Symbolic Algebra and Visualization Enhancements of a 2D-Visual Programming Environment for Multiphysics Mechatronic Systems Simulation

Panayiotis Steele
2005-2006
Mentoring Firm - NRL

Background

For some time now, there has been a need for a symbolic algebra module for Ptolemy II. Ptolemy II is a programming environment that contains VERGIL, a two-dimensional visual programming interface. This application is being developed in the Center for Hybrid and Embedded Software Systems (CHESS) in the Department of Electrical Engineering and Computer Sciences (EECS) of the University of California at Berkeley. It is used mainly for heterogeneous and concurrent modeling and simulation of physical systems.
Currently, the Ptolemy II program is limited to simple algebraic manipulations and up to three-dimensional representation and visualization of bodies. With the development of a module with symbolic algebra capabilities, the capabilities of Ptolemy will be greatly expanded.
Users will be able to model physical systems governed by complex physical rules.

Abstract

Development of a symbolic algebra module for a two-dimensional visual programming and modeling environment would greatly increase the productivity and efficiency of research dealing with composite materials at NRL. The work under this effort developed such a system. With this new technology, a researcher can enhance the ability of conducting complex virtual experiments remotely. This new program would greatly reduce the need for uncertain and expensive system design and prototyping.

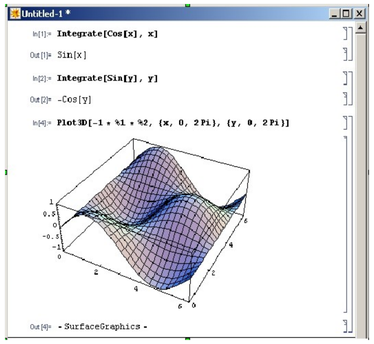


Figure 1: Mathematica allows users to perform tasks such as plot three-dimensional graphs of functions, along with many other mathematical tasks.

Commercial Software Used

Ptolemy II is the next generation of heterogeneous simulation and design software. It implements and extends the functionality of Ptolemy Classic, its predecessor, which was also developed at CHESS at the EECS Department of Berkeley. In addition to Ptolemy Classic¿s models of computation, Ptolemy II supplies new models of computation and a more user-friendly graphical user interface (GUI). The Ptolemy Project studies modeling, simulation, and design of concurrent, real-time, embedded systems, with a focus on the assembly of concurrent components.

VERGIL, the GUI of Ptolemy, is where programmers can design systems. By choosing from a wide variety of execution modes, almost any kind of system can be developed. Inside VERGIL, computation models are termed "directors" and modules "actors." By connecting actors¿ input and output ports, an efficient and easy-to-understand model of almost any system can be created.

Mathematica is the dominant program of mathematical computation, and has been ever since its first release in 1988. At first it was mainly used for physics, engineering, and mathematics. But over time, Mathematica has made a great impact on a wide variety of disciplines. It is now used extensively throughout the sciences--biological, social, and physical, for example--and has played a key role in many important discoveries. It is widely regarded as a great feat of engineering. Mathematica is one of the largest standalone applications ever created. Its enormous collection of unique algorithms and technical innovations provide it with enough versatility to solve almost any problem. Engineers rely on Mathematica as a standard tool for development as well as production.

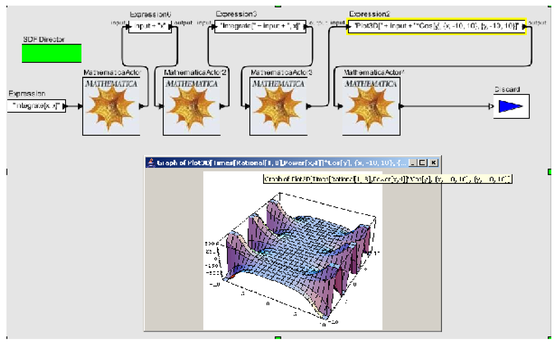


Fig. 2: A simple model demonstrating the how the graph of a 3D function is handled in the MathematicaActor.

Significance of the Project

In the Computational Multiphysics Systems Laboratory at NRL (CMSL), there is a complex mechatronic system named the 6D-Loader. It is termed mechatronic because it incorporates mechanical engineering with electronics, computer systems, and advanced controls. The system has six degrees of kinematic freedom for applying displacements and rotations to certain selected composite material specimens. It also measures the specimens¿ reaction forces and moments that are a consequence of the forces applied on the specimens.

However, the machine's kinematic behavior is described by a very complex system of non-linear equations. The behavior of any given specimen is governed by a coupled system of partial differential equations. Due to their complex nature, these equations are not solvable by hand. A computer algebra system, such as Mathematica, is essential to obtaining a solution to either of these systems of equations. Ptolemy II, along with its 2D visual programming editor, Vergil, will be used to model both the visual and functional behavior of the machine.

The completion of the symbolic algebra module has opened new realms of possibility for the CMSL. The 6D-Loader is a complex and often cumbersome machine to operate; thus it is only used for experiments once a year at most. Ptolemy II's integration with Mathematica, in the form of the module, allows the researchers at the CMSL to model the 6D-Loader¿s visual and functional behavior, along with the visual and functional behavior of the composite material specimens. When the models of both 6D-Loader and specimens are finished, a researcher will be able to simulate an experiment remotely. Essentially no preparation will be required to run such a remote experiment simulation.

Development

In order to develop the module, I needed to find a way to connect to the Mathematica kernel in a Java program. Since Ptolemy II is written entirely in Java, programmers wishing to extend it must program their modules in Java. The way presented itself through a library for Java programmers now distributed with Mathematica, J/Link. J/Link is the stipulated method to connect to the Mathematica kernel externally, i.e. outside of the Mathematica front end. It uses the previous interface, MathLink¿a library for C and C++ programmers¿and creates numerous higher-level methods for accessing the kernel, which MathLink did not have. J/Link has dual compatibility. For Mathematica programmers, it extends the Mathematica programming language, allowing them to use Java classes in their Mathematica programs. For Java programmers, it allows Mathematica to be used as an interactive shell for testing Java classes and applications one line at a time, and also use the Mathematica kernel as a computational engine¿in the background for Java programs needing a complex mathematical computational engine.

Programming the module for Ptolemy II fell into the last category, so I consulted the J/Link documentation for information about how to connect to the Mathematica kernel. First, a standalone application was developed to test the algorithm developed to access the kernel. Then, using the code conventions for creating actors specified by the Ptolemy Project, the algorithm used in the standalone application was used in the three classes--MathematicaActor, ExprEvaluator, and JLinkGfxPanel--that comprised the actor. This actor could connect to the Mathematica kernel, accept a valid Mathematica expression presented to its input port in String form, send it to Mathematica, and return the result. If the result was some kind of graphics, such as the graph of a function, then the actor displayed the result in a pop-up JFrame (see Fig. 2).

However, to model the behavior of the machine, the actor needed to also be able to accept Mathematica programs, in the form of text files, as input. To solve this problem, another actor, ProgramReader, was created. This actor took a specified text file, converted it into one line, and sent the resulting String object to its output port. When connected to a MathematicaActor, the ProgramReader allows the reading in and execution of Mathematica programs (see Fig. 3).

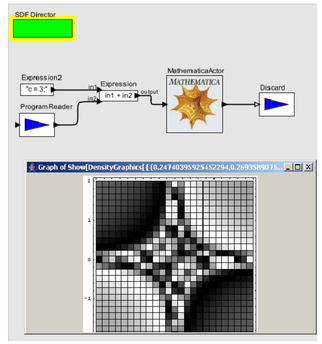
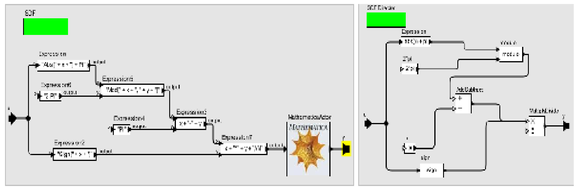


Fig. 3: A model demonstrating the loading and execution of a Mathematica program the computes a density plot.

Code Validation

Any Ptolemy model that uses mathematics can use a MathematicaActor. For instance, the Bouncing Ball model can be easily simulated using a MathematicaActor in place of the built-in Ptolemy II mathematics. Another more complicated example is the angle converter in the Furuta Pendulum example (from the Ptolemy II Tour page). It can be just as easily implemented with Mathematica commands (see Fig. 4) as with the Ptolemy II expression language. The only difference in the model's execution is the speed of the model; this issue, as of yet, has an unknown cause.



Fig, 4: Two ways of implementing the angle converter for the Furuta Pendulum.

Future Improvements and Conclusions

This project can still be extended. For improved visualization of three-dimensional graphs, the module can be integrated with a software package named JavaView. JavaView allows the user to rotate, translate, and resize Java3D images. It is already integrated with Mathematica, so it would not be much of an endeavor to integrate JavaView with Ptolemy II.

Another possible augmentation of this project is using the same Mathematica kernel for the whole execution of the program
. Currently, a new kernel is opened and closed every time a computation is requested in a model containing a MathematicaActor. This method of execution is not desirable, as it increases greatly the time that is required to run a model that loops around a MathematicaActor, or to run a model that contains several MathematicaActors. Keeping the original kernel open for the whole duration of such a model's execution would decrease the CPU cycles needed to perform the computations.

Finally, there is occasionally a need to explicitly specify in Vergil what kind of output and/or input the MathematicaActor and other modules in the same model will be receiving or sending, most notably when ProgramReader is part of the model. The cause of this need for explicit port typecasting is not known; further investigation must be done.

Despite the limitations of the project in its current state, it will still be of great use in the CMSL. We hope that, eventually, the MathematicaActor and its accompanying programs will be included in future releases of Ptolemy II, and be useful to Ptolemy users around the world.