Development of Monte Carlo Capability for Orion Parachute Simulations

James W. Moore[1]
Jacobs ESCG, Houston, TX, 77598

Parachute test programs employ Monte Carlo simulation techniques to plan testing and make critical decisions related to parachute loads, rate-of-descent, or other parameters. This paper describes the development and use of a MATLAB-based Monte Carlo tool for three parachute drop test simulations currently used by NASA. The Decelerator System Simulation (DSS) is a legacy 6 Degree-of-Freedom (DOF) simulation used to predict parachute loads and descent trajectories. The Decelerator System Simulation Application (DSSA) is a 6-DOF simulation that is well suited for modeling aircraft extraction and descent of pallet-like test vehicles. The Drop Test Vehicle Simulation (DTVSim) is a 2-DOF trajectory simulation that is convenient for quick turn-around analysis tasks. These three tools have significantly different software architectures and do not share common input files or output data structures. Separate Monte Carlo tools were initially developed for each simulation. A recently-developed simulation output structure enables the use of the more sophisticated DSSA Monte Carlo tool with any of the core-simulations. The task of configuring the inputs for the nominal simulation is left to the existing tools. Once the nominal simulation is configured, the Monte Carlo tool perturbs the input set according to dispersion rules created by the analyst. These rules define the statistical distribution and parameters to be applied to each simulation input. Individual dispersed parameters are combined to create a dispersed set of simulation inputs. The Monte Carlo tool repeatedly executes the core-simulation with the dispersed inputs and stores the results for analysis. The analyst may define conditions on one or more output parameters at which to collect data slices. The tool provides a versatile interface for reviewing output of large Monte Carlo data sets while preserving the capability for detailed examination of individual dispersed trajectories. The Monte Carlo tool described in this paper has proven useful in planning several Crew Exploration Vehicle parachute tests.

I. Introduction

THE test program to design and validate the parachute recovery system for the NASA Crew Exploration Vehicle (CEV) includes ambitious objectives and complex test techniques.[6,7] To examine the effectiveness of these techniques, engineers perform trajectory simulations prior to the tests. Due to the variability in atmospheric conditions and the random and chaotic nature of parachute inflations, many of the simulation inputs are not precisely known and Monte Carlo analysis is required. The CEV Parachute Assembly System (CPAS) development program has created a MATLAB-based Monte Carlo tool to facilitate these analyses. The Monte Carlo Decelerator Simulation (MCDS) disperses inputs, performs multiple executions of trajectory simulations, and provides tools to analyze the results. This paper will briefly describe some of the parachute simulations used by the program to provide insight into the Monte Carlo architecture. The Monte Carlo inputs will be discussed and the execution process will be explained. The Monte Carlo output review and analysis capabilities will be examined in detail. Finally, limitations and future improvements to the tool will be discussed.

II. CPAS Trajectory Simulations

The CPAS project employs four parachute trajectory simulations, three of which can be employed in a Monte Carlo analysis with MCDS. In some sense, MCDS can be thought of as a Monte Carlo analysis manager for several other simulations. To understand how MCDS works it is helpful to review the "core" simulations that it manages.

DSS (Decelerator System Simulation) is a legacy 6 Degree-of-Freedom (DOF) parachute trajectory simulation based on the UD233A[3] simulation used by the Space Shuttle Solid Rocket Booster parachute project. DSS is written in the Fortran programming language and user input is provided via text files. DSS is the highest fidelity NASA-maintained simulation used by the CPAS project.

[1] Analysis Engineer, Aerothermal and Flight Mechanics, 455 E. Medical Center Blvd., Webster, TX.

DSSA[1,2] (Decelerator System Simulation Application) is a 6-DOF parachute trajectory simulation based on DSS that includes the capability to model the extraction of a test platform from an aircraft. Like DSS, this simulation is written in Fortran. DSSA employs a spreadsheet front end to configure the input file for a compiled executable. MCDS was originally designed as a Monte Carlo extension of DSSA. The similarity of the input files and the advent of a common simulation output structure[4] made it relatively simple to add Monte Carlo capability to DSS.

DTVSim (Drop Test Vehicle Simulation) is a 2-DOF parachute trajectory simulation intended for conceptual planning of test trajectories. This simulation includes parachute inflation models and is appropriate for early assessment of parachute loads as well as available altitude studies, programmer parachute sizing, and reefing ratio selection. DTVSim is MATLAB-based, which simplifies the interaction with MCDS.

In addition to the simulations mentioned above, the CPAS parachute manufacturer, Airborne Systems, uses a proprietary simulation named DCLDYN[8] (Decelerator Dynamics) that provides an independent check on simulation results. Airborne Systems maintains a Monte Carlo capability that is not part of the MCDS architecture.

Because of the variety of core-simulations and input and output formats, MCDS was designed to perturb an existing set of simulation inputs (created in the traditional tools) rather than directly configure the various simulations within the Monte Carlo manager. In addition, no effort has been made to tailor the input or output formats of the various simulations to simplify the tasks performed by MCDS. This approach has made it possible to extend MCDS to three simulation environments without requiring code changes to any of the core-simulations.

III. Monte Carlo Work Flow

The MCDS system breaks a Monte Carlo analysis into four steps. This process is outlined in Fig. 1. The first step is to develop a nominal trajectory in the desired simulation tool. This may require some iteration until the desired test conditions and requirements are met.

In the second step, the analyst chooses a set of dispersions to be applied to the nominal simulation inputs. MCDS will generate a set of randomized inputs for each dispersed cycle (replication). The analyst must determine which inputs to disperse and the distributions from which to draw the randomized inputs. This is accomplished by providing MCDS with a dispersion rules file that identifies the variables to disperse, the type of probability distribution functions to use, the magnitude of the dispersions, and additional information about how to apply the dispersions to the nominal input file. Probability distribution functions employed by the CPAS project include the uniform and normal distributions. MCDS is easily extendable to any probability distribution that can be coded in MATLAB. The magnitude of the dispersion may be expressed as a standard deviation or as a set of max and min values within which to disperse. This two-step process for generating dispersed inputs is a concession that was made to allow the continued use of the traditional input file formats.

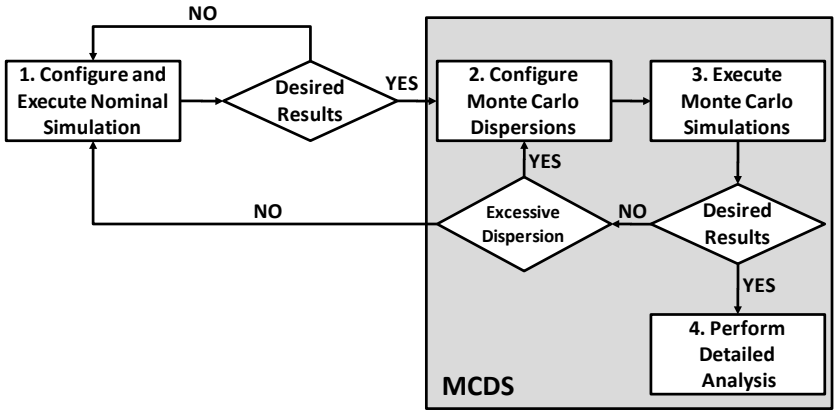

Figure 1. Monte Carlo work flow. *Monte Carlo simulations may expose configuration problems that were not apparent in the nominal simulation results. Therefore, undesired Monte Carlo results may require re-execution of the Monte Carlo set with modified dispersions or re-configuration of the nominal simulation itself.*

After the analyst has established a nominal case and decided on a set of dispersions to apply, the third step is to build and execute the dispersed simulations and perform a preliminary review. MCDS performs the build and execute tasks automatically with a process that is described below. Once the simulations have been executed the analyst reviews the results using MCDS. If the results are unacceptable, then the analyst may have to refine the dispersions or modify the nominal case.

When the analyst has confirmed that the nominal simulation is configured correctly and the dispersions are appropriate, the final step is to perform a detailed analysis of the results. MCDS provides tools to trace dispersed

results to their inputs, compare cycle results against each other, find peak values and other characteristics of the data, as well as tools that prepare the output for peer review. As shown in Fig. 1, MCDS manages steps two, three, and four. The initial configuration step is left to the legacy tools.

MCDS breaks the actual execution of the Monte Carlo (step 3 in Fig. 1) into four sub-steps as shown in Fig. 2. First, the dispersion rules file is read and the dispersions are applied. MCDS generates an array for each simulation input. The arrays are populated with random values taken from the distributions defined for each simulation input in the dispersion rules file.

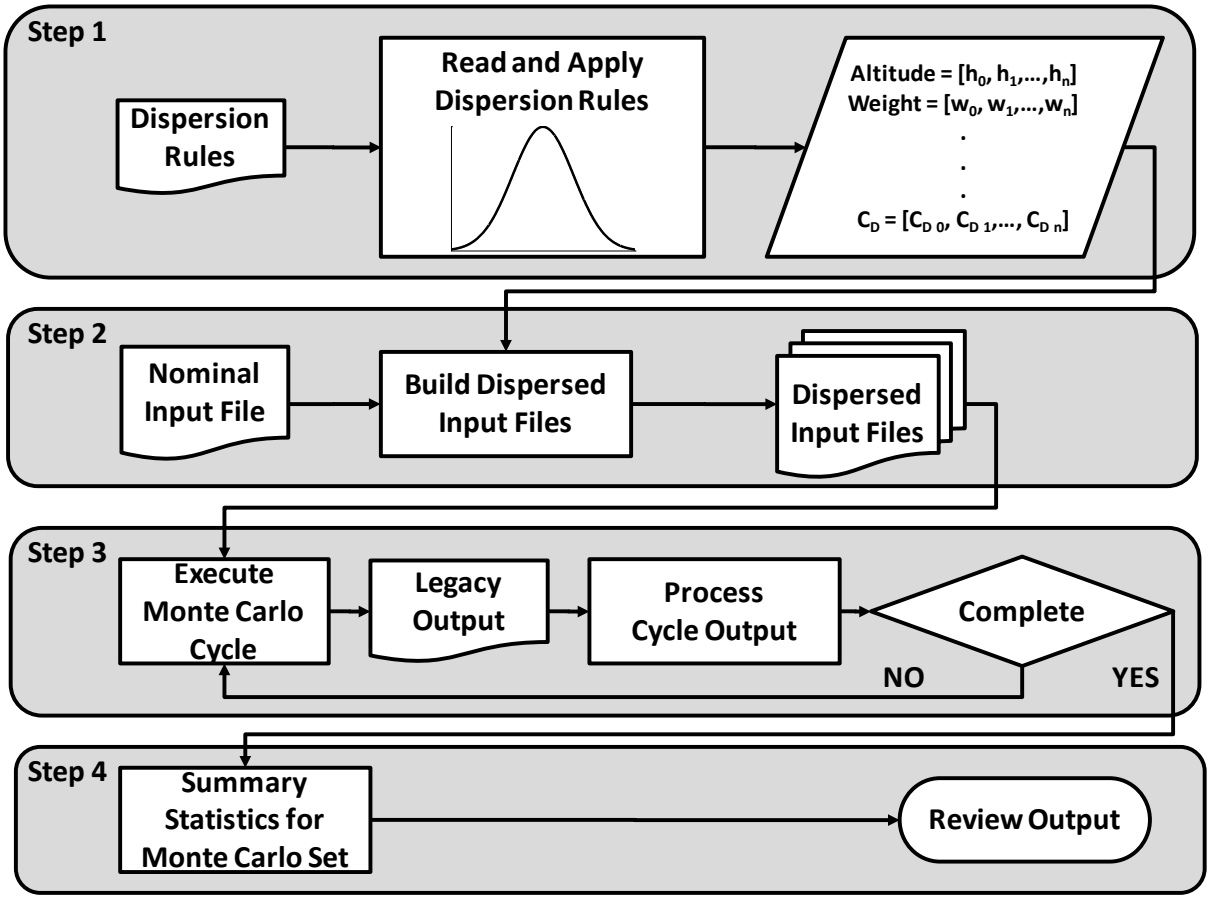

Figure 2. MCDS execution. *MCDS breaks the Monte Carlo execution into four steps: dispersion, input file building, simulation execution, and summarization.*

In the second step, MCDS builds a series of perturbed input files using the dispersed input arrays. For the first dispersed input file, the nominal input values are replaced by the first element of each dispersed input array. The second element in each array is included in the second dispersed input file, and so on. The reliance on the legacy input file formats complicates this process. The file formats must be interpreted and the inputs must be matched with the appropriate dispersions. The nominal input file is parsed for keys that identify each input variable. The nominal value of that variable is then replaced with the dispersed value. This is repeated for each input variable. When complete, the new text is saved as a dispersed input file.

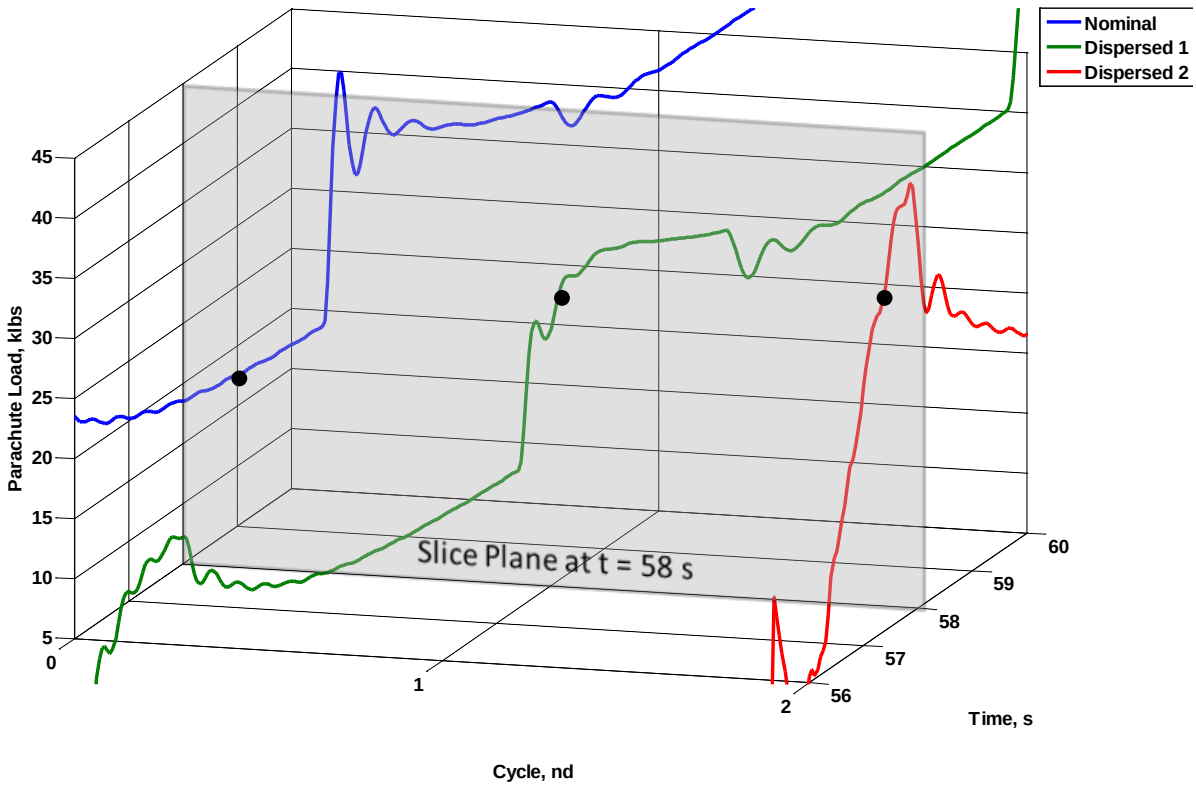

Figure 3. Monte Carlo slice schematic. *The Monte Carlo analyst may be interested in comparing the values of certain parameters when each simulation meets a specified condition. Here the parachute load at time equals 58 s is compared.*

In step three, MCDS executes the legacy simulation using the dispersed input files. Currently, this execution is performed in series. As each simulation completes, MCDS reads the legacy output files and collects information about the run. This information includes complete trajectory data as well as peaks and other slicing values at user-defined conditions. Here, slices are sets of output parameter values from each trajectory that are taken when the trajectories meet a specified condition. It is helpful to picture the slicing procedure with a 3-dimensional plot of the Monte Carlo trajectory data. A typical 2-dimensional plot is extended to 3-dimensions by using the cycle number as an additional independent variable. The slicing operation can be viewed as a plane cutting across all trajectories (and all output parameters) when the specified condition is met. Figure 3 illustrates the concept of a Monte Carlo slice. In this simplified example, the analyst desires the

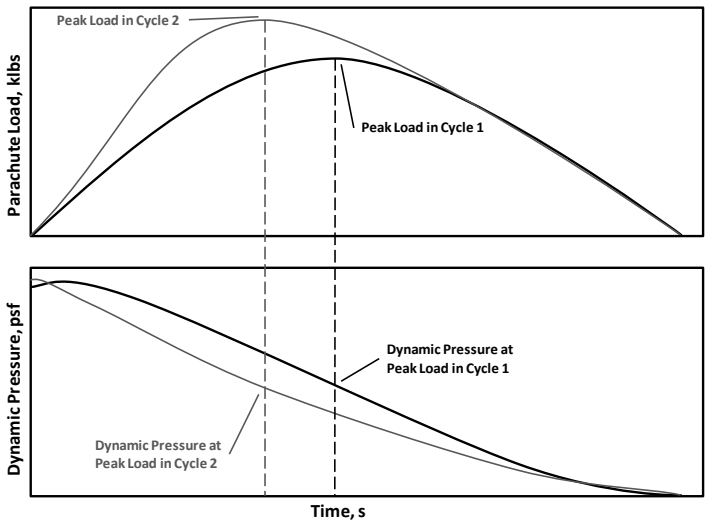

Figure 4. Comparing dispersed cycles at key events. *Slicing across each trajectory parameter allows comparison of dispersed cycles at key events. Here, the dashed lines represent slices.*

value of the parachute load in each dispersed cycle at the moment when time equals 58 s. Frequently, the simulation time is not a reliable basis for comparing trajectories since events will occur at different times between dispersed cycles. For example, the analysis will typically be interested in the peak load experienced during each inflation stage. This peak load will occur at different times depending on the input parameters.

In MCDS, the analyst defines the slices that are collected by providing a list of MATLAB functions that detect the desired conditions. Slices are collected across all trajectories and across each output parameter. In this way, the analyst can evaluate the values of each parameter, in each dispersed cycle, when a specified condition is met. Figure 4 illustrates how the slicing capability might be used to evaluate the dynamic pressure at the peak parachute load for each dispersed cycle. The dashed lines represent slices that detect the peak load in an individual cycle. The array index corresponding to the peak load can be used to find the value of any other parameter at that time. The most commonly used slicing functions are included as part of the MCDS package but the user may define new functions to meet analysis needs.

After all required data has been collected from the simulation output, the core-simulation is executed with the next dispersed input file. After all Monte Carlo cycles have been executed, MCDS performs basic statistical computations on the entire Monte Carlo data set. These will include finding the maximum, minimum, mean and standard deviation of each output parameter at each slice condition. Once the summary calculations are complete, MCDS converts to a customized plotting utility for reviewing the results.

IV. Output Review and Analysis

Monte Carlo analyses generate a significant amount of data. MCDS provides the trajectory analyst with multiple tools to extract the important information and present it to stakeholders for review. The tool includes methods to plot trajectory data, compare results of dispersed cycles, identify extreme values, and assess potential erroneous results.

Trajectory plots are useful for assessing overall performance of the dispersed cases. MCDS co-plots the nominal trajectory with a select number of dispersed trajectories. Figure 5 shows a typical plot of parachute load vs. time. In

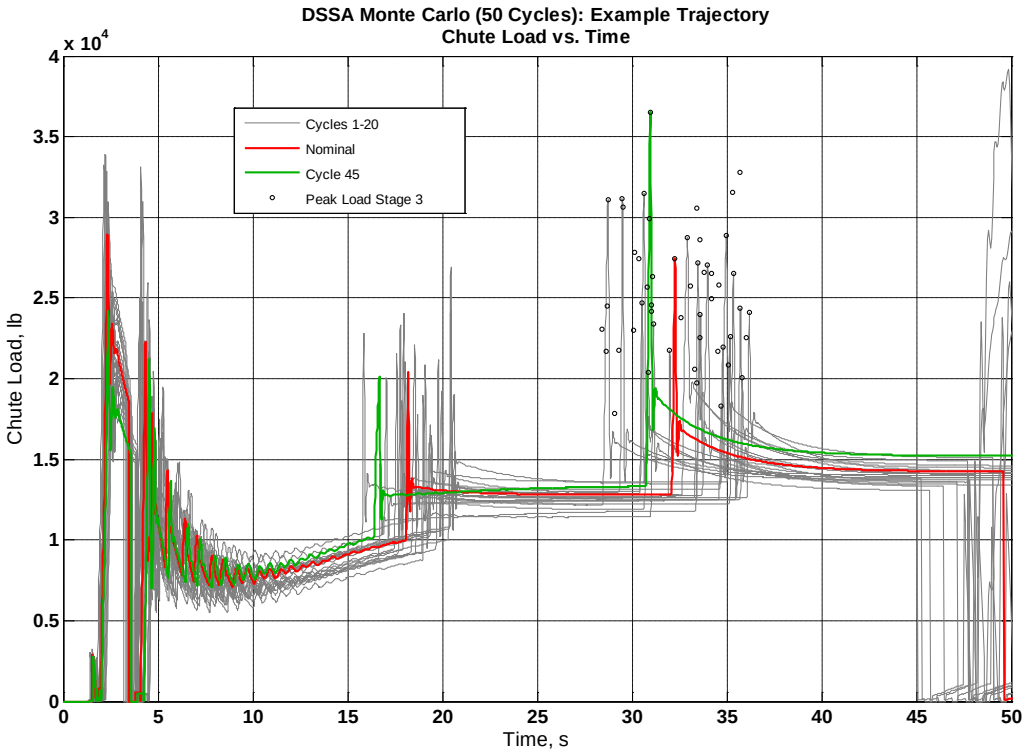

Figure 5. MCDS trajectory plot. *The trajectory plot is useful for comparing trends and identifying spurious cycles. The nominal case is highlighted in red. Here, the cycle with the highest, third-stage peak load is highlighted in green. The gray lines depict a subset of the dispersed cycles and the circles identify the third-stage peak load for all cycles.*

general, the dispersed trajectories will form a band of trajectories about the nominal case. This method is particularly useful for spotting cases that diverge from the basic trend followed by the majority of cycles. The user interface provides a selectable list of parameters to use as the dependent and independent variables so that any output variable may be plotted against any other output variable. However, most analyses involve time histories of parameters like altitude, dynamic pressure, and parachute loads. Due to memory constraints, typically only 20 cycles are plotted at a time. The analyst may select any dispersed trajectory to highlight. In Fig. 5, the nominal case is highlighted in red and the dispersed cycle with the highest peak load is highlighted in green.

The user interface allows the analyst to select from the list of available data slices (defined prior to execution). This will display the value of the dependent variable at the slice condition for the highlighted cycle. Statistics for the entire Monte Carlo set at the selected slice are also displayed. An option also exists to add a point on the plot corresponding to the slice point for each trajectory. In Fig. 5, the peak load during the third stage of inflation is shown for each cycle in addition to a subset of the trajectories. The user interface also provides methods for altering the scale of the plot, adding limit lines, and suppressing the display of cycles that may be erroneous or physically impossible.

While the trajectory plots provide a good way to assess the entire trajectory for one or more simulations, the data may need further reduction in order to quickly extract important characteristics of the entire Monte Carlo set. For this reason, MCDS includes two types of statistical plots. The first is a plot of all values of a specific parameter at a given slice condition. Figure 6 shows the statistical plot of third-stage, peak parachute load for all cycles. This is an alternate way to view the data in Fig. 5. The nominal cycle is labeled cycle number zero and is highlighted in red. The cycles with the minimum and maximum values are shown by downward and upward pointing triangles, respectively. The analyst may also highlight any other cycle. The mean and standard deviation of this parameter at this slice are displayed to the analyst (not shown in the figure). The mean and mean plus one-, two-, and three-sigma levels for the parameter are also displayed on the plot. A histogram is provided to assist in interpretation of the sigma levels. MCDS also provides the capability to compute the percent of cases that are above or below a specified value. For example, in Fig. 6 a reference line is drawn at 30,000 lb and the percent of cycles over this value is displayed to the analyst (not shown in the figure). This plot allows easy identification of the extreme cycles and

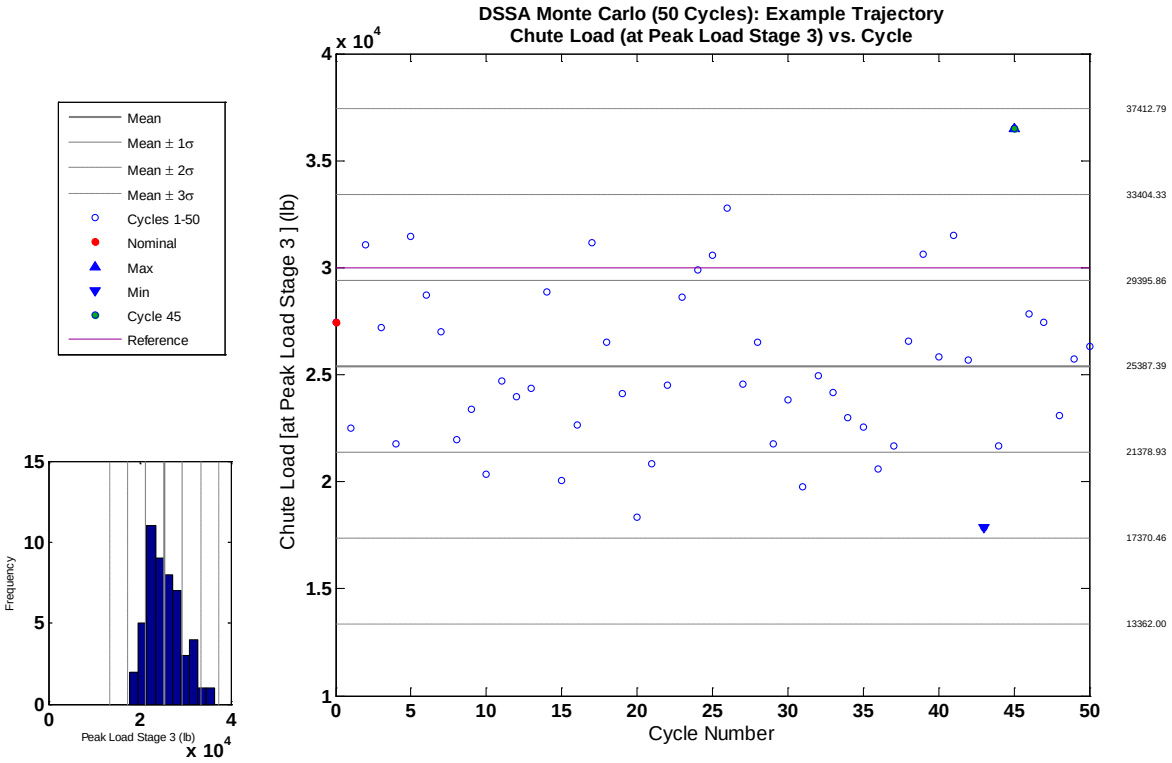

Figure 6. MCDS output statistics plot. *The value of any parameter at any user-specified slice condition may be compared across the dispersed cycles. Here, the peak load observed on the third inflation stage is compared for 50 dispersed cycles.*

helps characterize the spread of the data. The trajectory of the extreme cycle can then be easily examined by highlighting the cycle and then switching back to the trajectory plot view (Fig. 5). In this example, cycle 45 has been highlighted since it achieved the highest peak load for this disreef stage.

Occasionally, the analyst needs to identify the simulation inputs that contribute to an extreme result. The Input Statistics plot (Fig. 7) has proven useful for this purpose. This plot shows the value of a user-selected input variable for all Monte Carlo cycles. Once an input parameter is selected, the cycles that have extreme values (among the dispersed set) for that parameter are displayed (not shown in figure). When a particular cycle is highlighted by the analyst, any input parameters for which that cycle has the extreme value (among the dispersed set) are also displayed. In these example plots, cycle 45 was highlighted in the output statistics (Fig. 6) and trajectory views (Fig. 5). The plot is converted to the input statistics view (Fig. 7) and the over-inflation factor is selected as the input. In the new view, cycle 45 remains highlighted which indicates that the high peak load for this cycle is correlated with a high over-inflation factor. The input statistics plot may also be used to quickly assess whether the spread of dispersed inputs is as expected. The actual maximum and minimum dispersed input values are visible on the plot and displayed to the analyst. The title of the plot indicates how the dispersion rules file was interpreted by MCDS for the variable being plotted. In addition, the user interface provides a method to compare the complete input text file for any two cycles. This is helpful in determining why different cycles exhibit different results.

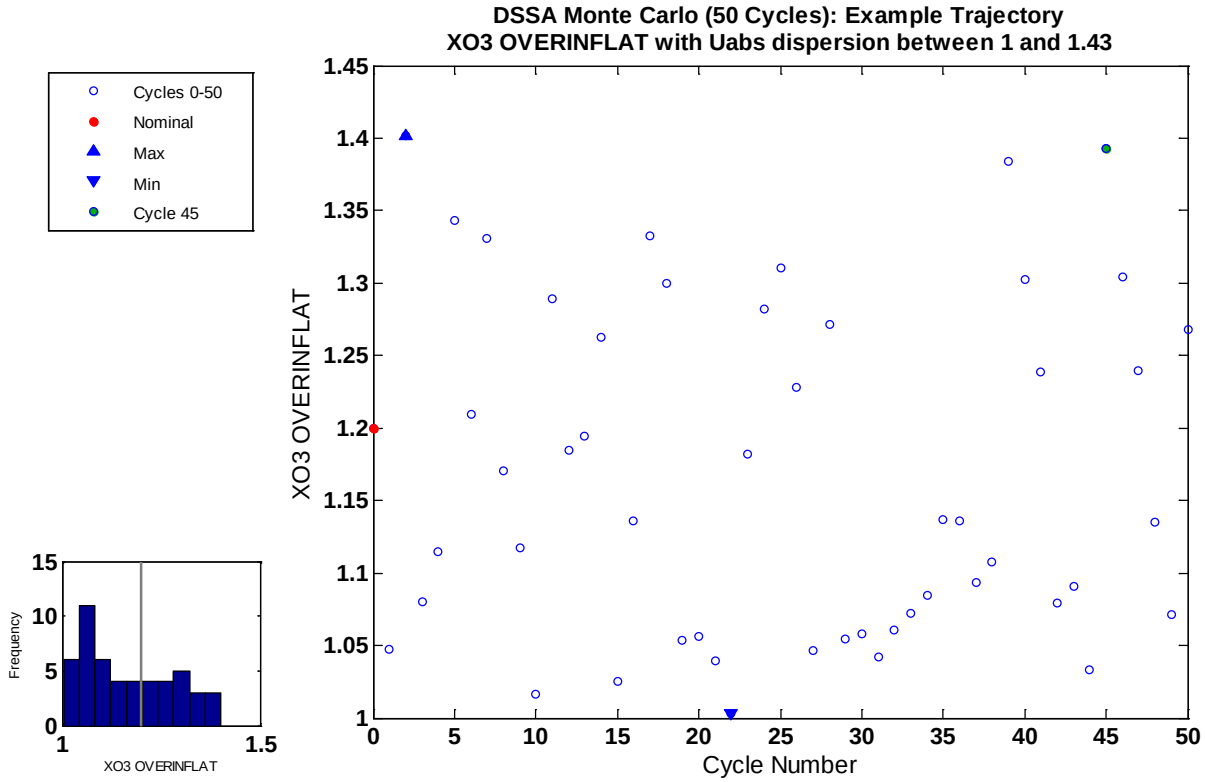

Figure 7. MCDS input statistics plot. *The value of any input parameter may be compared across the dispersed cycles. Here, an over-inflation factor for the third inflation stage is compared for 50 dispersed cycles. Note that the cycle with the highest peak load (highlighted in green) corresponds to one of the highest over-inflation factors.*

The three types of plots mentioned above and the associated slicing functionality and statistical data have proven to be sufficient for most analysis tasks performed by the CPAS simulation and analysis group. In the event of a unique analysis requirement, all the data for all cycles is stored and may be accessed directly using standard MATLAB commands.

V. Limitations and Future Improvements

The MCDS tool has evolved to perform specific analysis needs but has been written to provide generic analysis capability. The reliance on the traditional text-based input formats has had positive and negative effects. On the one hand, it has forced the MCDS code to remain mostly independent of the core-simulation being run, which has simplified extension to new core-simulations. On the other hand, it has required a somewhat complicated dispersion rules file. The analyst must have knowledge of the structure of the nominal DSS or DSSA input file in order to avoid errors in generating the dispersion rules. However, additional offline tools have been written to assist in this process.

A potential weakness of the current MCDS architecture is the serial execution of the dispersed cycles. This simplifies the execution commands and the post-processing of each cycle but increases the amount of time required to perform a Monte Carlo analysis. A single cycle requires five to twenty seconds to execute depending on the core-simulation. To date, most analyses have required only a few hours at most to execute, but future analysis tasks may require a larger number of replications and could extend the execution time to the point where it becomes problematic. In principle, MCDS could be modified for parallel execution without significant changes to the architecture.

MCDS was designed to provide generic analysis capability. For this reason, most of the data generated by each simulation cycle is stored for later evaluation. This allows the analyst to investigate issues noticed in the statistics by examining the trajectory of any parameter in any dispersed cycle without re-executing the Monte Carlo set. One problem with this approach is that it requires a large amount of memory, when in fact, much of the stored data may not be of interest for a particular analysis.

MCDS extracts full time histories for about 30 parameters. A full trajectory may include roughly 10,000 data points. For total recall of a thousand-cycle Monte Carlo run, it would be necessary to keep one-thousand, 30 by 10,000 matrices in memory. This is beyond the capabilities of the machines available for the analyses. In the current configuration, only 20 to 100 cycles are typically maintained in immediate memory for use in the trajectory co-plots.

There are many solutions to this problem. For example, the simulation output could be sampled at a lower frequency, or the tool could store only a few characteristic data from each cycle (i.e. peak load). The current approach used by MCDS is to continue to collect the large data samples but periodically store off the collected data in files and then clear the memory. This allows quick access to any cycle without re-execution of the simulation.

These limitations suggest a few improvements to MCDS that could improve performance and functionality. For very large Monte Carlo sets, it may be desirable to allow the analyst to identify a minimum list of output parameters to process after each cycle execution. This would have the effect of reducing post-processing time and the amount of data carried in memory. It may eventually be desirable to allow direct configuration of the core-simulation input parameters. This would eliminate the need to configure inputs in the traditional simulation environments and would simplify the dispersion process since parsing and modifying an existing input file would not be necessary. A new, common method for configuring input files for the three parachute simulations described above is currently in development.[4] This capability will facilitate direct configuration of the core-simulations within MCDS.

VI. Conclusion

The MCDS Monte Carlo tool described in this paper has been used for trajectory predictions on many of the Generation II CPAS drop tests.[5] It has facilitated multiple Monte Carlo analyses and short turn-around assessments. The generic nature of the data structure and plotting capability has proven to be versatile and has allowed analysts to extract pertinent information from very large data sets without the need to resort to custom analysis codes and scripts. Reliance on the legacy simulations and input/output formats has provided challenges but has also preserved compatibility and allowed analysts to continue to work with familiar tools. The MCDS tool will continue to be refined as new analysis needs develop.

Acknowledgments

The author wishes to thank Kristin Bledsoe, Usbaldo Fraire, Aaron Morris, Leah Olson, and Eric Ray of the CPAS analysis team for their feedback during development of the MCDS tool and for helpful criticism of this paper. The creation of the tool would not have been possible without the expertise on the execution of the core-simulations provided by Peter Cuthbert of NASA-JSC. Peter Schulte of the University of Texas-Austin and Stephanie Shelton of the University of Alabama-Huntsville also provided a much appreciated review of this paper.

References

1. Cuthbert, P.A., "A Software Simulation of Cargo Drop Tests," *17th AIAA Aerodynamic Decelerator Systems Technology Conference*, Monterey, California, May 2003, AIAA Paper 2003-2132.
2. Cuthbert, P.A., Conley, G. L., Desabrais, K. J., "A Desktop Application to Simulate Cargo Drop Tests," *18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, Munich, Germany, May 2005, AIAA Paper 2005-1623.
3. Moog, R.D., et al., "Parachute Simulation User's Guide Computer Program UD233A," *Martin Marietta Corp., Denver Aerospace Division*, Denver, Colorado, USA, Feb. 1986.
4. Moore, J. W., "A Hybrid Parachute Simulation Environment for the Orion Parachute Development Project", *21st AIAA Aerodynamics Decelerator Systems Technology Conference*, Dublin, Ireland, May 2011 (submitted for publication)
5. Morris, A.,et al., "Summary of Generation II CPAS Parachute Performance", *21st AIAA Aerodynamics Decelerator Systems Technology Conference*, Dublin, Ireland, May 2011 (submitted for publication)
6. Morris, A.,et al., "Summary of CPAS Test Techniques", *21st AIAA Aerodynamics Decelerator Systems Technology Conference*, Dublin, Ireland, May 2011 (submitted for publication)
7. Ray, E.S., "Challenges of CPAS Flight Testing", *21st AIAA Aerodynamics Decelerator Systems Technology Conference*, Dublin, Ireland, May 2011 (submitted for publication)
8. Taylor, A. P., Murphy, E., "The DCLDYN Parachute Inflation and Trajectory Analysis Tool – An Overview", *18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, Munich, Germany, May 2005, AIAA Paper 2005-1624.