.

For all of our test examples, we assume that the data obtained using the nominal values for the highest-fidelity model are the 'true' data, and we compare the differences between the 'true' results and the results obtained when the design values are dispersed. For the spring-mass-damper systems, the critical values are the positions of the masses at every timestep. For the gravity model, the critical values are the values of the gravity vector at a 40 point location subset randomly selected from the 32,000 point locations available in the model.

## IV. Results

### *1. Simple Second Order Spring Model*

The simple second order spring-mass-damper model shown in Figures 2 and 3 was intended to be an easily understood benchmark for the sensitivity analysis shown here. The system-level requirement of interest for this for this test case was the final equilibrium position of the vehicle, with the equilibrium position of the nominal case considered to be the ideal. Two sensitivity analyses were run: the first asked the analysis to find the input variables associated with position values near the ideal, the second asked the analysis to find the input variables associated with position values farthest from the ideal. The analysis for each case automatically generated 10 solutions. Every solution implicated a combination of the input force (f) and spring equilibrium value (L0) as being the most critical to either success or failure. An example is shown in Figure 8. The blue data points represent values close to the ideal while the red data points represent values from the ideal. What is immediately apparent from the plot is that the most critical relationship is the linear relationship between the two values. Sensitivity results from the "best" analysis chose ranges outlining the blue points across the diagonal of the plot, while sensitivity results from the "worst" analysis chose ranges outlining the red points at the top

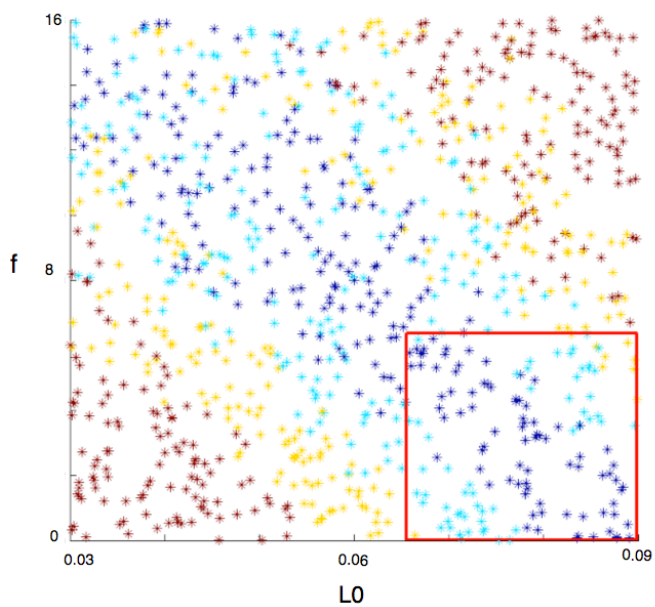


**Figure 8: A figure showing the highest-ranked solution in the sensitivity analysis results.** *Each data point represents a different simulation trial, and the color of the data point represents the difference between the desired and actual vehicle equilibrium positions, with blue being most ideal and red farthest from the ideal. This plot highlights the clear linear relationship between the force and the spring equilibrium position.*

right and bottom left corners. A sensitivity analysis that included PCA data would not have been limited to results along the variable axes, and may have been able to select the entire linear range; this analysis will be performed in future work. The analysis performed here is completely automated and is capable of finding correlations for more complicated systems, as shown in the next two test examples.

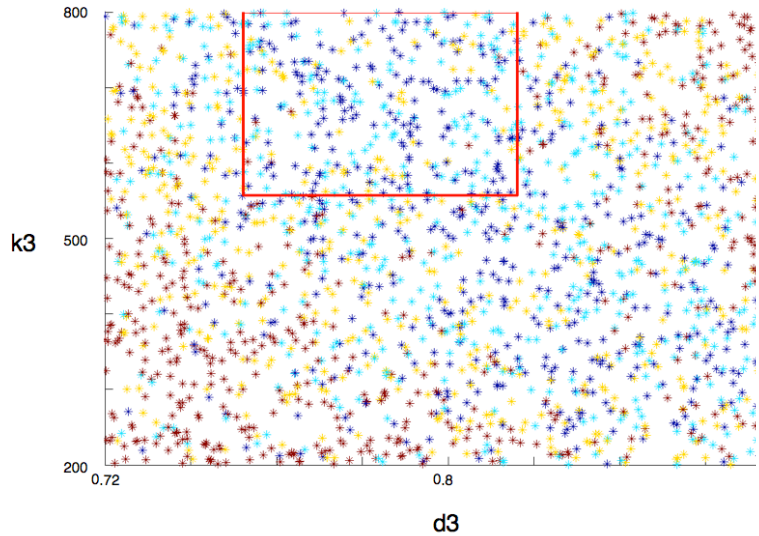


**Figure 9: A figure showing the relationship between k3 and d3 for the two mass spring system.** *Blue data points were runs that had final positions near the ideal, while red data points involved final positions farthest from the ideal.*

*2. Two Mass Spring Model*

The two mass spring-damper system shown in Figures 4 and 5 is arguably more difficult to analyze using a traditional sensitivity analysis since the differential equations describing the system are much more complicated and are interrelated. The Margins Analysis simply looks for relationships within the output data, without regard for the form of the equations. The system-level requirement of interest for this for this test case was the final equilibrium position of both of the vehicle, with the equilibrium positions of the nominal case considered to be the ideal. Two sensitivity analyses were run: the first asked the analysis to find the input variables associated with position values near the ideal, the second asked the analysis to find the input variables associated with position values farthest from the ideal. The analysis for each case automatically generated 10

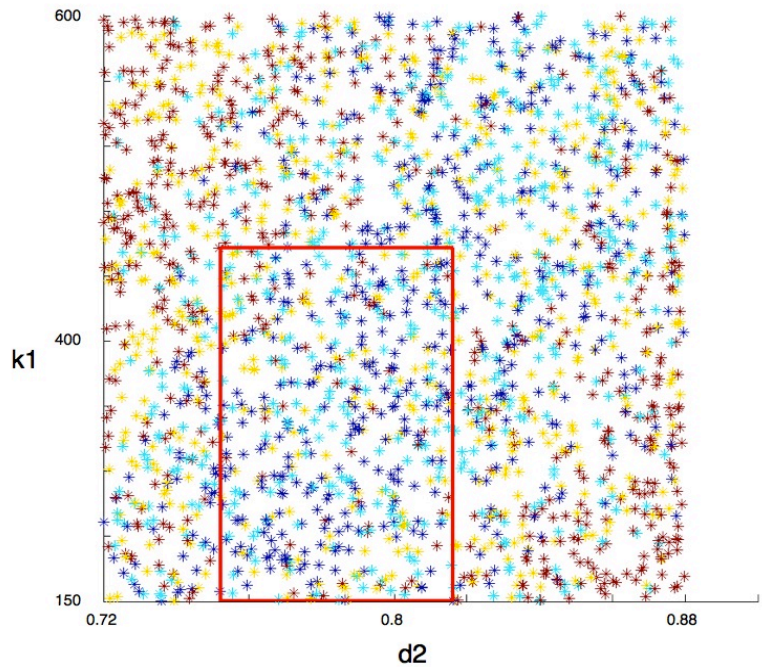


**Figure 10: A figure showing the relationship between k1 and d2 for the two mass spring system.** *Blue data points were runs that had final positions near the ideal, while red data points involved final positions farthest from the ideal.*

solutions. For this case, all 20 solutions involved 4 variables: k1, k3, d2, and d3. Example components for these solutions are shown in Figures 9 and 10. Both Figures 9 and 10 were selected from the sensitivity analysis that was looking for relationships explaining the "best" data. The red boxes are regions that the analysis chose as having a high number of "good" points and a small number of "bad" points. In Figure 8, we see that there is an inverse linear relationship between k3 and d3, and that the best data lies on a diagonal in the center of the plot. The worst data lies in the upper right and lower left corners of the plot. Figure 9 shows a linear correlation between k1 and d2 (similar plots show the relationship between k1 and d3). The sensitivity analysis selecting for behavior farthest from the ideal selected the same plots, but highlighted the data in the corners were the majority of the data points are red. An analyst can quickly see from these plots that the important relationships for tolerance control will be the spring constants and the distances in the system.

*3. Gravity Model*

The gravity model was the most complicated system run for this analysis, and was the only analysis involving multiple fidelities. We ran four separate sensitivity analyses for the gravity model—one that dispersed the order between 47 and 53 and one each at orders 10, 50, and 75. The other dispersed parameters were the gravitational constants for the Moon, Earth, and Sun, along with the Moon diameter, for a total of 5 dispersed parameters. Order 50 is the recommended fidelity in the model's documentation. The gravitational parameters are in units of meters$^3$/(sec$^2$) and are nominally: Sun=132712440.028*10$^9$, Earth=398600.4376*10$^9$, Moon=4902.801056*10$^9$. Each of the gravitational parameters are dispersed by +/- 2*10$^7$. As

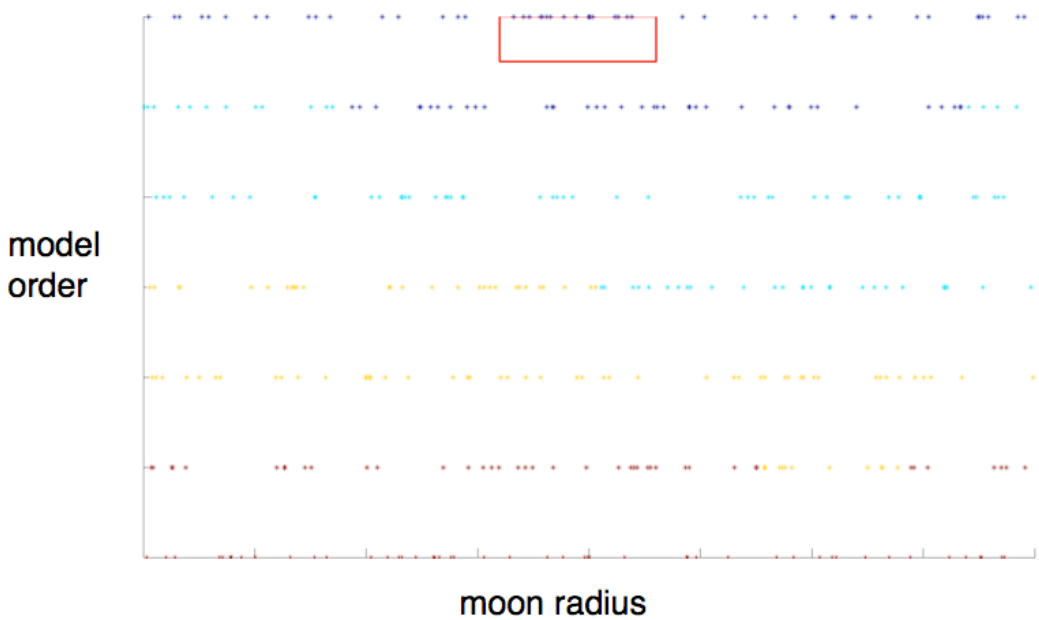


**Figure 11: Results from the Margins Analysis showing that the discrete model order has a much greater effect than any of the uncertain parameters in the model.**

an example, the actual uncertainty in the Earth's gravitational parameter is about 1 in 500,000,000. The radius of the moon is nominally 1,736,000 meters and is dispersed by +/- 2000 meters. Each trial is penalized by the root mean square of the distances between that trial's gravity vectors and the gravity vectors of a much higher-order run at the nominal gravitational parameters and Moon radius.

We first perform an analysis to see which of the parameters creates the most difference in the behavior. The results from the sensitivity analysis that included dispersed order are shown in Figure 11. The order of the model for this analysis ranges discretely from 47 to 53 (remember that 50 is the recommended order for the model). The analysis quickly illustrates that the order of the model is the largest determinant of the model behavior, even though the mean RMS Error of the model is relatively flat (as shown in Figure 6). The blue points are the points with the least error, while the red points are the ones with the most error.

We now attempt to see if we can use a lower order model to predict the behavior of higher order models. We choose three differing fidelities, at orders 10, 50, and 75 respectively, and treat each of these three fidelities as a different model. Figures 12-14 consistently show that the radius of the moon is the next most sensitive parameter. This trend is clear even though there are fewer trials at order 50 and many fewer trials at order 75. In this case we see that the behavior of the models is clearly qualitatively the same. Future work will focus on demonstrating that the models are quantitatively the same or different.

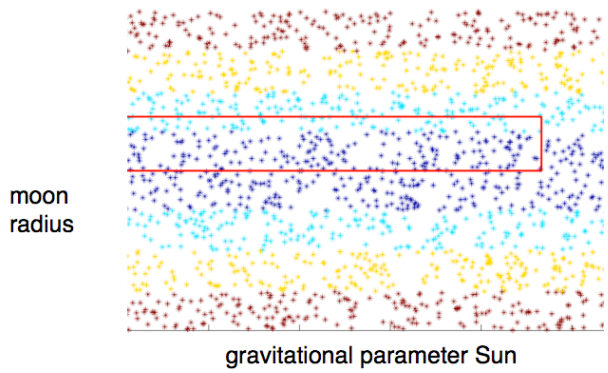


**Figure 12: Results from the Margins Analysis run at order 10 with 469 individual trials.**

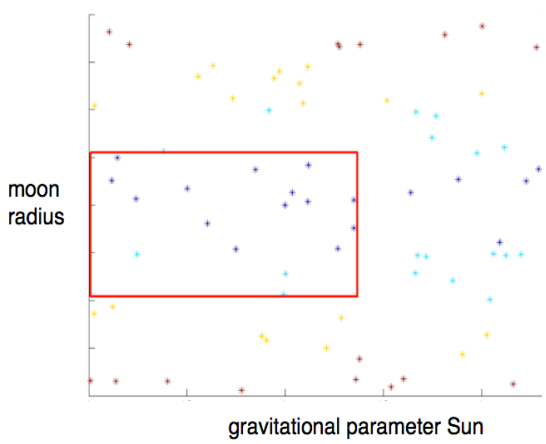


**Figure 13: Results from the Margins Analysis run at order 50 with 70 individual trials.**

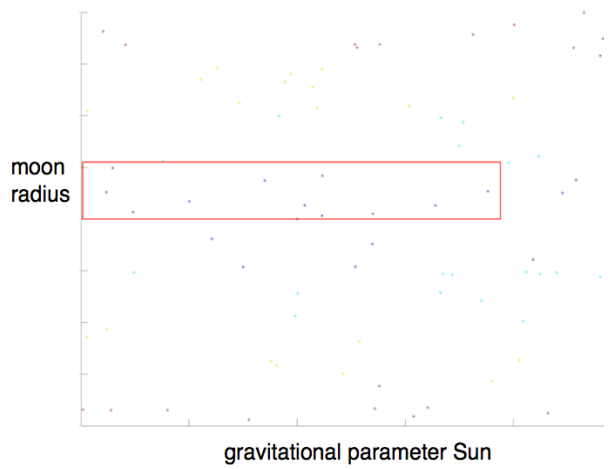


**Figure 14: Results from the Margins Analysis run at order 75 with 23 individual trials.**

## V. Conclusion

The results shown here demonstrate the benefits of performing a Margins Analysis type sensitivity analysis across models to validate space systems. At the most basic level, such an analysis reveals the most brittle inputs to the model. In the case of our simple spring-mass-damper system, we saw that the key set of variables for control was actually the linear relationship between the force and the equilibrium distance of the spring. In the case of the more complicated two mass spring-damper system, we saw that the key relationships to monitor were the relationships between the linear spring constants and the distances between the springs. Finally, for the gravity model, we saw that accurate gravity vectors depend most on the order of the model (even near the recommended order), followed by the precision of the moon's radius. The gravitational parameters for the Sun, Moon, and Earth had little to no effect.

A significant benefit of performing this type of analysis is that it forces the designers and engineers to examine the assumptions and appropriate ranges for the models. Unless these assumptions and ranges are instrumented, the sensitivity analysis will be unlikely to reveal their effects. We need research on design practices that will allow us to automatically collect this key information during the design process.

Another important area for future research is to be able to estimate how many low-fidelity and high-fidelity trials need to be run in order to have statistical confidence that low-fidelity experiments predict the behavior of high-fidelity experiments. Experiments at the highest fidelity (for example, hardware-in-the-loop testing) tend to be expensive in terms of time and also tend to come late in the design process. With as few experiments as possible, you'd like to be certain that the high-fidelity components behave in the ways predicted by your model. As an example of this sort of analysis, examine Figures 12 through 14. The trials in Figure 12 ran in approximately 11 sec., while the trials in Figure 14 take over 240 sec. to run. The standard statistical test that determines whether two datasets are from the same distribution is the two-sample Kolmogorov-Smirnov test. A future research direction should be to build ways of estimating the number of tests likely to be needed in order to achieve acceptable confidence intervals for a two-sample Kolmogorov-Smirnov test, based only on the data from lower fidelity tests.

Additionally, it is likely that we will often say with statistical certainty that our higher-fidelity systems are not adequately modeled by our lower-fidelity systems. In that case, we'd like to be able to run a sensitivity analysis that tells us which component models within the system are most in need of examination. In order to do this, we will need to a way to (preferentially automatically) instrument two different systems to determine in what ways the component models differ. A sensitivity analysis of the Margins Analysis type will have some advantage over traditional sensitivity analyses for this exploration because it easily handles both continuous and discrete input variables simultaneously. It is likely that the sensitivity analysis results given in this paper will aid the further, proposed analyses by focusing the efforts.


## Acknowledgments

This research was conducted at NASA Ames Research Center. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

## References


[1]Harland, D. and Lorenz, M. *Space Systems Failures: Disasters and Rescues of Satellites, Rockets, and Space Probes*. Springer, Chichester, 2005, pp. 216-218.

[2]Jaffe, M.S., Leveson, N.G., Heimdahl, M.P.E. and Melhart, B.E. *Software requirements analysis for real-time process-control systems. IEEE Transactions on Software Engineering*. Vol. 17, No. 3, 1991, pp. 241-258.

[3]Leveson, N.G. *Safeware: System Safety and Computers*. Addison-Wesley, 1995.

[3]Gundy-Burlet, K., Schumann, J., Barrett, T., and Menzies, T., "Parametric Analysis of ANTARES Re-entry Guidance Algorithms Using Advanced Test Generation and Data Analysis," *9th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2007.

[4]Schumann, J., Gundy-Burlet, K., Pasareanu, C., Menzies, T., and Barrett, T. "Tool Support for Parametric Analysis of Large Software Systems", *Proceedings of Automated Software Engineering, 23rd IEEE/ACM International Conference,* 2008.

[5]Gundy-Burlet, K., Schumann, J., Barrett, T., and Menzies, T., "Parametric Analysis of a Hover Test Vehicle Using Advanced Test Generation and Data Analysis," *AIAA Aerospace*, 2009.

[6]Rose, K., Smith, E., Gardner, R., Brenkert, A. and Bartell, S. "Parameter Sensitivities, Monte Carlo Filtering, and Model Forecasting Under Uncertainty," *Journal of Forecasting*, Vol. 10, 1991, pp. 117-133.

[7]Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S., *Global Sensitivity Analysis: The Primer*, Wiley, Chichester, 2008, Chaps. 1, 5.

[8]Barrett, A., "A Combinatorial Test Suite Generator for Gray-Box Testing", *Third IEEE International Conference on Space Mission Challenges for Information Technology*, 2009.

[9]Fischer, B., and Schumann, J. "Autobayes: A System for Generating Data Analysis Programs From Statistical Models," *Journal of Functional Programming*, Vol. 13, 2003, pp. 483-508.

[10]Hu, Y., "Treatment Learning: Implementation and Application," Masters Thesis, Department of Electrical Engineering, University of British Columbia, 2003.

[11]Hu, Y., and Menzies, T. "Data Mining for Very Busy People," *IEEE Computer*, Vol. 36, No. 11, 2003, pp. 22-29.

[12]Leveson, N. *Safeware: System Safety and Computers.* Addison-Wesley, Boston, 1995.

[13]NASA-STD-7009, Standard for Models and Simulations (11 July 2008).

[14]Gay, G., Menzies, T., Davies, M., Gundy-Burlet, K. "How to Automatically Find the Control Variables for Complex System Behavior." *Automated Software Engineering,* Dec, 2010. (to be published)

[15]Davies, M., Gundy-Burlet, K. Visualization of Global Sensitivity Analysis Results Based on a Combination of Linearly Dependent and Independent Directions. *AIAA Infotech 2010,* Atlanta, GA, Apr. 20, 2010.

[16]Roncoli, R. Lunar Constants and Models Document. JPL Technical Document D-32296. Sept. 23, 2005.