IEEE NPSS (Toronto), UOIT, Oshawa, ON, 2-3 June, 2011
International Workshop on Real Time Measurement, Instrumentation & Control [RTMIC]

Symbolic Computation Techniques for Efficient Control Plant Modeling and HIL Simulation: The Case of MapleSim

Tom Lee, Maplesoft

Orang Vahid, Mechanical and Mechatronic Engineering, University of Waterloo

Abstract

This paper describes recent developments in creating better models that still run efficiently for rapid virtual simulations, including realtime, or hardware-in-the-loop (HIL) simulations. The presented techniques have been motivated by the needs of the global automotive industries in particular. Progress in automotive model-based design (MBD), however, has faced very serious recent impediments due to increasing complexity of systems such as those required for hybrid electrical vehicle (HEV) or electric vehicle (EV) design. Such systems are naturally multi-domain and possess challenging dynamics. Conventional numerical simulation tools, many find, are unable to deliver either the necessary system fidelity, or sufficient speed in realtime. This paper describes the techniques embodied in recent simulation tools that deploy symbolic computation techniques. Symbolic approaches, as compared to conventional numeric approaches, manage inherent model equations in a natural algebraic way – i.e. the equations are both exposed and manipulable and not simply encased in compiled black boxes. With direct access to such techniques, various algebraic and computation optimization techniques can be readily applied. A case study applying these techniques for designing a radar-tracking gimbal controller, via inverse kinematic analysis is presented.

Keywords: Dynamic modeling, Realtime simulation, Symbolic computation, MapleSim, Maple

1. Introduction

Industry continues to seek more detailed and more efficient techniques for system simulation within control system design. As products become more complex and sophisticated in function, conventional techniques of physical prototyping is simply too costly in key industries such as automotive, aerospace, and robotics. This paper describes recent developments in creating better models that still run efficiently for rapid virtual simulations, including realtime, or hardware-in-the-loop (HIL) simulations. The presented techniques have been motivated by the needs of the global automotive industries in particular. In auto, the emerging techniques of Model-based Design (MBD) where software plant models now attempt to delay the first physical prototypes, thereby increasing simulation in safer, more cost-effective environments, is becoming the de facto practice in key subsystem design including powertrain, driveline, chassis, etc.

Progress in MBD, however, has faced very serious recent impediments due to increasing complexity of systems such as those required for hybrid electrical vehicle (HEV) or electric vehicle (EV) design. Such systems are naturally multi-domain and possess challenging dynamics. Conventional numerical simulation tools, many find, are unable to deliver either the necessary system fidelity, or sufficient speed in realtime. This paper describes the techniques embodied in recent simulation tools that deploy *symbolic computation* techniques.

2. Symbolic Computation

The vast majority of software techniques applied in engineering are numerical in nature. Typically all quantities (or variables) must be assigned a specific floating point value before any actual calculation can proceed. So if y is the sum of a and b, then, a and b must be given numerical values prior to evaluation of y. Conversely, symbolic systems will allow a and/or b to remain unspecified and calculation can proceed. So if z is the sum of y and a third parameter c, a symbolic system will evaluate z as y + c = a + b + c. In other words the framework is naturally algebraic and not just numerical. With this as a foundation, the most prominent symbolic computation systems will extend the algebraic manipulations to compute an extensive range of important mathematical tasks. Manipulation of trigonometric and other transcendental functions, symbolic matrix operations, and where feasible, the computation of closed form analytical solutions to algebraic and differential equations, are all examples of fully mature techniques.

The system, Maple, is a well-known example of a comprehensive, general-purpose, mathematical computation system based on a symbolic framework. Preservation of mathematical structure in computation. Initially proposed in 1983 [**1**], it has an extensive range of symbolic, numeric, programming, and utility features that can manipulate most mathematical operations. Key mathematical operations supported that are relevant to the modeling of dynamic physical systems include, differentiation, integration, Laplace and inverse Laplace transforms, matrix operations including computation of eigenvalues, Laplace and inverse Laplace transforms, symbolic complex number operations, and the solution of differential equations. Aside from the available computation functions, engineering model equations developed under a symbolic framework are,

- Inherently parametric: since values need not be specified to proceed with computations model expressions of symbolic (unspecified) parameters are readily and efficiently managed.
- Analytical solutions or symbolic analysis: although closed form, fully symbolic solutions are often infeasible for realistic systems, certain analytical techniques can be transformed to produce information that are valid for the entire parameter space, or all time as opposed to specific instances. See the inverse kinematics example later in this paper.

Conversely, a more conventional numerical formulation technique results in model equations that can only be computed for output values given specific input values. Any adjustments or considerations on the structure or inherent relationships with the model will require a complete reformulation of the model. In this sense, models created within a numerical formulation technique are often considered compiled "black boxes".

3. Acausal modeling environments

Symbolic computation techniques have been used for a range of highly specialized engineering model development projects for some time but tools like Maple have not had a significant impact in more mainstream engineering design activities. More recently several important developments have appeared within the engineering modeling world that, when combined with symbolic computation technology, is proving to be a very effective platform for addressing key industry needs.

One such modeling trend is the growing popularity of acausal modeling tools. Acausal tools are in contrast to causal tools where system models are represented in software as a network blocks that represent diagramatically the mathematical relationship within the governing differential equations of a model. Once set up, this diagram also represents an algorithm schematic for the numerical integration of the differential equations. In this way, this scheme is “causal” as each iteration of a numerical integration is dependent on the corresponding values of the previous step.

In many ways, signal-flow models can be considered, “virtual analog computer” models, referring to the circuit-based devices of the 1960’s and 1970’s used for generating solution signals of hard-wired representations of differential equations [**2**]. Signal flow models use the same conceptual and mathematical framework of the analog computer but one enters the model on a robust, flexible, digital software environment and unfortunately, this similarity is now part of an analytical bottleneck. They require the full mathematical (differential) equations of the plant model, prior to implementing them into software as block diagrams. As with the analog computer, significant amounts of effort must go into the manual derivation of these equations. Of course, the actual implementation on the computer is much easier today than with the analog computer. The reality is, though, the plant model derivation can be very time consuming. Some estimate that upwards of 80% of a modeling and simulation project time is spent on the derivation and that time is typically spent on error-prone manual methods (paper, pencil, calculators, reference books, etc.) [**3**]. So even with immensely powerful signal-flow tools, a large part of the engineering project remains poorly supported by good software.

In response, new systems such as Dymola® [**4**], SimulationX® [**5**], and MapleSim® [**6**] among others, have appeared. Collectively, these constitute the acausal modeling tools. The primary difference is that the model blocks and schematics are direct reflections of the component structure of the physical system. For example, if the model is electronic equation, one would connect a resistor to an inductor, etc. as opposed to representing the governing equations that relate the resistor and inductor parameters. This close conceptual connection to the actual physical topology of the system has also produced the term “physical modeling” as the method embodied in acausal systems.

This component-based worldview immediately provides relief from the burden of equation development prior to defining the system on the computer. One does not need to have the equations at hand to begin and complete the model and as a consequence, these modeling tools have had very rapid acceptance in industry especially among those engaging in complex system modeling for which the manual derivation required by the signal flow methods are prohibitively onerous.

4. Multidomain systems

The component-based framework also resolves another major challenge in addition to equation derivation: multi-domain systems. Signal-flow models require careful reconciliation of the mathematics from one domain to another. The addition of an electric motor to a mechanical device, intuitively simple in the physical sense, require more care than connecting blocks. Acausal techniques typically deal with this complexity automatically as all of them are based on mathematical formulation techniques that necessarily has to reconcile the mathematical subparts of the models whether from within the same domain or between domains. Literally, with these systems, you connect a motor to a mechanical device and the mathematical juggling required is part of the internal formulation algorithm.

One very notable dimension to the issue of rapid multidomain model development is the modeling standard Modelica [**7**], a non-proprietary, object-oriented, equation based language to describe and generate simulation procedures for complex physical systems containing, mechanical, electrical, electronic, hydraulic, thermal, control, electric power, process-oriented subcomponents, and other domain components. Systems such as Dymola and MapleSim, among others, use the Modelica language extensively to ensure sufficient prebuilt component availability and the necessary algorithms to ensure model correctness even for multidomain systems. Being a standard, Modelica also provides certain measures of inter-system compatibility – i.e. exchanging components or even complete models from system to system.

5. Natively symbolic modeling systems

In some sense Modelica, and all other physical modeling tools must implement some form of symbolic manipulation of the mathematics in order to automate the formulation of the model equations. Additionally, most systems also apply some form of model simplification to achieve computational efficiencies. A distinct subset of the available physical modeling tools are built on a framework of a general-purpose symbolic computation platform. The most common example is MapleSim that is built on the well-known Maple language – a general purpose symbolic mathematical system. Such an architecture offers greater access to symbolic operations in all aspects of the model equation management. At a basic level, operations such as the recognition of terms multiplied by 0 or 1, or term cancellations, or applications of simplifying trigonometric identities can be readily applied. Another very important example is the reduction of high index differential algebraic equations (DAEs) for multibody systems, to lower index DAEs. These equations arise in situations where there are hard geometric constraints that result in the coupling of algebraic equations to the differential equations of the system model equation set. All of these techniques are the direct consequence of a natively symbolic framework.

6. Code generation for plant models in realtime simulation

In the case of MapleSim, another consequence of being natively symbolic is the optimization of source code generated for the purposes of realtime simulation (e.g. in HIL) or other simulation applications where maximum speed is required. Historically the conversion of mathematical expressions to code to programming languages has been a fairly common aspect of many commercial symbolic computation systems. In the case of MapleSim, this basic technique as implemented in Maple has been refined to be particularly useful for realtime simulation.

Mathematics

$$f(x) = \begin{cases} x & x \le 0 \\ \dfrac{1}{x} & otherwise \end{cases} \longrightarrow$$

Maple

```
f := proc(x)
    if x <= 0.0
then x
       else 1.0/x
    end if;
end;
```

$\longrightarrow$

C

```
double f (double x)
{
  if (x <= 0.0e0)
    return(x);
  else
    return(0.1e1 / x);
}
```

Figure 1: From mathematics to computer code

Two of the most significant refinements are the recursive simplification of the expression set and the addition of utility code for specific target environments. The first refers to the successive identification and replacement of commonly appearing terms with simple value variables. As shown in Figure 2, f1 consists of a calculation involving sin(a), cos(a) and b. Both trigonometric terms, however, are repeatedly used throughout the equation set. The code optimization tools will automatically isolate such repeated terms and assign a simple variable name and then reform the expression that originally involved these terms, only in terms of these new variables. Effectively, computationally expensive function calls have been replaced by very efficient table look-up operations throughout the model. Furthermore, this process is fully recursive and is applied through all of the mathematical dependencies within the model equation set.

$$\begin{bmatrix} f1 = \sin(a)\ (\cos(a) + b) \\ f2 = (1 + \cos(b) + b)\ \sin(a)\ \cos(a) \\ f3 = (a - b\sin(a))\ \cos(a) \end{bmatrix} \longrightarrow \begin{bmatrix} t2 = \sin(a) \\ t4 = b\ t2 \\ t3 = \cos(a) \\ t1 = t2\ t3 \\ f1 = t1 + t4 \\ f2 = (1 + \cos(b) + b)\ t1 \\ f3 = (a - t4)\ t3 \end{bmatrix}$$

Figure 2: Code optimization

For complex models, this technique has very significant final impact. Table 1 shows the raw symbolic reduction results of replacing more costly operations with simple variables. In the case of the full vehicle model and the space rover, the improvements are several orders of magnitude. In reality, this improvement is then subject to all of the other computational overhead in simulation and when all of these factors are involved, the net speed up is more typically one order of magnitude as shown in Table 2.

Table 1: Reduction of mathematical operations with the use of symbolic optimization

	Operation	Initial	Symbolic Optimized
Active Suspension	functions	91	**5**
	additions	23	**3**
	multiplications	35	**5**
Vehicle Model	functions	708,738	**80**
	additions	110,168	**8,362**
	multiplications	318,479	**8,942**
Battery Start-up	functions	1,380	**83**
	additions	454	**126**
	multiplications	934	**116**
Ackerman Steering	functions	2,956	**20**
	additions	814	**75**
	multiplications	1,389	**86**
Space Rover	functions	2,726,936	**90**
	additions	426,755	**6,386**
	multiplications	1,048,636	**7,431**

Typically, the code generated by the process are in standard C or realtime platform-specific forms such as s-functions in the case of Realtime Workshop® or LabVIEW® VI's or dSPACE® blocks.

Table 2: Examples of simulation speed improvement as a result of symbolic optimization

	Reference based on signal flow software (s)*	Optimized code performance (s)*	Speed improvement
Double Pendulum	137	14	9.9x
Four Bar Linkage	288	70	4.1x
Stewart Platform	710	74	9.6x

* Simulation cycle time = 10ms

7. General contribution to modern engineering workflow

At the simplest level, the above technique replaces the paper-pencil modeling that has traditionally been the norm for deriving the equations necessary for many forms of simulation (See Figure 3). With increasing complexity, appropriate symbolic techniques have enabled users to more efficiently manage the mathematics of derivation and reduce human error. In some sense, this is "calculus automation". Another influencing factor is, of course, engineers under the age of 40 will likely have experienced such tools as part of their mathematics education and many are now beginning to make the connection between the technology and potential application even if it were not explicitly addressed in their training.

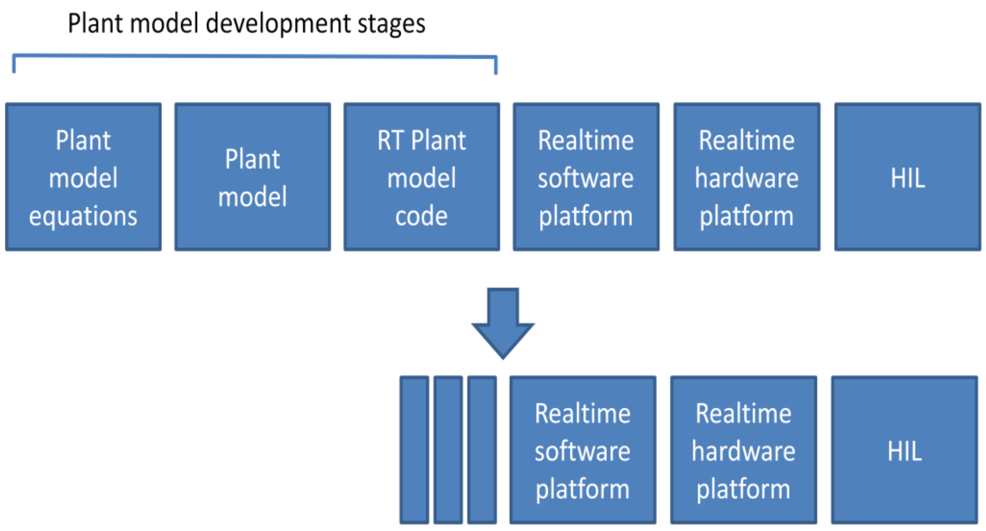

Plant model development stages
Plant model equations
Plant model
RT Plant model code
Realtime software platform
Realtime hardware platform
HIL
Realtime software platform
Realtime hardware platform
HIL

Figure 3: Time savings from automated plant modeling for HIL

Overall, the net benefit is two-fold:

1. Reduction of model development time: For complex models the effort required to develop sufficiently complex models, in terms of mathematical derivation and conversion of the blocks to signal-flow schemes, can literally be person-months for real systems. Vendor claims are that comparable model development process with a symbolic acausal system is on the order of a few person-weeks.
2. Increasing fidelity for realtime simulation: With traditional techniques, even if one could develop a higher fidelity model manually, there is the question of whether such models will run within a realtime cycle (typically on the order 1 ms). The code optimization technique can maximize the feasibility probability. For example, in Table 2, the Stewart platform in realtime using conventional techniques shows a compute time of 710 µs which is dangerously close to being at the limit of feasibility. In fact, when the typical overhead in I/O and other factors are considered, it may be very challenging without further, manual tuning of the code. Conversely the symbolically optimized model code runs in 74 µs which provides sufficient headroom for the realtime simulation.

8. Case study: Radar Gimbal Tracking Controller

A tracking radar on an aircraft uses a 2-DOF gimbal mechanism (elevation, θ_{el}, and azimuth, θ_{az}) to adjust the radar dish to point at a selected target, typically located by a global positioning system that provides the latitude, longitude and altitude ($\mathbf{r}_t = [x_t \quad y_t \quad z_t]^T$) of the target. The aircraft can approach the target from any position and altitude (defined by $\mathbf{r}_a = [x_a(t) \quad y_a(t) \quad z_a(t)]^T$), and with any orientation (defined by three Euler angles, $\xi(t), \eta(t), \zeta(t)$). The goal is to determine the required azimuth and elevation angles such that the radar always point towards the target.

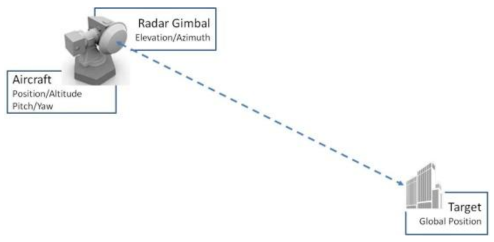


Figure 4: Radar tracking problem

To tackle this problem, a "mechanism" model is built in MapleSim using Multibody components as shown in Figure 5. The mechanism can be described in three parts: the aircraft, modeled as a 6-dof platform; the gimbal, modeled as a 2-dof platform; and the target, defined as a point in space as absolute X, Y, Z coordinates. The "line of sight" is defined as a position constraint, connected between the centre of the radar gimbal and a spherical joint at the target, with a prismatic joint. This mechanically connects the gimbal to the target, making the gimbal angles mathematically deterministic.

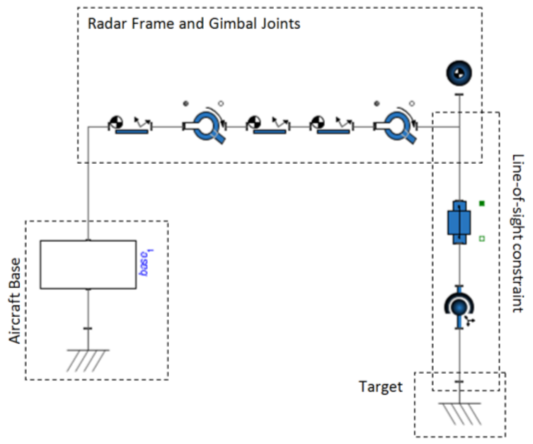


Figure 5: Radar gimbal tracking control system topology

As shown in Figure 6, by only selecting coordinates in the aircraft and gimbal platforms of coordinates (and ignoring the coordinates associated with the line-of-sight), the 6-DOF model in Figure 5, can be modeled using 8 generalized coordinates resulting in two constraint equations.

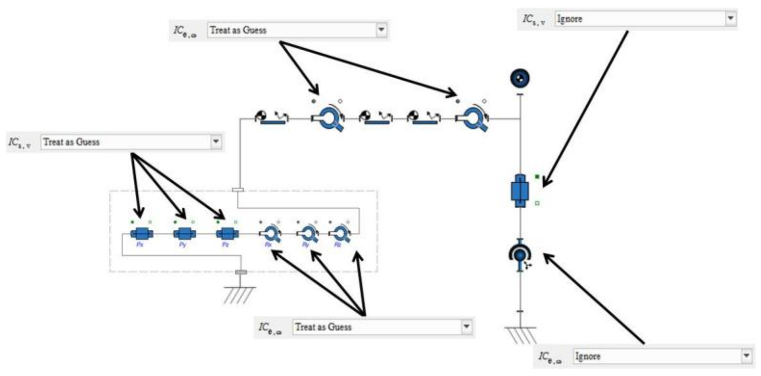
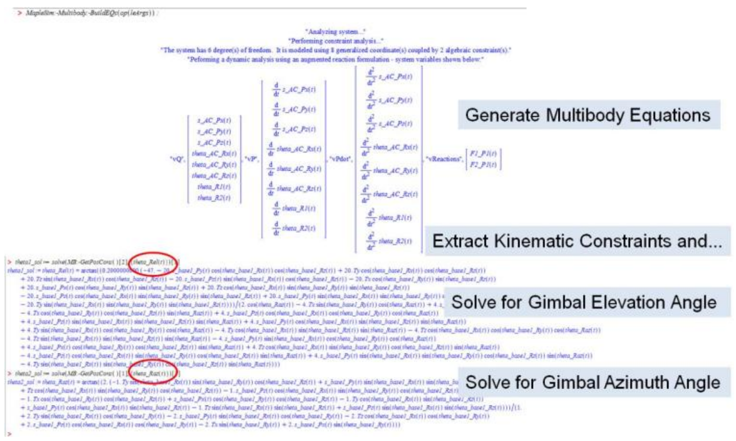

$IC_{\theta,\omega}$ Treat as Guess
$IC_{s,v}$ Ignore
$IC_{s,v}$ Treat as Guess
$IC_{\theta,\omega}$ Treat as Guess
$IC_{\theta,\omega}$ Ignore

Figure 6: Selection of coordinates for the generation of the equations of motion

The constraint equations take the form

$$f_i(\theta_{az}, \theta_{el}, \mathbf{r}_\mathrm{a}, \xi, \eta, \zeta; \mathbf{r}_\mathrm{t}) = 0, \quad i = 1,2. \tag{1}$$

The constraint equations (as well as the equations of motion) are derived automatically using the Mutlibody Analysis Tamplate in Maple (See Figure 7). In these equations, the target coordinates, x_t, y_t, and z_t appear as parmaters.



```
> MapleSim:-Multibody:-BuildEQs(op(leArgs)) :
                    "Analyzing system..."
                "Performing constraint analysis..."
"The system has 6 degree(s) of freedom.  It is modeled using 8 generalized coordinate(s) coupled by 2 algebraic constraint(s)."
 "Peforming a dynamic analysis using an augmented reaction formulation - system variables shown below:"
```

Generate Multibody Equations
Extract Kinematic Constraints and...
Solve for Gimbal Elevation Angle
Solve for Gimbal Azimuth Angle

Figure 7: Automatic generation of the equations of motion and constraints in Maple

The symbolic capabilities of Maple can be further utilized to simplify the constraint equations even further. Benefiting from the particular form of the resulting constraint equations, Maple's symbolic solver can be used to convert Eq. (1) to

$$\begin{aligned} \theta_{az} &= g_1(\mathbf{r}_\mathrm{a}, \xi, \eta, \zeta; \mathbf{r}_\mathrm{t}) \\ \theta_{el} &= g_2(\theta_{az}, \mathbf{r}_\mathrm{a}, \xi, \eta, \zeta; \mathbf{r}_\mathrm{t}) \end{aligned} . \tag{2}$$

These equations can be solved for any given aircraft position and orientation and target position, sequentially. To implement the above solution in the radar gimbal model, the unsolved parametric Eqs. (2) are converted into a "custom component" using MapleSim's custom component creation capability that uses Maples symbolic engine. This custom component is shown in Figure 8. The six time-dependent "inputs" on the left define the aircraft's position and orientations. The two "output" on the right are the desired azimuth and elevation angles. As mentioned above, the target position information is included as three parameters.

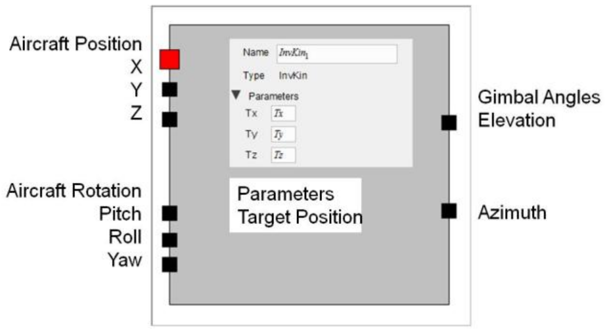


Figure 8: Converting the constraint equations to a custom component

Figure 9 shows the modified model of Figure 5. In this model the line-of-sight constraint is removed and the custom shown in Figure 8 is inserted. A block is added that generates aircraft's arbitrary position and orientation data. The azimuth and elevation joints of the radar gimbal are driven with the desired angles from the custom component using "torque drivers" and simple PID controllers (see Figure 10).

The graphical representation of the model (generated by MapleSim's 3D visualization and animation environment) is shown in Figure 12. Sample tracking results are shown in Figure 13. It is worth mentioning that, through seamless connection to Maple, MapleSim has the capability of optimizing the controller parameters based on a defined performance criterion.

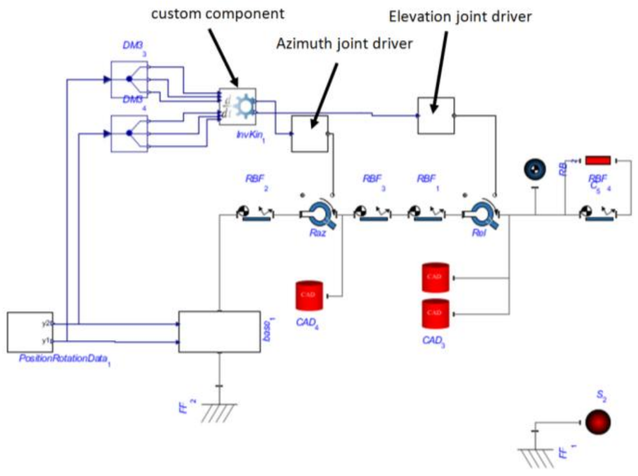


Figure 9: Modified radar tracking model using the inverse kinematic custom component

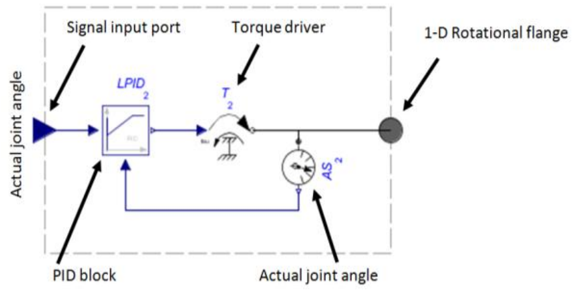


Figure 10: Joint driver subsystem

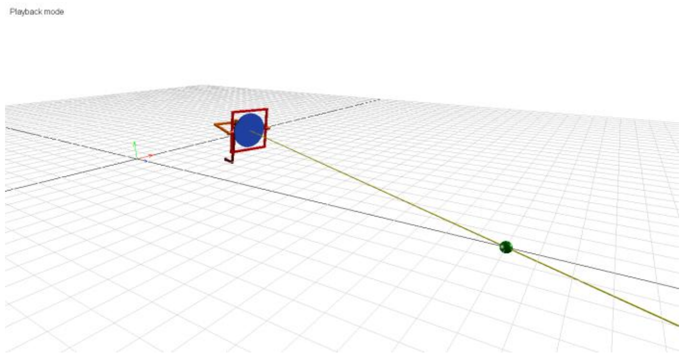


Figure 11: Gimbal tracking visualization in MapleSim

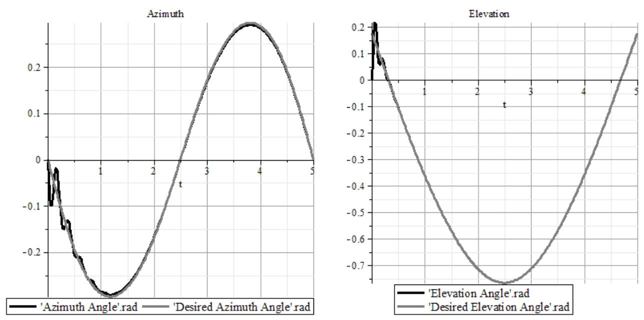


Figure 12: Sample simulation results

The next step – once the modeling and tuning phases are completed – maybe to perform realtime simulation and possibly incorporate HiL testing. MapleSim has built-in capabilities for connecting to Simulink (via s-functions), LabView, and dSpace platforms. Here, we give simple examples based on the gimbal model of Figure 9 for Simulink and LabView RT.

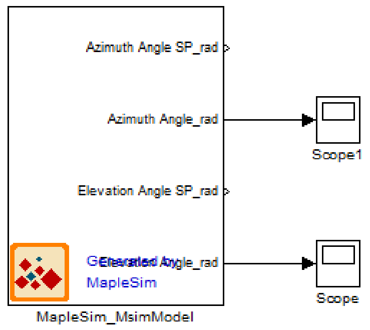


Figure 13: Simulink S-function created by MapleSim's Simulink Connector utility

Figure 13 shows a simple Simulink model that includes the entire model of Figure 9 as a auto-generated S-function block. The model takes roughly .2 s to run 10 s of simulation time. This model is fully compatible with the Real Time Workshop.

Figure 14 shows a screenshot of a LabView model containing the model of Figure 9. The corresponding 'dll' file was automatically generated by MapleSim's LabView Connector utility. On an NI PXIe-8108 platform the model runs with a 20µs cycle time.

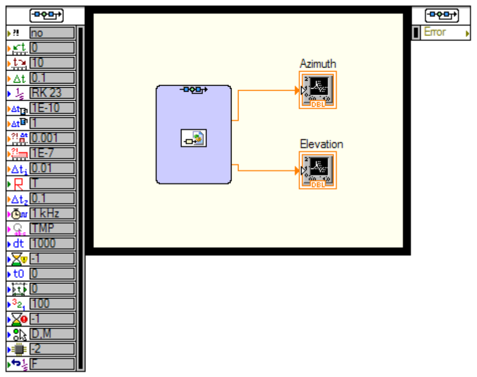


Figure 14: LabView RT model created using MapleSim's LabView Connector utility

9. Conclusions

This paper presented the fundamental elements of new developments in dynamic system modeling. The primary focus was the novel application of symbolic computation methods that can potentially reduce model development time as well as improve model execution time for realtime applications. A case study presenting the model and advanced inverse kinematic analysis for a radar gimbal mechanism was provided.

References

1. Char, B.; Geddes, K.; Gonnet, G.: "The Maple symbolic computation system", ACM SIGSAM Bulletin, Volume 17 Issue 3-4, August-November 1983.

2. Gordon, G.: System Simulation Second Edition, Prentice Hall, 1978.

3. Estimates from user survey conducted by Maplesoft, 2007, unpublished.

4. URL: http://www.3ds.com/products/catia/portfolio/dymola

5. URL: http://www.itisim.com/simulationx_505.html

6. URL: http://www.maplesoft.com/products/maplesim/index.aspx

7. URL: http://www.modelica.org

Authors

Tom Lee is Vice President of Applications Engineering at Maplesoft. He is also an adjunct professor in Systems Design Engineering, University of Waterloo. He holds a Ph.D. in Mechanical Engineering, and a B.A.Sc. and M.A.Sc. in Systems Design Engineering, all from the University of Waterloo.

Orang Vahid received his BS from Sharif University, Tehran, Iran in 1994, MS from Amir Kabir University, Tehran, Iran in 1997, and MASc and PhD from the University of Waterloo, Waterloo, Canada in 2005 and 2009, respectively. He has spent two years as Postdoctoral Fellow at the Mechanical and Mechatronics Engineering Department of the University of Waterloo conducting research in various fields such as modeling and power optimization of planetary rovers and green intelligent transportation systems. His research interests include multi-body dynamics, modeling and analysis of automotive driveline components, automotive noise and vibration, and nonlinear vibration.