AIAA Modeling and Simulation Technologies Conference and Exhibit
18 - 21 August 2008, Honolulu, Hawaii

AIAA 2008-7094

Improving System Models Using Monte Carlo Techniques on Plant Models

Amory Wakefield[1] and Steve Miller[2]
The MathWorks, Inc.

Model-Based Design has become a standard in the aerospace industry. A system model is at the center of the development process, supporting the design of complex algorithms combined with physical hardware. In addition to the advantages that come from modeling control algorithms, modeling plants can lead to more robust designs. Using commercial-off-the-shelf (COTS) software allows both the controller and hardware to be modeled and simulated in a single environment. Modeling plants in the same simulation environment as an embedded controller enables engineers to test a controller with multiple plant parameters, as well as simulating with nominal or ideal values. Modeling variable physical parameters provides a better representation of what will happen in real hardware. Monte Carlo analysis is a standard method of simulating variability that occurs in real physical parameters. In aerospace applications, Monte Carlo techniques can be used to ensure high-quality, robust designs. Even using a shared COTS environment, fully testing or optimizing a design can take thousands of simulation iterations and days to complete. Depending upon the complexity of the system and fidelity of the model, each iteration could take hours to run. Simulation time can become a critical bottleneck in the development process. Being able to run multiple, independent scenarios in parallel can lessen this time significantly. In this paper, we will discuss techniques for modeling, optimizing, and testing plant models to build better system models in MATLAB® and Simulink® from The MathWorks. We will also present new techniques for speeding up Monte Carlo techniques by using high-performance computing clusters.

I. Introduction

Model-Based Design has become a standard in the aerospace industry.[1,2,3] A system model is at the center of the development process, supporting the design of complex algorithms combined with physical hardware. In addition to the advantages that come from modeling control algorithms, modeling plants can lead to more robust designs. The first design an engineer creates is rarely the design that will end up in production. The efficient development of complex systems requires many iterations to improve upon the initial design. This requires an ability to rapidly set up tests, simulate, and evaluate the test results. A control algorithm designed with an inaccurate or nonexistent plant model is unlikely to work properly, and a system designed without the controller cannot be properly tested. Using commercial-off-the-shelf (COTS) software that allows both the controller and hardware to be modeled and simulated in a single environment enables faster development and higher-quality designs by allowing testing to happen on the entire system early in the design cycle.[4,5,6,7] Problems or errors are found before implementation on hardware, when they are both cheaper and faster to fix.

Aerospace systems are generally composed of numerous subsystems, each of which may be designed by a different team of engineers, sometimes spread around the globe. The different teams must work together to optimize the design, or even to get it to function according to specified requirements. It is necessary to perform tradeoff analyses at the system level, including all the subsystems that are involved. Without a single, standardized modeling environment, co-simulation techniques or manual calculations are

[1] Technical Marketing Manager, Simulation and Test Applications, 3 Apple Hill Drive, Natick, MA.
[2] Technical Marketing Manager, Physical Modeling, Adalperostr. 45, Ismaning, Germany.

needed. These can be difficult or very time consuming. A shared modeling environment and common modeling standards can also help teams communicate quickly and effectively across organizational, linguistic, and cultural barriers.

Modeling plants in the same simulation environment as an embedded controller enables engineers to test a controller with multiple plant parameters, as well as to simulate with nominal or ideal values. Modeling variable physical parameters provides a better representation of what will happen in real hardware. Monte Carlo analysis is a standard method of simulating variability that occurs in real physical parameters. In aerospace applications, Monte Carlo techniques can be used to ensure high-quality, robust designs. Even using a shared COTS environment, fully testing or optimizing a design can take thousands of simulation iterations and days to complete. Depending upon the complexity of the system and the fidelity of the model, each iteration could take hours to run. Simulation time can become a critical bottleneck in the development process. Being able to run multiple, independent scenarios in parallel can lessen this time significantly. New high-performance computing tools and multiprocessor machines have eliminated the time and resource limitations in many cases by providing the processing power needed to vary large numbers of parameters in complex dynamic models.

In this paper, we discuss techniques for modeling, optimizing, and testing plant models to build better system models in MATLAB® and Simulink® from The MathWorks. We also present new techniques for speeding up Monte Carlo analyses by using high-performance computing clusters.

II. Monte Carlo Methods

Monte Carlo methods may be the most commonly applied statistical method in engineering and science disciplines, used in everything from financial modeling to theoretical physics problems. Literally tens of thousands of books on technical subjects reference them. While many specific applications exist, in their simplest form, Monte Carlo methods involve using random numbers and probability distributions to explore problems.[8] In this paper, we discuss the use of Monte Carlo techniques as applied to system simulation.

Among the numerous benefits of using Monte Carlo methods for simulation is that it allows you to model systems that are too complex for an analytic solution. It also allows you to simulate uncertainty. Even when a design has few uncertain parameters, the conditions or environment it will operate in almost always contains uncertainty that needs to be considered; one common example of uncertainty is the wear on a physical system, which will tend to erode performance over time. When attempting to design a system that will work over a long period of time, simulating how it will perform over its entire lifetime involves all kinds of uncertainty. One can use Monte Carlo methods to predict the likelihood of failure after a certain time.

A third benefit is providing increased confidence that a model is robust using Monte Carlo testing. An example of this is the process of taking a completed algorithm design and testing it closed-loop using not only nominal parameter values, but random values that span the entire range of possible situations the design will encounter. Take the previously published example of a DC motor.[9,10] Every physical component used to make the motor contains some tolerance on its dimensions. Simulating solely with the nominal values of each dimension may not give an accurate representation of the range of performance of the motor. One could test the minimum and maximum values of each dimension, but that would also not necessarily create an accurate model, unless each of those dimensions followed a uniform distribution. As many physical parameters fit a normal, or Gaussian, distribution, using Monte Carlo techniques to generate random values based on that distribution and then simulating with those will give a much more realistic prediction of how the motor will behave when built.

One drawback of Monte Carlo simulations is that they can involve running thousands of iterations of a single model. In fact, the more complex the system being simulated or the more uncertainty it contains, the more simulations you will have to run. It can sometimes take days to run a complex model hundreds or thousands of times.

In this paper, we focus on the third benefit of Monte Carlo simulation for control system design, as well as mitigating the main drawback of time needed to complete the simulations. We describe using Monte Carlo methods to ensure a plant model more accurately reflects real-world plants. We then describe how Monte Carlo testing can use that same plant model to test an algorithm model for robustness.

III. Distributed Computing

Running simulations in a high-performance computing (HPC) environment was once prohibitively expensive due to the cost of installing and maintaining sufficient computing power. HPC was available mainly to government agencies and large research laboratories, the only groups with the means to purchase supercomputers. Today, most supercomputers have been replaced by COTS computer clusters that provide affordable, high-performance, distributed computing environments. With the increased availability of multicore and multiprocessor computers, and the growth of computing clusters, many organizations already have vast CPU processing power installed on site.

Though the processing power is available, running an individual engineer's simulations on the cluster or on multiple processors is not always straightforward, particularly from within a COTS simulation software tool. One difficulty engineers encounter is that the time to set up and maintain their COTS simulation software on a cluster can eat into the time savings the additional computing power can provide.[11] In fact, the difficulty in setting up distributed applications is still a barrier to tapping the processing power in one computer, let alone a cluster of perhaps hundreds. Yet, taking advantage of these computing resources is a critical step to increasing the number of Monte Carlo simulations one can reasonably run.

Some computing problems are called embarrassingly parallel or coarse-grained because they can be segmented easily to run on several nodes without communication, shared data, or synchronization points between the nodes. The run-time on each node dominates the time needed to start and stop the application. Monte Carlo simulations fall into this category, and thus are an obvious candidate to run on computing clusters. Some form of integration between the simulation software and the HPC cluster is needed to maximize the investment in computing power.

One possibility to run simulations on a cluster without needing COTS software installed on every node is to compile the model into an executable.[12] For example, one can generate ANSI/ISO C/C++ code from Simulink models. The separate executable can be executed from a DOS command line (on Windows® platforms), thus requiring scripts to get data into and out of the executable. While this technique is very useful for accelerating simulations, to use it to run Monte Carlo analyses, the user must script the parameter variations and collection of results from the multiple executables into a usable format. If using a remote cluster, additional scripting may be needed for communication with a scheduler and job creation. This can be tedious and often requires help from a cluster administrator to customize a particular simulation setup.

SystemTest™[13] and MATLAB Distributed Computing Server™[14] are examples of COTS tools that eliminate the tedious scripting and compilation needed to use standalone executables. These tools make it possible to set up a series of parameters in a graphical user interface (GUI) to vary in a model for Monte Carlo testing and then define it as a job to be run on multiple processors. Launching the job from the GUI eliminates the scripting needed to communicate to a cluster, create a job, and compile the results from individual simulation runs. This makes it possible to distribute complete Simulink models for execution in a cluster or in a multicore or multiprocessor computer without needing to write any lines of code. This integration of COTS simulation tools with HPC schedulers is another time-savings for engineers looking to speed up Monte Carlo runs.

IV. Case Study

A. The Model

The system we are analyzing is a hydromechanical actuator that controls the angle of an aileron. The purpose of this model is to analyze the effects of parameter variation in the servo-valve, the analog circuit controlling the opening of the servo-valve, and the effects of varying the load on the aileron. We will study the effects these parameters have on the system's stability and performance by examining the rise time and settling time.

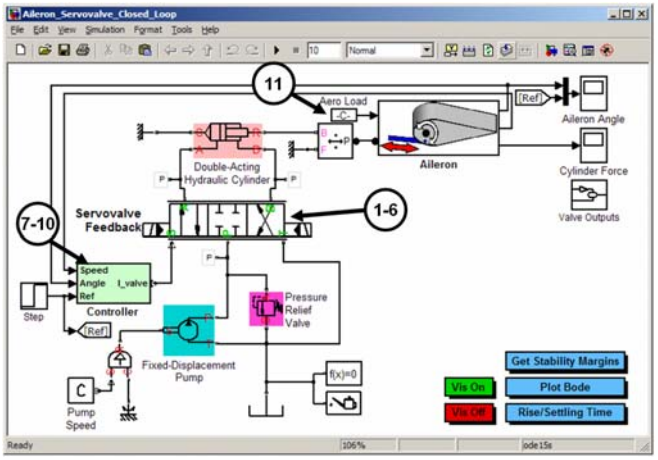

Figure 1. Model of a closed-loop aileron system. Blocks with parameters of interest are labeled with numbered circles: 1-6) hydraulic servo-valve parameters, 7-10) analog circuit implementation of the controller, 11) aerodynamic load.

Using COTS physical modeling tools,[15,16,17,18] we developed a model of the plant and the controller in Simulink. We chose to use the physical modeling tools to develop a high fidelity plant model and simulate how the plant will behave in the real world as closely as possible. Figure 1 shows the complete system model, including the servo-valve, double-acting hydraulic cylinder, aileron, and the analog circuit implementation of the controller. Figure 2 shows in detail the analog circuitry of the controller, implemented using SimElectronics™.

By using a closed-loop model, we are able to vary both controller and plant parameters and simulate the entire system's performance in one environment. We have chosen parameters in both the plant and the controller to vary using Monte Carlo techniques. We modeled these parameters using nominal values, but know that real measured values will vary. We also have some tolerance data on how much these might vary in real, physical components. We need to use this data to understand how they could impact the performance of the system.

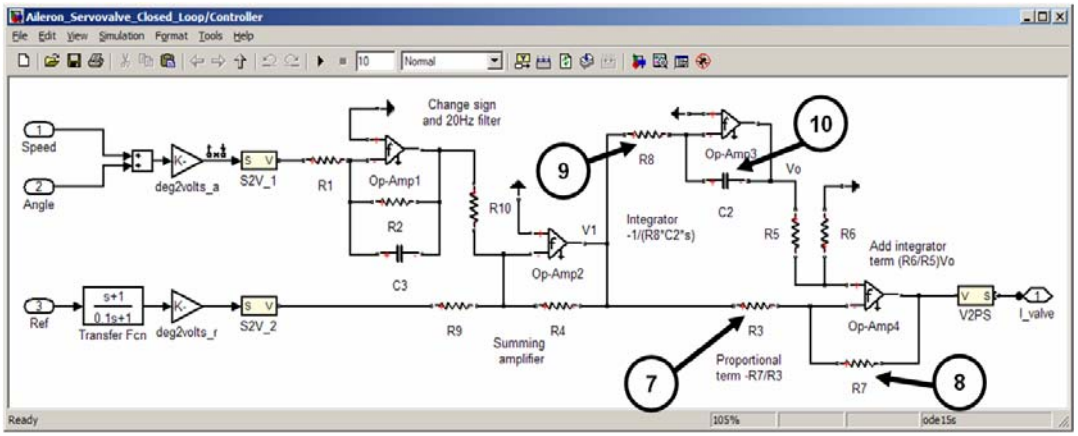

Figure 2. Model of the analog circuit implementation of the controller. Blocks with parameters of interest are labeled with numbered circles.

B. The Monte Carlo Test

To examine the known variation in these physical parameters, we set up a Monte Carlo test in SystemTest. We used Probability Distribution test vectors and some MATLAB code to set it up. Our goal is to understand the stability of the system and the impact all these parameters' variation might have on it. Using normal (Gaussian), lognormal, and uniform probability distributions, we generated 1000 random values for 10 of the parameters. For the final parameter, we wanted to insure we tested specific values, so we used a MATLAB expression to generate values that span the entire range of possible values. A Limit Check element was used to set pass criteria based on rise time and settling time calculated running with the nominal values in the model.

Figure 3 shows the flow of our test setup. We are running the Simulink model, then calculating the rise and settling times using MATLAB, then running a Limit Check element to determine whether the test passes or fails. Figure 6 also shows the list of test vectors, which are overriding the parameters in the Simulink model during the test. An advantage of using a separate test executive like SystemTest to run the Monte Carlo test is that it does not require us to modify the model under test to vary the parameters. Another is that storing the test setup independently of the model enables parallel development on the test and the model.

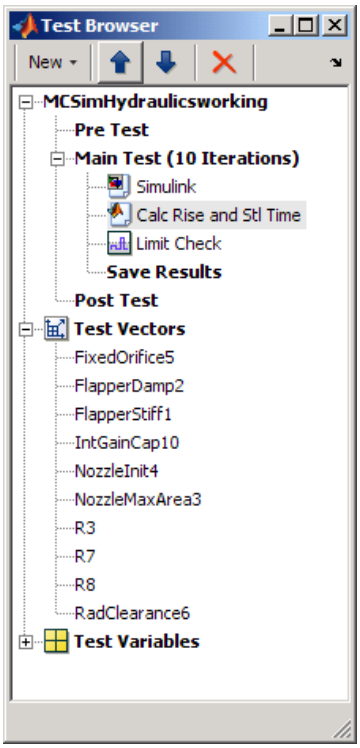

Figure 3. Monte Carlo test setup in SystemTest.

C. The Results

The first test took over seven hours to run, which was far too long for us. We ran it again on a computing cluster. Using SystemTest, we could distribute the test automatically using MATLAB Distributed Computing Server without altering our test setup in any way. We ran the test on an 8-core cluster, configured using two 64-bit Linux machines with four workers on each. This resulted in faster completion of just over an hour. The detailed results are shown Table 1.

Note that the speed improvement is not linearly dependent on the number of workers. This has been previously reported[10,19] and is due to the overhead for copying over the files from the host machine to each machine on the cluster, transmitting the input and output data between the workers and the job manager, and the network latency. In general, total simulation time should be significantly longer than this overhead to achieve speed benefits from distributing tasks. The results obtained here are specific to the model and the example cases used.

Because SystemTest is tightly integrated with Parallel Computing Toolbox and MATLAB Distributed Computing Server, the test report and MAT-file of the results are identical to those generated using a single processor and taking almost seven times as long to complete.

The first test also illustrates that our design is quite sensitive to some parameters. Figure 4 illustrates one example of the impact on the cylinder force curve. R7, labeled Parameter 8 in Figure 2, caused a lot of instability in the cylinder force, causing most test cases to fail. This also lengthened the test, because simulating the oscillations in detail made each simulation take longer than it had taken to run using the nominal values.

For the second test, we tightened the tolerance (standard deviation) on the R7 specification by a factor of 10. This eliminated the instabilities; however, we still did not meet the rise and settling time tolerances we had set (10% from nominal) in almost all cases. The second test took only one hour to run, but, since we had access to a cluster, we also ran it in distributed mode and it finished in about 10 minutes time using the same cluster described above. Table 1 shows the exact times of this second test as well.

At this point, further analysis of our design, criteria, and specifications is needed to determine the best way to improve our design. This could include varying more parameters in our model, re-examining our controller design, comparing the plant model to lab data to ensure our model is accurate. We could also apply other control design software, such as Simulink Response Optimization™ to help improve our design.

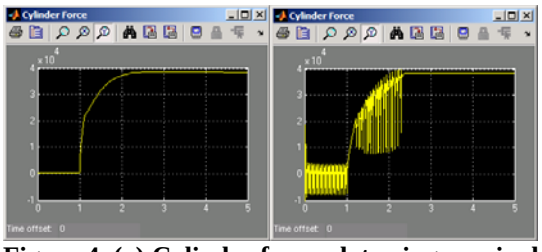

Figure 4. (a) Cylinder force plot using nominal parameter values. (b) Representative cylinder force plot from first Monte Carlo test. Note the frequent oscillations.

For the purposes of this paper, our testing is complete. We have run Monte Carlo tests on a substantial set of parameters, found at least one component of our design that must have a tight tolerance on its value, and shown a technique to speed up the testing by almost a factor of 7.

Table 1. Exact execution time for each test and calculated factor of time reduction using a computing cluster.

	1-core (Windows machine)	8-core (Linux cluster)	Time Reduction
Test 1	7:38:41	1:07:54	6.8x
Test 2	1:04:47	0:11:38	5.6x

V. Conclusion

This paper discussed three main points: system modeling with accurate plant models, Monte Carlo testing, and distributing tests. Putting these together greatly increases the quality of a design in a reasonable amount of time.

Model-Based Design means having a model of the design. By designing and simulating only a controller, an engineer can miss key information about the design that modeling the plant could illustrate. By using Simulink and its add-on tools, a designer can model an entire system, including physical components with a high degree of accuracy. This enables better testing of designs early in the development process, leading to better designs, early discovery of errors, and a higher-quality end product.

Monte Carlo techniques are one way of examining a design, testing its robustness, and ensuring coverage of the whole design space. Using probability distributions to randomly generate parameter values better simulates real-world variation of physical components, both those in the plant and the controller. The test we ran in this paper illustrates how variation in just one component can sometimes have a dramatic effect on the performance of a design. Correctly specifying physical components is one solution. Designing controllers that take into account the full variability of a plant is another. Monte Carlo can help both. COTS testing tools can make Monte Carlo simulations easier to manage and faster to run, especially when combined with a HPC cluster.

We also demonstrated how distributed testing, or running tests on a compute cluster, can significantly reduce the time needed to fully test a design. In the relatively simple example model and test setup that we used, we had to run the 1000 iterations twice, changing a single parameter's tolerance. Rather than this test taking over seven hours to complete, we were able to show identical results generated in one hour by taking advantage of a cluster of machines. In the second case, instead of taking an hour, we were able to complete testing in eleven minutes. This is a powerful tool for decreasing total testing time.

In summary, core features of Model-Based Design are simulation and continuous testing. These activities result in higher-quality designs and quicker design iterations. Combining Monte Carlo tests and a distributed environment makes it faster and easier to fully test a design and iterate it. The ability to model an entire system in the same environment and simulate it is key to achieving these benefits.

Acknowledgments

The authors would like to thank Amit Mahajan for his help in setting up a compute cluster and his advice on running the test distributed to achieve the best results. The authors would also like to thank Valery Tchkalov and Rick Hyde for their help in developing the model used in this paper.

References

[1] Barnard, P., "Graphical Techniques for Aircraft Dynamic Model Development," *AIAA Modeling and Simulation Technologies Conference and Exhibit* [CD-ROM], Providence, RI, 2004.

[2] Aberg, R., and Gage, S., "Strategy for Successful Enterprise-Wide Modeling and Simulation with COTS Software," *AIAA Modeling and Simulation Technologies Conference and Exhibit* [CD-ROM], Providence, RI, 2004, ID: 2004-4929.

[3] Krasner, J., "Model-Based Design and Beyond: Solutions for Today's Embedded Systems Requirements," Embedded Market Forecasters, American Technology International, Framingham, MA, January 2004.

[4] Wood, G. D., and Kennedy, D. C., "Simulating Mechanical Systems in Simulink and SimMechanics," Technical Report 91124v00, The MathWorks, Inc., Natick, MA, 2003.

[5] Tung, J., "Using model-based design to test auto embedded software," *EE Times* [online journal], 09/24/2007 7:54 AM EDT, URL: http://www.eetimes.com/showArticle.jhtml?articleID=202100792 [cited 24 September 2007].

[6] Denery, T., Ghidella, J. R., Mosterman, P. J., and Shenoy, R., "Creating Flight Simulator Landing Gear Models Using Multidomain Modeling Tools," *AIAA Modeling and Simulation Technologies Conference and Exhibit* [CD-ROM], Keystone, CO, 2006, ID: 2006-6821.

[7] Popinchalk, S., Glass, J., Shenoy, R., and Aberg, R, "Working in Teams: Modeling and Control Design within a Single Software Environment," *AIAA Modeling and Simulation Technologies Conference and Exhibit* [CD-ROM], Hilton Head, SC, 2007.

[8] Drakos, N., *Introduction to Monte Carlo Methods*, Computational Science Education Project. URL: http://www.ipp.mpg.de/de/for/bereiche/stellarator/Comp_sci/CompScience/csep/csep1.phy.ornl.gov/mc/mc.html [cited 4 June 2008].

[9] Kozola, S. and Doherty, D., "Using Statistics to Analyze Uncertainty in System Models," *MATLAB Digest* [online journal], May 2007, URL: http://www.mathworks.com/company/newsletters/digest/2007/may/uncertainity.html [cited 21 February 2008].

[10] Wakefield, A., "Using Multiple Processors for Monte Carlo Analysis of System Models," *SAE Congress*, Detroit, MI, 2008, ID: 2008-01-1221.

[11] Hutton, C., "HPC in the Kitchen and Laundry Room: Optimizing Everyday Appliances for Customer Satisfaction and Market Share," *SC07*, Reno, NV, 2007.

[12] *Real-Time Workshop® User's Guide*, The MathWorks, Natick, MA, September 2007.

[13] *SystemTest User's Guide*, The MathWorks, Natick, MA, September 2007.

[14] *MATLAB Distributed Computing Engine System Administrator's Guide*, The MathWorks, Natick, MA, September 2007.

[15] *Simscape User's Guide*, The MathWorks, Natick, MA, March 2008.

[16] *SimElectronics User's Guide*, The MathWorks, Natick, MA, April 2008.

[17] *SimHydraulics User's Guide*, The MathWorks, Natick, MA, March 2008.

[18] *SimMechanics User's Guide*, The MathWorks, Natick, MA, March 2008.

[19] Ghidella, J., Wakefield, A., Grad-Freilich, S., Friedman, J., and Cherian, V., "The Use of Computing Clusters and Automatic Code Generation to Speed Up Simulation Tasks," *AIAA Modeling and Simulation Technologies Conference and Exhibit* [CD-ROM], Hilton Head, SC, 2007.