The Coupled-Domain System Simulation/Simulatability Problem

J. Jopling†, D. Rose‡, R. Fair†

†Department of Electrical and Computer Engineering
‡Department of Computer Science
Duke University, P.O.Box 90291, Durham, NC, 27708-0291, jj@ee.duke.edu

ABSTRACT

This paper discusses the difficulties in current approaches to modeling and simulation of coupled-domain systems and presents new approaches and solutions to simulation execution and the model-simulation gap.

The ongoing paradigm shift toward coupled-domain systems and total system simulation has spawned a wealth of activity in modeling efforts and languages with which to represent models, generally leaving simulation implementations taken for granted. New approaches to simulation of coupled-domain systems have been proposed, implemented and studied, independant of a standard modeling language. The notion of a 'model compiler' is introduced to help bridge the growing gap between modeling and simulation.

Keywords: System simulation, coupled-domain simulation

1 INTRODUCTION

The recent paradigm shift toward coupled domain systems has spurred an interest in incorporation of subsystems from different energy domains into a complete system model and ultimately a complete system simulation. To this end, popular modeling languages have added extensions to support such schemes(VHDL→VHDL-AMS, Verilog→Verilog-A, etc.) while providing the modeler the convenience and comfort of a language with which he/she is already familiar. This is the layer of abstraction that is necessary to facilitate effective modeling of such systems amidst their complex relationships (Figure 1).

However, the complement to the capability of representing the system in a language is that of simulating that representation. Unfortunately, a majority of recent efforts have been in how to represent the model, with the implementation of the simulation taken for granted. The immense success of SPICE (and its variants) as a circuit simulator lends to the natural approach to solve the composite system modeling/simulation problem by merely extending the modeling language abstraction and relying on the 'tried-and-true' methods of circuit simulation. This has resulted in a growing gap between the capabilities of the modeling language and the capabilities of the suporting simulator.

To resolve this phenomenon, new approaches to simulation of coupled-domain systems have been proposed, implemented and studied independantly of a standard modeling language. The DAENS simulator is an attempt at representing and solving the system as a set of fully implicit differential-algebraic equations (DAEs). The DAENS2 simulator took the knowledge gained from the DAENS study and attempts to solve the system as a set of potential-controlled flux DAEs. The difficuties with those approaches revitalized an interest in a state-based approach.

As a consequence of the simulation approach and implementation being devised completely independant of a modeling language, the gap between modeling and simulation is at first left even more glaringly open. However, this is resolved simply by the idea of a 'model compiler'. This is an entity that would span the void between a model represented in a convenient language and the much more cumbersome representation of information needed by the supporting simulator. Knowing the rules and structures of each, the model compiler would interpret the conveniently coded model and translate it into the structures and models (if necessary) required by the simulator (Figure 5).

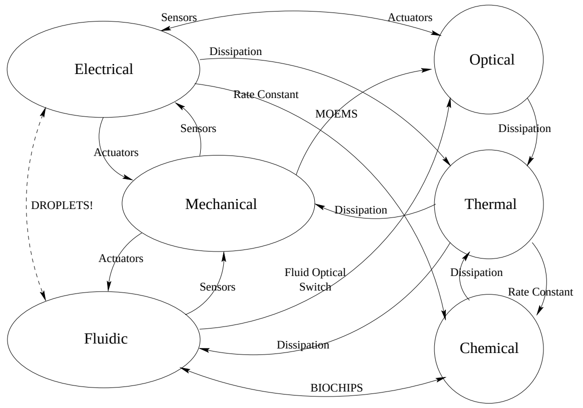

Figure 1: Snapshot of coupled domain relationships.

2 APPROACH

To be able to represent coupled-domain systems, it is desireable that the modeling language inherently support multiple domain behavior. In addition, having stepped out of the purely electrical domain, it is necessary that the modeling language support both temporal derivatives and spatial derivatives/discretizations. VHDL-AMS, MAST, and Verilog-A, for example, support the representation of arbitrary domains; however, implementations are still realized atop legacy simulation approaches. VHDL-AMS also specifically supports discretizations, although simulators to date have not capitalized on this aspect. The power of SPICE may still be leveraged by generating equivalent circuits; however, the multiple domains must be abstracted out, and the resulting representation is often counter-intuitive and unnecessarily complex.

To be able to effectively simulate coupled domain systems, it is desireable to have algorithms that are stable, fast, and also specifically address coupled-domain issues, such as stiffness. Using C/C++ is advantageous in that it is a general purpose language; however, that generality comes with the added cost of having to code everything manually. SPICE, as mentioned, is a widely used simulation tool; however, its core algorithm has changed little since its creation in the early 1970's and may suffer from instabilities and nonconvergence [6]. VHDL-AMS simulators, as mentioned, don't yet support all aspects of the modeling language; furthermore, many implementations rely on unstable algorithms.

Given these facts, it is time to reasses what would now be the best engineering approach/tradeoffs to assembling a coupled-domain system simulator, much as SPICE was the engineering solution to the circuit simulation problem a few decades ago. Since then, technology has matured. Computing power is many times greater, advances in numerical analysis have introduced new algorithms with desirable properties, and network analysis and graph theory have matured and provide additional insight and algorithms.

A new supporting environment for coupled-domain system simulation is needed. Matlab was chosen as the cornerstone upon which to build this support. Matlab provides a foundation of robust and optimized numerical algorithms, a scripting language with inherent support for disctretizations, support for abstract data types, built-in graphical interface capability, and the flexibility of allowing arbitrary user-defined functions/models. In the long term, it would also be possible to export the system to a compiled executable, which would improve speed and distributability.

3 DAENS

DAENS (Differential and Algebraic Equation Network Solver) attempted to first address this problem as a system of fully implicit nonlinear differential-algebraic equations (DAEs):

$$F(\vec{x}, \dot{\vec{x}}, t) = 0 \tag{1}$$

where $\vec{x}$ is a vector of all system variables, be they voltages, currents, displacements, forces, etc. Based on previous work [1,2], the Newton waveform relaxation (NWR) algorithm was chosen to solve this system. Solving for the Newton direction of the above implicit system, employing the operator Fréchet derivative, gives:

$$J_1 \frac{\partial}{\partial t} \Delta\vec{x} + J_2 \Delta\vec{x} = -F \tag{2}$$

where $J_1 = \partial F/\partial \dot{\vec{x}}$ and $J_2 = \partial F/\partial \vec{x}$. This is now a linearized DAE to solve for the Newton direction $\Delta\vec{x}$. To accomodate this form, model descriptions were defined that provided their corresponding branch constitutive relationships (BCRs), in the implicit form $f(\vec{y}, \dot{\vec{y}}, t)$, and the Jacobian information $\partial f/\partial \vec{y}$ and $\partial f/\partial \dot{\vec{y}}$.

Figure 2 shows a fictitious system with coupled electrical, mechanical and fluidic components. Figure 3 shows the results of simulating this system using DAENS.

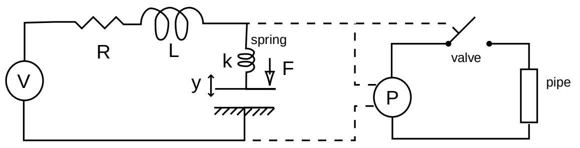

Figure 2: Sample electro-mechanical-fluidic system.

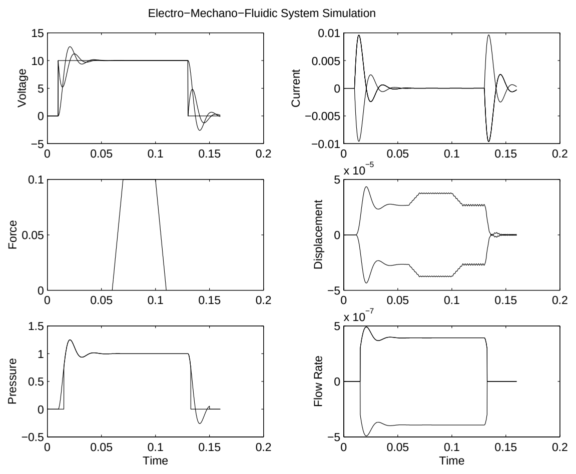

Figure 3: DAENS test case simulation results.

The coupling between the systems is apparent from the plots. The electrical system is activated first, exhibiting the typical RLC response. The electrostatic attraction between the plates of the capacitor causes a displacement with no external force applied. When a force is applied, the displacement is affected accordingly. In addition, the controlled pressure source and valve behave accordingly.

DAENS, though, still suffers from the complexities of solving a fully implicit set of DAEs. The coefficient matrix J_1 is easily singular, a characteristic of DAEs. In addition, it is not guaranteed that this formulation yields an "index-1" DAE, a subclass of DAEs that exhibit properties that facilitate numerical solution [3,7]. DAENS relies on Matlab's capability of detecting the DAE and its index. As such, DAENS walks a fine line between successful simulation of a well-posed model and failed simulation of a less than ideal model. A better approach was needed.

4 DAENS2

DAENS2 learned from the challenges of assembling DAENS in hopes of overcoming the obstacles that plagued the first version. Predominantly, DAENS2 addressed the issue of the form of the J_1 mass matrix and attempted to reapproach the solution to allow a more robust simulation capability against the quality of model formulation by restructuring the rules to facilitate the resulting DAE being of index-1.

The highly singular nature of the DAENS J_1 matrix is predominantly a result of posing the original problem as a fully implicit set of DAEs. DAENS2, instead, added the constraint that the models be posed as potential-controlled flux equations:

$$\phi = f(\psi, \dot{\psi}, t) \tag{3}$$

and assembled the overall system explicitly from flux conservation (equivalently, KCL in the electrical domain), essentially eliminating one set of variables. This approach isn't wholly unreasonable, either, since most actual devices are potential controlled. And, should the need arise, traditional techniques may be employed to fit flux controlled devices into the framework. The resultant Newton equation becomes:

$$J_1 \frac{\partial}{\partial t} \Delta \dot{\vec{\psi}} + J_2 \Delta \vec{\psi} = -F(\vec{\psi}, \dot{\vec{\psi}}, t) \tag{4}$$

In addition to having nice properties on the J_1 matrix, the elimination of a subset of variables and BCRs reduces the dimension of the resultant system to be solved.

Unfortunately, so far DAENS2 has not offered considerable improvement over DAENS. This may partly be due to having used the original DAENS framework to build the new DAENS2. As a result, some of the improvements may not fully have been capitalized. Its failure, however, prompted a renewed look at a state-based approach, which could potentially eliminate the difficulty of working with DAEs completely.

5 SSSS

The State-Space System Simulator (SSSS) is motivated by the notion that the entire behavior of a system can be captured by the dynamics of its 'states' (differential variables) and an algebraic output/observation function. What is needed is a way to transform between the traditional nodal analysis representation and an equivalent state representation. This is not a novel idea, and in the early days of electrical circuit simulation was discussed alongside what was known as a 'hybrid' solution approach, because the resultant equations were not exclusively in terms of voltage or currents, but instead involved both. A state space system is represented as follows:

$$E\dot{\vec{x}} = A\vec{x} + B\vec{u} \tag{5}$$

$$\vec{y} = C\vec{x} + D\vec{u} \tag{6}$$

where $\vec{x}$ is the vector of states, $\vec{u}$ is the vector of independant input sources, and $\vec{y}$ is the vector of desired outputs/observations.

It is possible to perform this transformation on a connected network (graph) by appropriately partitioning the incidence matrix, A_i, for the network and performing a series of elimination operations to arrive at equations for the differential variables entirely in terms of input sources and the states, themselves. The coefficients A and B contain information from the linearized behavior of other elements in the network. The partitioning of A_i is performed first by classifying types of edges and arranging them in a prescribed priority order. Then, a spanning tree of the network graph, when constrained to the previous prioritization, is extracted. One immediate side-effect is that additional information must be stored in the model (or extracted by some other means) as to the classification of each edge.

This procedure is known for a linear, scalar system, from previous work in circuit simulation [4,5]. However, extension of this approach to a nonlinear, vectored system is not immediately apparent. The SSSS simulator builds upon this procedure, extending it to meet the demands of a coupled-domain system simulator. A simple example electrical circuit and the resultant network graph and tree are shown in Figure 4. The solid lines represent the spanning tree, the dotted lines indicate edges in the network not in the tree (links), and the bottom node is ground.

One of the great advantages to the state-based approach is that for a linear system, Matlab already provides state-space simulation tools. Once the transformation has been made, Matlab can easily solve for all the voltages and currents using its built-in state space toolbox.

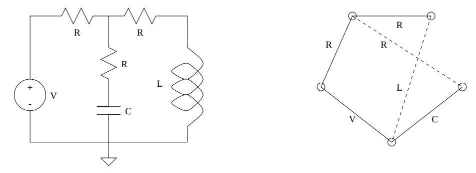

Figure 4: Simple circuit and corresponding graph and tree.

Unfortunately, SSSS must be extended to nonlinear, time-varying state-space systems, which Matlab does not explicitly support. As such, SSSS exploits an additional advantage of this transformation - that it results in a regular ODE for the state equation. Working directly with the state ODE and observation equation, SSSS is able to setup an inner and outer iteration, where it solves the linearized system in the inner loop and updates the time-varying matrices of the nonlinear system in the outer loop. In this manner, the original linear approach is extended to the nonlinear case. The extension to the vectored case is still being studied.

Furthermore, this transformation should greatly improve simulation time, since a rather large DAE system can be reduced to an ODE dependant on only its internal states, potentially a significant reduction. In addition, the output function ultimately would only involve the particular variables of interest, but (being an algebraic output function) this is not as computationally intensive and will not be the primary source of speed improvements.

6 MODEL COMPILER

Each of the simulation approaches above was designed and implemented from the ground up with only the simulation of coupled-domain systems in mind. As such, they do not align themselves conveniently with any existing modeling language, nor with an intuitive form for representing models. As such, it may seem this has only exacerbated the problem of a growing modeling-simulation gap.

However, this gap may be bridged by the concept of a “model compiler”. The concept is illustrated in Figure 5. In this framework, the user is allowed to work with a language which they find comfortable, and use the compiler to interface to one of the new simulation approaches. There are a number of new issues involved with implementing such a compiler, and related work is underway to accomplish that.

7 SUMMARY

This paper has presented a description of the problems today with coupled-domain system simulation, and some of the work in an ongoing effort to address this problem. The new simulation approaches have been conceived and explored, and the notion of a model compiler is introduced to intergrate such new approaches into a full system and the already existing design cycle.

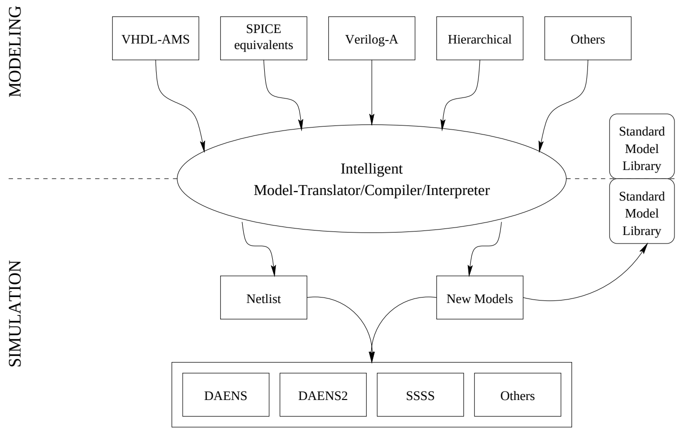

Figure 5: Relationship between modeling and simulation, illustrating the role of the model compiler.

8 ACKNOWLEDGEMENT

The research effort is sponsored by the Defense Advanced Research Projects Agancy (DARPA).

9 REFERENCES

[1] D. Erdman. “The Newton Waveform Relaxation Approach to the Solution of Differential Algebraic Systems for Circuit Simulation”. PhD. Thesis, Duke Univerisity, 1989.

[2] D. Erdman and D. Rose. “Newton Waveform Relaxation Techniques for Tightly Coupled Systems”. IEEE Transactions on Computer-Aided Design. Vol 11, No 5. May 1992 pp598-606

[3] K. Brenan, S. Campbell, and L. Petzold. “Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations”. North-Holland. New York. 1989.

[4] R. Jensen and B. Watkins. “Network Analysis: Theory and Computer Methods”. Prentice-Hall, Inc. New Jersey. 1974.

[5] L. Pillage, R. Rohrer, and C. Visweswariah. “Electronic Circuit and System Simulation Methods”. McGraw-Hill, Inc. New York. 1995.

[6] R. Kielkowski. “Inside SPICE: Overcoming the Obstacles of Circuit Simulation”. McGraw-Hill, Inc. New York. 1994.

[7] L. Shampine, M. Reichelt, and J. Kierzenka. “Solving Index-I DAEs in MATLAB and Simulink”. SIAM Review. Vol 41, No 3, 1999. pp538-552