Testing Improvements to ADAM’s Behavioral Design Pattern & Conversion of Modeling Capabilities from MATLAB to C++

Dr. Steven Fiorino1, Brannon Elmore1, Joel Mecak1
1Department of Engineering Physics, 2Cedarville University, 3Applied Research Solutions

Abstract

The DoD and military research departments use the AFIT Directed Energy and Atmospheric Models (ADAM) to model high-energy laser systems and engagements for research purposes. While version 4 of ADAM is a validated MATLAB software that accomplishes its purpose, it can be made to run faster with less platform dependency. To accomplish this, the LEEDR (Laser Environmental Effect Definition and Reference) portion of the MATLAB software began conversion to C++ in the summer of 2017 and has now in beta testing. This new version is more modular and runs much more quickly than its MATLAB counterpart. However, it still needs a module that will provide quick validation by comparing C++ outputs with MATLAB outputs.

Methodology

Because C++ is a lower-level language, it is realistic to expect the programmers to make mistakes when converting the tens of thousands of lines of MATLAB code to C++. To mitigate this fear and reduce the number of mistakes, the developers designed many tests in the form of “Test Suites”—a place where ADAM can be tested. When these tests run, they compare hundreds of thousands of outputs from MATLAB to the same outputs from C++. Obviously, given the same inputs, the two programs should produce the same outputs. Each module of C++ ADAM is thus tested to provide the same validation as its MATLAB counterpart.

Motivation

A C++ version of ADAM allows for major improvements over MATLAB. C++ is faster, more modular, and more accessible than MATLAB. Though MATLAB was developed with optimized matrix math in mind, it is still a higher-level language and suffers severely as the size of the program increases. C++ on the other hand is a lower-level language. It allows the programmer to control many more aspects than MATLAB. While this extra control is useful for optimizing run time, it also means that the programmer must consider more facets that perhaps they could have ignored when using MATLAB. Even without considering the lower level aspects, C++ is fundamentally faster since it is compiled into an executable that runs directly on the hardware instead of relying on the JIT compiler that slows programs down significantly in MATLAB. Another way to increase the speed is to use caution when writing functions that run calculations. Rerunning all calculations when merely one input changes is a waste of computing resources. Only affected outputs should be rerun. Developers made these kinds of changes in both the MATLAB and the C++ version of ADAM.

Conclusion

ADAM is made up of three smaller programs which are abbreviated LEEDR, HELEEOS, and HELTDA. The C++ version of the LEEDR component of ADAM is nearly complete. It is set to release on the 29th of July 2019. After its release, LEEDR will continue to see updates in parallel with its MATLAB counterpart. Furthermore, developers will begin porting the HELEEOS (High Energy Laser End-to-End Operational Simulation) module of ADAM to C++ as well. With the new C++ version of the ADAM software, researchers in the DoD and military will be able to accomplish even more in their fields of research.

Acknowledgements

Thanks to:
Joel Mecak for his mentorship and willingness to explain LEEDR
Brannon Elmore for his guidance and instruction

Figure 1. Old Ctrtnm vs the new Ctrtnm. While the change is subtle, it has a significant effect on the output.

Figure 2. This graph compares computation time between C++, MATLAB, and other languages on a log scale. C++ is faster than MATLAB by a considerable margin.

Figure 3 (left) A portion of the path radiance controller. When path radiance runs, it provides information about what an observer would see from any given perspective provided a set of weather conditions.

Figure 4 (right) A portion of the “release test suite”. It runs a subset of all tests and will provide a satisfactory demonstration that the program is functional.

Figure 5. (left) Output from one of the molecular test suites showing the comparison between the C++ output, the MATLAB output, and the percent error between the two.

Contact Information: Joshua C. Tarwater
jctarwater@cedarville.edu