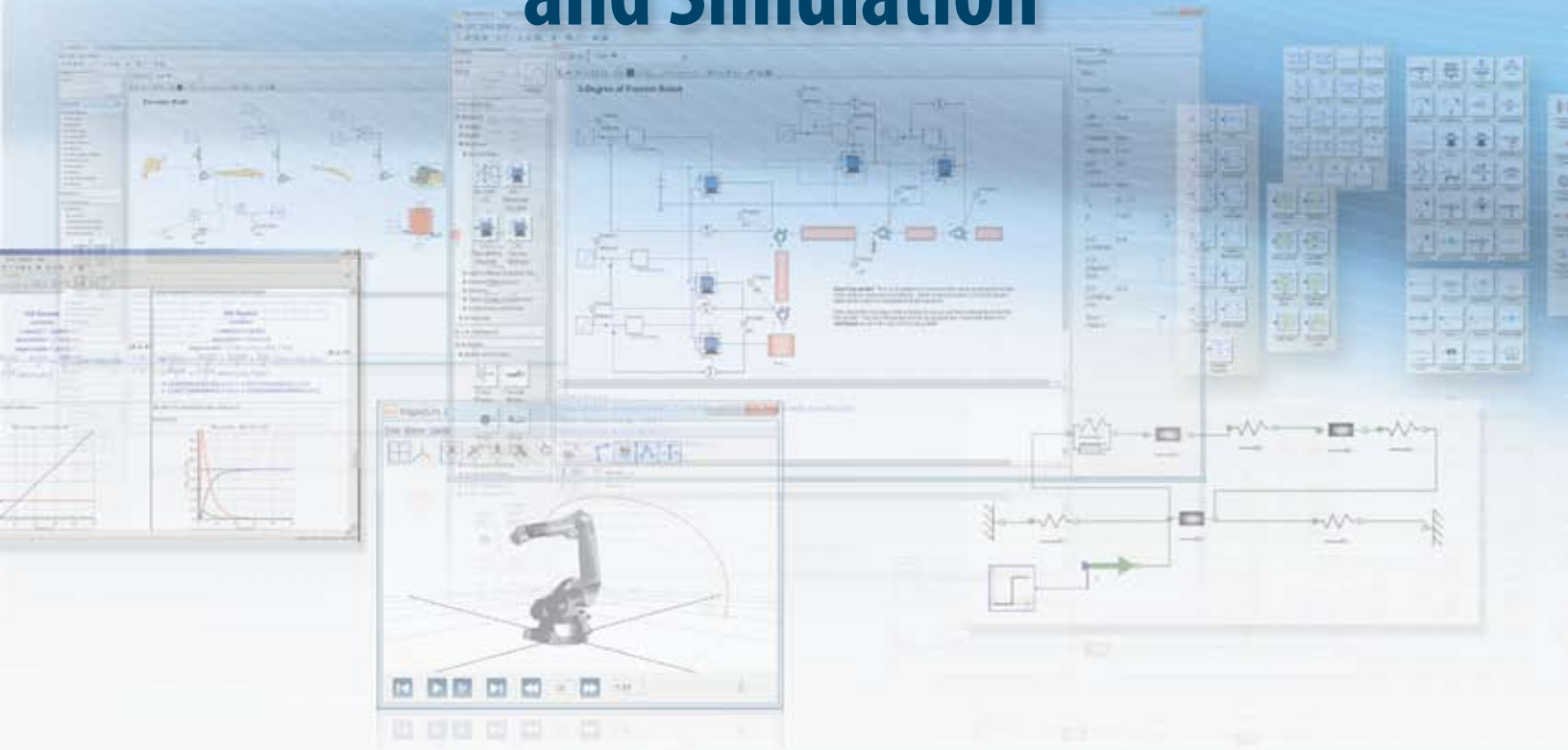


MapleSim: Technological Superiority in Multi-Domain Physical Modeling and Simulation



Abstract

Engineers constantly demand innovation in modeling and simulation. This is driven by the need to simulate, for example, ever more sophisticated vehicle systems while minimizing development time and cost. The computer models necessary to generate these designs consume project time at an increasing rate, due largely to the limitations of existing simulation tools and the need to switch between tools to address multiple physical domains. While many existing simulation tools are based on a design metaphor that works well for control system design, engineers do not find them intuitive for physical modeling. Additionally, these tools often produce simulations that are too slow to model complex systems and, in some cases, lack the mathematical power to solve the governing equations.

These are some of the issues that have driven the development of MapleSim, a new multi-domain simulation tool from Maplesoft. This article will explore the historical development of the early tools for simulation, and then expand on their design deficiencies. This is followed by a discussion of MapleSim, its unique design characteristics, and how it bypasses the limitations of current modeling and simulation technology.

MapleSim: Technological Superiority in Multi-Domain Physical Modeling and Simulation

Introduction

Model-based development (MBD) involves developing high-fidelity models of engineering systems and then simulating them to understand how they will function under operating conditions, identifying potential problems, and developing solutions before going to the prototyping stage. Not only does this process give engineers a much better starting point when they begin creating prototypes, it can reduce the number of prototyping cycles. Fewer cycles reduce the time-to-production significantly, and save hundreds-of-thousands, perhaps even millions, of dollars in prototyping costs.

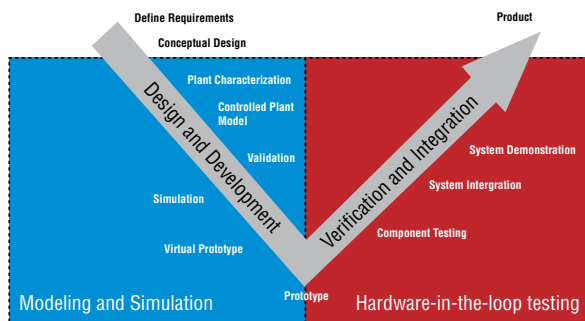


Figure 1. System development process

Over the last 15 years, MBD has evolved into a widely adopted process. The process is typically represented as a V (Figure 1). The right side of the V is characterized by rigorous testing programs, often using models that have been produced in the Design and Development phases and running them in real time for Hardware-in-the-Loop (HIL) testing. Eventually, this process leads to the complete integration of the system that is ultimately released as part of the product.

The left side of the V encapsulates the use of simulation software for virtual prototyping, where many of the design issues can be identified and addressed on desktop computers before the first prototype is built. While many of the software tools used for this process typically address numeric simulation, they do not provide an environment that engineers find intuitive for physical modeling. This is because they are based on a signal-flow metaphor that has changed little from the days of the analog computer.

Modeling software that bypasses the limitations of signal-flow tools has emerged, but inherent restrictions in their technology and scope prevent them from being more widely adopted. These deficiencies include:

- Modeling functionality that may be limited to a single physical domain

- Numerical solvers that cannot adequately address the math behind increasingly complex systems
- No support for design documentation

These limitations, and a growing market demand, have motivated the development of MapleSim, a tool for physical modeling and control systems development.

A Brief History of Modeling and Simulation

The Birth of Analog Computers

The earliest attempts at modeling physical systems were called “analog”; that is, the use of mechanisms that behave the same or similarly as the system under study. These “analogous” mechanisms could be used to predict the behavior of a wide range of systems, such as Kelvin’s 1879 tide predictor (Figure 2). This was essentially a set of pulleys, cranks, and ropes that emulated the motion of the seas, given certain properties about the location under study. The tide predictor was followed by more sophisticated mechanisms such as masses, springs, and dashpots that could be configured to model the dynamics of systems. Then came electronic components, such as inductors, capacitors, and resistors that could emulate the same behavior but at a lower cost and with better precision.

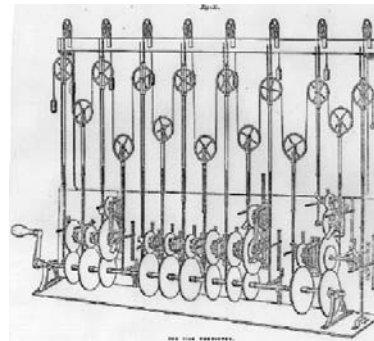


Figure 2. Kelvin's tide predictor, 1879

Those components, along with the invention of the operational amplifier, gave rise to the analog computer in the 1940s. This provided scientists and engineers with a set of tools that allowed signals to be added, multiplied, and integrated – the three primary operations required to solve systems of differential equations.

The analog-computation method for solving ordinary differential equations (ODEs) involved rearranging the equations into integral form and simulating them with integration circuits with a combination of resistors, capacitors, inductors, and op-amps. By attaching a pen recorder or voltage gauge on an output line, it was possible to arrive at the predicted behavior of highly complex systems (Figure 3). For example, a network of operational amplifiers and linear passive components would be sufficient to simulate a mass-spring-damper, while the concept of current and voltage would be used to simulate pressure drop and flow rate of water through a pipe.

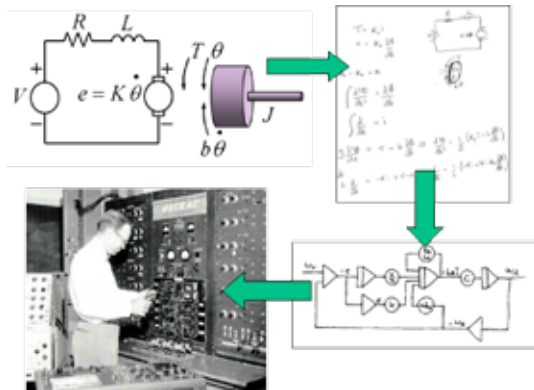


Figure 3. Model development process for analog computers

While electronic analog computers were a major breakthrough, they were expensive tools available only to a select few.

Signal-Flow Block Diagram Tools: A Legacy of the Analog Computer

With the birth of the digital computer in the 1950s, a wide range of numeric solvers that emulated the analog computational approach were developed, but with a wider breadth of application. While expanding the scope of dynamic system simulation significantly, these were essentially specialized programming languages that required highly specialized skills and digital computers that were still very costly.

Eventually, with the introduction of low-cost personal computers and graphical user interfaces, the 1980s saw the birth of the block-diagram environment that allowed a much larger number of engineers to develop dynamic system models on their own desktops.

The block-diagram environment was effectively a virtual analog computer on a digital computer with the latest numeric solvers (Figure 4). For the first time, engineers could produce models on their desktops by “wiring” together block components that represented the components their

predecessors would have done with electrical components on real analog computers, but at a fraction of the cost.

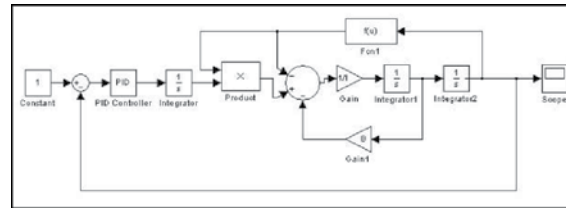


Figure 4. Typical block-diagram GUI

Consequently, this revolutionized the way engineering simulations were performed and several tools of this nature are used extensively today. By far, the most popular of these is Simulink[®] from The MathWorks[™], but there are also SystemBuild[™] (National Instruments[®]), EASY5[™] (MSC Software[®]), and VisSim[™] (Visual Solutions), among others.

Signal-flow tools were rapidly adopted because of their relatively low cost and highly intuitive user interface. Most engineers would have taken a controls course in their undergraduate studies and part of that course was to learn how to draw schematic block diagrams to represent dynamic systems and controllers. Hence, they were already familiar with some of the principles involved.

Limitations of the Signal-Flow Modeling Paradigm

No Support for Automated Plant Modeling

The signal-flow approach is a laborious process that has not changed in over 50 years:

- 1) The ODEs that describe the system dynamics must be derived prior to using the signal-flow tool, usually by hand based on the engineer’s knowledge of the underlying physics.
- 2) The ODEs are rearranged into a series of integral equations.
- 3) The integral equations are entered in block diagram format on a computer.

This is still a manual process that is time-consuming, costly, and requires intense human effort and a level of skill and knowledge that is not always available.

Fixed Causality

The signal-flow paradigm forces model representations into a fixed causality, meaning that a component can only be represented one way. For example, consider a resistor whose characteristic equation is

$$V = I \times R,$$

where V , I , and R are the voltage, current, and resistance, respectively. The voltage can only be calculated if the resistance and current are known. However, if the resistor is driven by a voltage source and we want to calculate the current, then a different representation is needed:

$$I = \frac{V}{R}.$$

Therefore, to fully represent a component for use in different ways, all possible permutations of the governing equations are required. While this concept is simple for a resistor, the implications for complex components with multiple inputs and multiple outputs are far greater.

This makes the creation of generalized component libraries very difficult and forces engineers to choose the correct form of the equation, depending on the inputs, prior to building the model.

In essence, these legacy computer tools are causal in nature; each block has a distinct cause and effect. A signal flows into a block, the block performs a mathematical operation on that signal, and the result flows out of the other side. This is illustrated schematically in Figure 5.

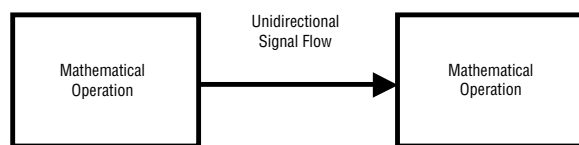


Figure 5. The causal approach to block diagram programming

This approach is useful for modeling systems that are purely described by signals. These include control systems, digital filters, and other signal processing systems.

It is not, however, intuitive to engineers when describing systems such as an automotive drive train, which consists of a network of interacting physical components. This requires a completely different modeling paradigm.

Algebraic Loops

Dynamic system models use one or more variables that vary with time; these are called states and they essentially define the dynamics of the system. At any simulation step, signal-flow tools use the computed value of each state variable from the previous step to compute the current value. Therefore, all state variables must have initial values so that the values in the first step can be calculated.

However, there are many situations where the user can define a loop that does not have state. A very simple example is given in Figure 6.

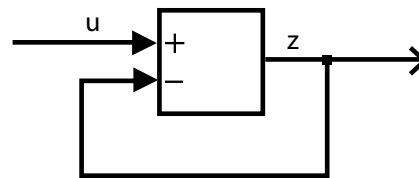


Figure 6. A simple algebraic loop

Mathematically, this represents

$$z = u - z,$$

which implies that the output value is directly related to the input. Of course, there is a simple solution,

$$z = \frac{u}{2},$$

but standard signal-flow tools cannot reach this solution algebraically. (Moreover, algebraic loops are usually much more complex than this.)

Signal-flow tools have to resort to iterative numerical solvers to resolve algebraic loops. This process cycles through the loop until the input equals the output. This requires intervention by the user to implement and puts considerable delay in the simulation because it must go through many iterations, calculating a solution at every time step.

Also, in some complex models, the solution can be very sensitive to the iterative solver and numeric tolerances used, and can easily become unstable.

The root of the difficulty is in the composition of systems. Even if subsystems are described by ODEs, interconnected systems may not be described by an ODE because interconnections may create algebraic constraints. Mathematically, these types of systems are defining Differential Algebraic Equations (DAEs).

Differential Algebraic Equations

DAEs are very common in engineering problems and are the basis for solving complex systems that are typically represented as networks of components: electrical circuits, hydraulics, pneumatics, and mechanical systems, to name a few. The nature of these representations introduces algebraic constraints, with the drawbacks illustrated previously.

Consequently, tools that use physical components (and not causal signal-flow blocks) to model the system have emerged. These tools usually feature a DAE solver, but they tend to be domain-specific. For example, several very popular electric circuit modeling tools have emerged based on the SPICE representation. These tools feature highly specialized DAE solvers.

The need for a more generalized approach to modeling networked physical components has led to several, often competing, approaches based on graph theory. Two major approaches are Bond Graphs and Linear Graphs. Each has its strengths and weaknesses, but emerging products typically adopt either one or the other.

Probably the most difficult domain for which to provide a generalized solver is in multibody mechanical systems. Several approaches have emerged, including Newton's equations, Lagrange's equations, Kane's method, and Blajer's projection method. The common theme is that they all use graph-theoretic methods for defining the system topology and produce DAEs to represent the system mathematically.

Simplistically, DAEs are sets of ODEs that must also satisfy algebraic equations that have been introduced through the addition of physical constraints in the system. For example, a simple RC circuit can be modeled with an ODE, but the addition of a parallel capacitance introduces an algebraic constraint that must be satisfied before the ODE can be solved.

Further constraints add complexity to the DAE representation. Depending on the nature of the constraint, complexity is indicated by an increase in the index of the DAE. The index is a count of the number of times the DAE needs to be differentiated in order to reduce the problem to an ODE, at which point it can be solved using standard differential equation solvers.

Differentiation, however, magnifies numeric errors, and therefore multiple differentiations will quickly render solutions unusable unless techniques are used to reduce the errors. Current index reduction techniques attempt to reduce the errors, but they are generally only useful for up to index 3. This is very limiting, since a generalized approach to physical modeling can easily produce DAEs of index 3 or greater.

For example, even a simple pendulum resolves to an index 3 problem if absolute Cartesian coordinates are used. If the angular position of the pendulum is selected as the generalized coordinate, the index is decreased to 0; that is, an ODE. This is illustrated in Figure 7.

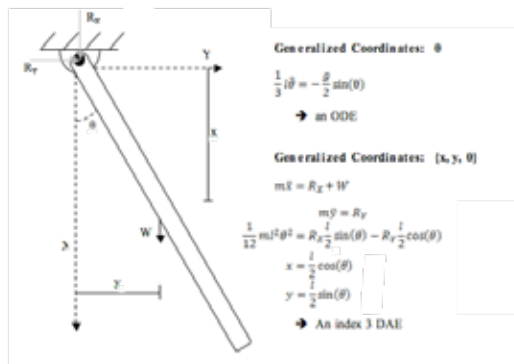


Figure 7. Importance of coordinate selection when modeling a pendulum

MapleSim Design Features

Modeling at the Physical Component Level

MapleSim allows engineers to use both causal and acausal modeling paradigms. With acausal modeling, each block is a discrete physical component, such as a resistor or a mechanical joint. Each block contains information about which physical laws it must obey, and two connected components exchange information about which physical quantities (such as energy, current, torque, heat, and mass flow, etc.) must be conserved. Figure 8 gives a schematic of the process. (This should be compared to the corresponding causal diagram in Figure 5.)

MapleSim provides a broad range of acausal physical components across several physical domains; the equations that define their behavior do not have to be derived or entered manually. Engineers can also create their own components; these are generated from the differential equations, algebraic expressions, or transfer functions that define the dynamics of the component.

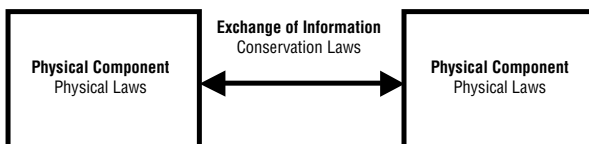


Figure 8. The acausal approach to block diagram programming

Acausal modeling better represents how physical components interact; this has two important corollaries:

- MapleSim block diagrams bear a far closer visual resemblance to the real system than models created by tools that use the causal, or signal-flow, approach. This is emphasized in Figure 9, which compares the causal and acausal approach to modeling a double mass-spring-damper.
- MapleSim is easier to use and features a shorter learning curve than signal-flow tools. This is because its use is not predicated on the ability to derive system equations, but on the inherent talent of engineers to visualize how physical components connect and interact in real systems.

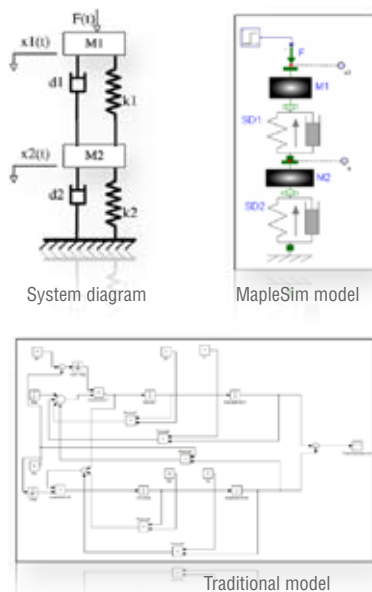


Figure 9. A comparison of the causal and acausal approach to physical modeling

Hybrid Physical and Signal-Flow Modeling

MapleSim allows both modeling at the physical component and signal-flow level within the same model. This enables both the plant model and the control system to be prototyped with the modeling approach that suits each task best.

An example of a hybrid system is given in Figure 10. It is a model of a DC motor with an associated control system. Note that both electrical and mechanical components are integrated in the same model.

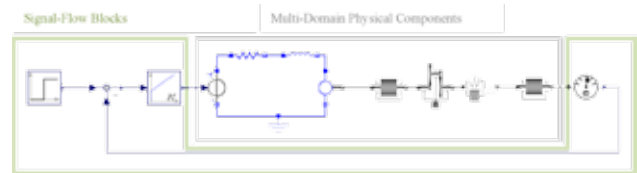


Figure 10. Mixed signal-flow blocks and multi-domain physical components

Multi-Domain Modeling

Many physical modeling tools (such as those based on SPICE) are dedicated to a single domain. This is highly restrictive because engineers often need to model the subtle effects that one domain has on another. For example, given the appropriate set of operating conditions, there is a high correlation between the variation of the voltage across a motor and the vibration of an attached mechanical arm. That critical relationship would be far more difficult to discover (and to correct, if needed) if the electrical motor and the mechanical system were simulated in separate systems.

MapleSim is naturally multi-domain, and contains components from rotational and translational mechanics; analog, digital, and multiphase electric circuits; multibody mechanics; and thermodynamics. It also has standard continuous and discrete functionality.

“One of the most exciting aspects of the MapleSim project is the opportunity to work on the modeling process from the beginning where the system is specified by high level description in the form of a component diagram, to a set of mathematical equations representing the model of the system, and finally to the simulation results. Each stage of the process presents a different set of challenges. From the translation of the graphical representation to the mathematical equations, the challenge is in the generation of equations. From equations to simulation, the challenge is in the manipulation of equations, as well as in the simplification of the model itself. And, of course, simulation is just one of the features offered by MapleSim. The integration of MapleSim and Maple provides a natural extension that allows users to perform advanced analysis on their MapleSim system model and to create high impact technical documentation. This makes MapleSim a very powerful, and yet flexible, integrated tool for modeling, simulation, and analysis of everyday physical dynamic systems.”

Dr. Gilbert Lai
Lead Developer, Simulation Engine Library

MapleSim will not allow connections where they are not physically meaningful; an inductor, for example, cannot be coupled directly with a rotational inertia component.

A simple MapleSim multi-domain model is given in Figure 11, in which the resistor in a DC motor produces heat that varies with the current passing through it. The heat drains into a heat sink whose temperature varies with the heat produced by the resistor. This model combines three physical domains: electrical, 1-D mechanical rotational, and thermal.

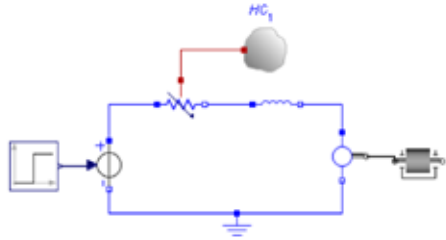


Figure 11. A simple multi-domain model

Design Documentation and Model Analysis

MapleSim is supported by Maple's computational Engine. Maple is a software tool that helps technical professionals do math on a computer. It has a broad range of numeric and symbolic math functions, together with tools for visualization and deployment of solutions. A Maple worksheet is both an executable statement and documentation of the problem domain; ideas and assumptions can be recorded in the same environment as the math and algorithms.

A diagram of Maple features and how they map onto the engineering problem solving process is given in Figure 12, while Figure 13 shows an example of a Maple document.

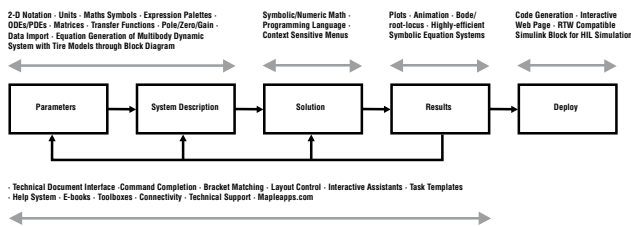


Figure 12. Maple and the engineering problem solving process

Maple provides the symbolic and numeric routines that MapleSim uses to simplify and solve system models. These algorithms are the result of over 25 years of continual investment in research and design.

Maple worksheets can also be used to analyze and document MapleSim models. Through an interactive link, a Maple worksheet can be used to extract, view, and manipulate the symbolic equations that are created by a MapleSim model (see Figure 13).

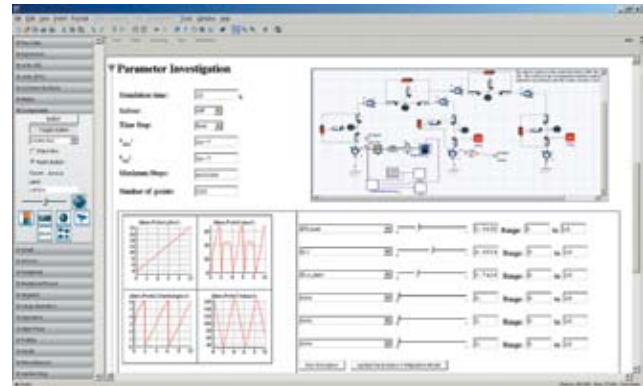


Figure 13. The Maple environment

Worksheet templates are provided for plotting Bode and root-locus diagrams, optimization, generating C code, and creating custom components. In particular, these templates can be used to find the controller gains that minimize rise time and attenuate overshoot, and create custom components from algebraic expressions, differential equations, and transfer functions.

These worksheets can be then electronically attached to MapleSim models. This means that both the model and the design documentation are kept in a single location, ensuring that the origin of design parameters can be audited and traced.

Symbolic Simplification and Computational Efficiency

After automatically generating the system equations, MapleSim simplifies them with symbolic techniques that include index reduction, differential elimination, separation of independent systems, and elimination of redundant systems. Symbolic simplification has two primary benefits:

- By symbolically resolving algebraic loops and through reducing the complexity of DAEs, symbolic simplification makes many (previously intractable) problems numerically solvable.
- The simplified equations are provided to the numerical solvers in a computationally efficient form. This reduces the total simulation time, in some cases, by many orders of magnitude.

Computational efficiency is particularly important for HIL studies because it allows an engineer to develop higher-fidelity models while maintaining real-time performance.

As an example, engineers recently used MapleSim to develop a full-vehicle multibody model of a Chevy Equinox including pneumatic tires (22 degrees-of-freedom and 26 state-space variables) and export the model to a dSpace simulator (using third-party tools). Even a modest performance simulator (a 1 GHz PowerPC) achieved update rates of 63 μ s. This was at least 16 times faster than the most popular existing tools, thus emphasizing the fact that MapleSim is better suited for real-time applications.

Multibody Dynamics

MapleSim uses graph-theoretic methods to generate the equations of motion for multibody systems. One feature of this approach is that it allows for the systematic separation of terminal equations (which describe how individual components interact) from topological equations (which describe how components are connected).

This is significant because by managing topological equations intelligently, MapleSim's multibody engine can directly control the state variables for a given system. Since the nature and number of the equations of motion are a direct result of the chosen state variables, controlling the state variables gives the multibody engine unprecedented control over equation complexity during the equation formulation process. This delivers computationally efficient sets of symbolic equations that are simplified further by Maple, and delivers faster simulations than approaches restricted to only using absolute coordinates as state variables. Benchmarks have shown much higher execution speeds than the most popular existing multibody tools.

The significance of selecting state variables and coordinate systems intelligently is evident even when modeling a system as simple as a pendulum (see Figure 7). (The other choices of coordinate systems can result in an index 3 DAE.)

Conclusion

This article has demonstrated how MapleSim addresses the limitations of current modeling and simulation technology. MapleSim does this by:

- Allowing engineers to use both acausal and causal modeling paradigms in the same model, so plant models and control systems can be prototyped in a manner that suits each task best
- Using symbolic technology to remove the complexities that would cause other tools to fail during numerical simulation, and to create computationally efficient models
- Offering multibody technology that produces highly efficient descriptions of even the most complex 2-D and 3-D multibody systems
- Supporting design documentation and engineering analysis through integration with Maple
- Allowing natural multi-domain modeling that enables the interaction between many physical domains to be simulated

The development of MapleSim was, in part, motivated by a growing market demand for tools that can successfully model the increasingly complex systems that engineers need to simulate. MapleSim was built from a basis of technological superiority, and a development plan is in place to maintain this position of strength.

"As an engineer, I want to be able to rapidly construct my initial model and have the software automatically handle peripheral design decisions. However, once the process is further along, I want to be able to manage those details in order to optimize the system I'm modeling - and not be limited to the software's generic algorithms. One of the challenges in implementing the coordinate selection algorithms for the multibody engine was finding this balance between automation and user-control. Linking preliminary coordinate selection to initial conditions specification allows for models to be initially set-up in an intuitive manner—coordinate selection "just happens"—without any explicit action on the part of the user. However, this same mechanism allows advanced users to leverage their insight into the model to generate more efficient models."

Dr. Chad Schmitke

Lead Developer, Multibody Dynamics Library

The range of analytical products available from Maplesoft serves to complement the existing tools by providing a highly flexible, easy-to-use environment in which the engineer can explore ideas and concepts way beyond the inherent limits of more traditional tools.

Products used in this article

Maple™

Maple™ is the essential technical computing software for today's engineers and scientists. Whether you need to do quick calculations, develop design sheets, or produce sophisticated high-fidelity simulation models, Maple provides the necessary technology to reduce errors and dramatically increase your analytical productivity.

Maple combines the world's most powerful mathematical computation engine with an intuitive user interface. Its smart document environment automatically captures all of your technical knowledge in an electronic form that seamlessly integrates calculations, explanatory text and math, graphics, images, sound, and more.

www.maplesoft.com/products/maple



MapleSim™ is a high-performance multi-domain modeling and simulation tool that will revolutionize how you bring new products to market. In MapleSim, the world's most advanced symbolic computing engine comes together with traditional numeric solvers to supercharge the simulation and modeling process.

Systems are described in a compact and intuitive component diagram using next-generation physical modeling techniques, making them easier to build and understand. Model equations are automatically generated and simplified, yielding concise models and high-speed simulations of sophisticated systems. With MapleSim, you will produce better products and dramatically shorten the product development cycle.

www.maplesoft.com/products/maplesim

For more information contact Maplesoft at:

1-800-267-6583 (USA and Canada) or via email at info@maplesoft.com. If you are outside of the US and Canada, please contact a reseller in your region.

A list of resellers is available at www.maplesoft.com/contact/international.



www.maplesoft.com | info@maplesoft.com

Toll-free: (US & Canada) 1-800-267-6583 | Direct: 1-519-747-2373