MODELING PHYSICAL SYSTEMS AS PHYSICAL NETWORKS WITH THE SIMSCAPE LANGUAGE

S. Miller, The MathWorks GmbH
Corresponding Author: S. Miller, The MathWorks GmbH, Adalperostrasse 45, 85737 Ismaning, Germany;
steve.miller@mathworks.de

Abstract. Engineers are relying more and more on simulation to accelerate development processes. Combining computer simulation with Model-Based Design techniques enables engineers to not only complete tasks in a shorter period of time, it makes it possible to find design errors before building hardware, making the errors easier and less expensive to fix.[1] As the systems they develop require integrating control systems with physical systems spanning multiple physical domains (mechanical, electrical, etc.), a useful model of the physical system (plant) is critical for developing an optimized system. Traditional methods (C, FORTRAN, etc.) and signal-based or input-output (causal) methods in graphical software tools (Simulink, etc.) were useful for control engineers because they were written in a language they could easily understand and integrate with their control system models. However, these models can be difficult to reuse, leading to redundant work. A technique which has been applied to electrical systems for quite some time is now being used on multidomain physical systems. The physical network approach, often referred to as acausal modeling, enables engineers to create reusable models of physical components that can span multiple physical domains. Simulation tools that support this method (Simscape, etc.) then build up the equations for the system and solve the differential algebraic equations directly, resulting in accurate simulations of the entire system. This paper focuses on the physical network technique for modeling physical systems and its use in Simscape within the Simulink environment.

1 The need for an additional modeling methods

Input-output methods (also known as causal modeling) have been used to model physical systems for quite some time[2]. One of the main reasons this method is used is because it is the natural language of control engineers. In a standard control loop, the plant is represented as a transfer function with an input and an output. Finding a mathematical representation composed of blocks with inputs and outputs fits naturally into this system, and is easy for a control engineer to use and understand.

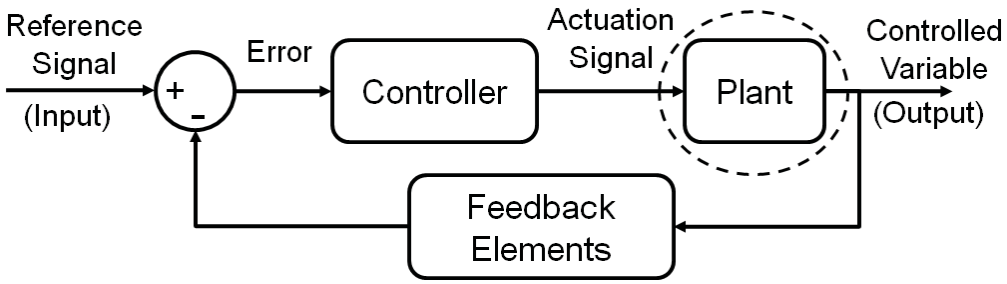
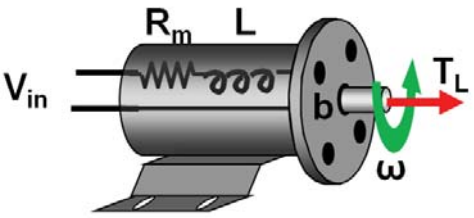


Figure 1. Diagram representing typical control loop structure. The plant, or physical system, is often modeled using the input-output method because it fits well into this structure.

Another reason is the use of data-driven modeling. Techniques that use system identification theory take measurement data and produce a transfer function that reproduces the behavior of the system. Because it is based on measurement data, these models can be very accurate for behavior about a specific operating point and can execute very quickly. These types of models typically (but not always) execute very quickly because the modeler has specified the exact calculations that the solver must complete.



$$\dot{i}_m = (V_{in} - K_b\omega - i_m R_m)\frac{1}{L_m}$$

$$\dot{\omega} = (K_t i_m - T_L - b_m\omega)J$$

Figure 2. Diagram of a DC Motor and equations

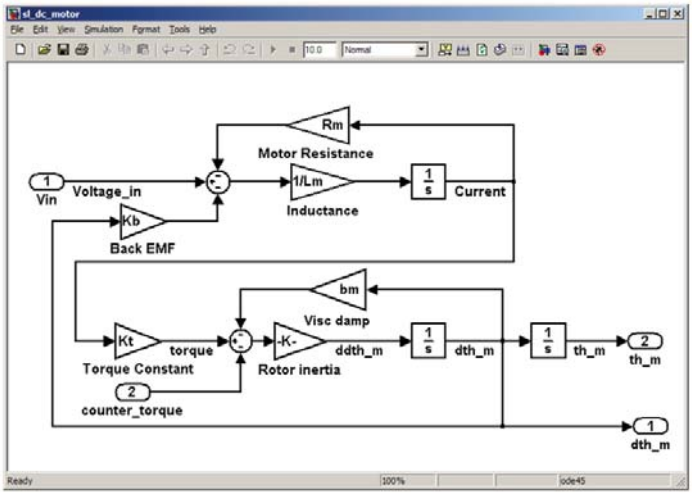


Figure 3. Implementation of equations in Simulink, demonstrating input-output modeling method.

However, while this method works very well for control systems, it has some disadvantages when modeling physical systems. Physical systems are often expressed in the form of DAEs, which are composed of sets of equations that must be solved simultaneously. This type of system can only be approximated using an input/output modeling technique. Another disadvantage is that the models that are created depend upon which elements they are connected to. It is necessary to know which inputs are available and which outputs must be calculated in order to connect it with the rest of the system. This type of model is difficult to reuse in other models or applications, for it requires that the other components in the system are modeled in the same manner. This becomes especially complex when modeling components that cross multiple physical domains, like a DC motor or a hydraulic cylinder. In this case, the model is dependent upon multiple other components that may have been modeled by different engineers, restricting the options of implementing the equations and therefore limiting the chances of reusing the model in other applications.

Due to these reasons, engineers began looking for a better method for modeling these types of systems. For purely electrical systems, Kirchoff's laws have been used for quite a long time to express the equations for a entire system by applying a few basic mathematical rules to a network of electrical components represented by their individual mathematical models[3]. For example, the component model of a resistor was represented by v=iR, and this component model of an ideal resistor was identical for all resistors in the electrical network, independent of where the resistors were placed in the circuit. The equations for the entire system could be derived by applying Kirchoff's laws at the nodes of the electrical circuit. This method permits the component models to be modular and reusable while also being able to mathematically represent an entire circuit. Physical modeling languages focused on electrical systems have been in existence for quite some time (VHDL-AMS, VerilogA, etc.)[4].

It was seen that other physical domains were analogous to electrical networks, and that similar rules could be applied to systems composed of one or more other physical domains. With this came the rise of other physical modeling languages such as Simscape[5], Modelica[6], 20Sim, and others that wished to provide for multidomain physical systems the same benefits that Kirchoff's laws provided for electrical systems. Using this method, an engineer could build up a library of component models that could be reused in a variety of models in a variety of applications. When used in a modeling environment that also permits input/output modeling methods, it enables engineers to model the entire system with each component (physical or control element) modeled in a language natural for that domain. When combined with tools for generating C code, the engineer then has the power to apply the Model-Based Design process from the advanced development all the way through production code generation, which is a must for engineers doing control system development.

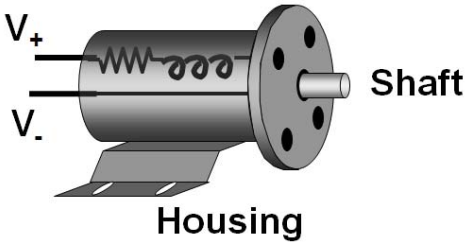


Figure 4. Diagram of a DC Motor with connections that match with the physical network method.

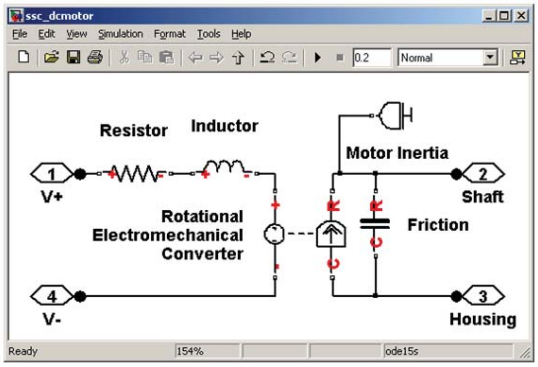


Figure 5. Implementation of a DC motor model using the physical network method in Simscape.

2 Basics of the physical network method

The basics of the network method requires describing the physical components in the system using variables particular to the domain that are analogous to voltage and current for electrical systems. These variables can be determined by examining the flows of energy into and out of the component. For example, if we look at a DC motor, the component (in its most basic form) involves two physical domains – electrical and mechanical.

The electrical power can be represented as:

$$P = v\, i$$

The mechanical power entering or exiting the component can be expressed as:

$$P = T\, \omega$$

The variables used in these equations are analogous to each other. Understanding the relationship between the variables in these domains makes it possible to apply constraints to this system for each domain in order to develop the equations for the system.

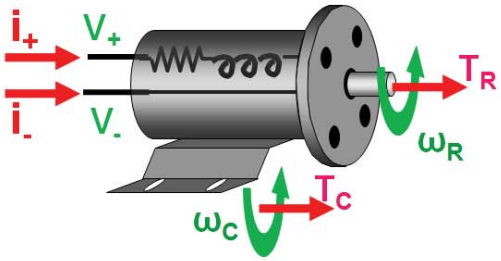


Figure 6. Diagram of DC Motor showing physical network variables

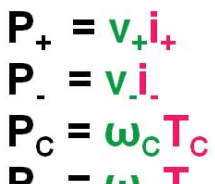


Figure 7. Equation representing energy flowing in and out of the motor

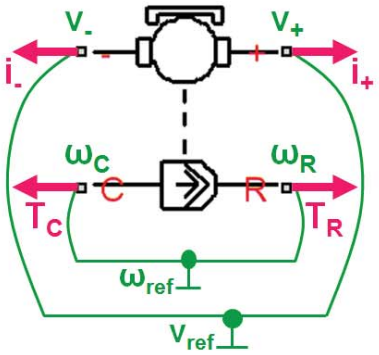


Figure 8. Diagram of physical network element with across and through variables labeled

Kirchoff's voltage law states that the directed sum of the electrical potential differences around any closed circuit must be zero. Stated in a different way, this implies that the voltage of all components' ports attached to an electrical node must be the same. This can be related other physical domains. For a hydraulic node, the pressure at all of the components' ports attached to that node must be the same. In our example of a DC motor, the velocity at all of the components' ports attached to a mechanical node must be the same. These variables (pressure, velocity) are analogous to voltage in an electrical circuit, and in physical networks they are often referred to as *across variables*.

Kirchoff's current law states that the sum of currents flowing towards an electrical node is equal to the sum of currents flowing away from that node. This can be related to other physical domains as well. At a hydraulic node, the amount of fluid flowing into that node must be equal to the amount of fluid flowing out of that node.

This constraint must be applied to all components' ports attached to that hydraulic node. Similarly, for a mechanical node, the forces (or torques) applied in one direction at a node minus the forces (or torques) applied in the opposite direction at that node must be equal to zero. These variables (flow rate, force/torque) are analogous to current in an electrical circuit, and in physical networks they are often referred to as *through variables*.

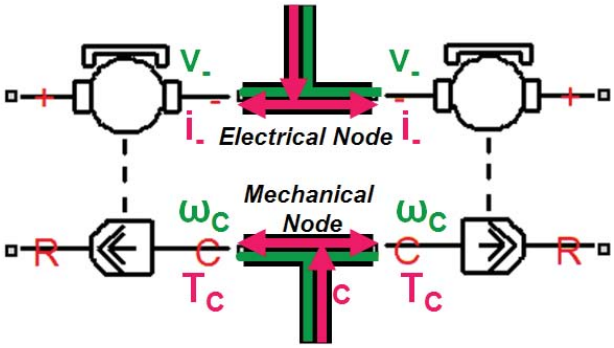


Figure 9. Diagram representing two nodes of a physical network representation.
Applying Kirchoff's laws to these nodes allows the software to determine the equations for the entire system.

Expressing the mathematical model for a component in terms of these variables makes it possible to formulate the equations for the entire system by applying the above laws as constraints to each node. The fact that this analogy can be applied to each physical domain enables engineers to create components spanning multiple physical domains quite easily.

3 Implementing Physical Networks in the Simscape Language

A basic understanding of physical networks makes it easy to understand how such models can be created. Looking at a specific implementation of this technique makes it easy to see how this theory can be applied to actual engineering problems. We will look at a model of a DC motor implemented in the Simscape language in order to see one concrete example of the physical network method for modeling physical systems.

In the case of the DC motor, we have definitions of two physical domains – mechanical and electrical. For each of those domains, we need to define the across and through variables, including their units. Those definitions, implemented in the Simscape language, is shown below.

```
domain electrical

  variables
    v = { 1 , 'V' };
  end

  throughs
    i = { 1 , 'A' };
  end

end
```

Figure 10. Declaration of the electrical domain in the Simscape language

```
domain rotational

  variables
    w = { 1 , 'rad/s' };
  end

  throughs
    t = { 1 , 'N*m' };
  end

end
```

Figure 11. Declaration of the rotational mechanical domain in the Simscape language

The component definition will need to make use of these variables in order to define the equations. It will also have to define the relationship between these variables and the ports (the connections to the nodes) in order for the tool to be able to properly apply the constraints. In addition, other internal variables may be needed for the equations. The definition for the ports, which references these domain definitions, is shown here:

```
nodes
  p = foundation.electrical.electrical; % +:right
  n = foundation.electrical.electrical; % -:left
  R = foundation.mechanical.rotational.rotational; % R:right
  C = foundation.mechanical.rotational.rotational; % C:left
end
```

Figure 12. Declaration of the electrical and mechanical ports of a DC motor in the Simscape language

With these port definitions, the relationship between the ports and the across and through variables is expressed in the Simscape language as shown below:

```
function setup
  through( tq, R.t, C.t ); % through variable tq from r to c
  across( w, R.w, C.w ); % across variable w from r to c
  through( i, p.i, n.i ); % through variable i from p to n
  across( v, p.v, n.v ); % across variable v from p to n
  if Rwind <= 0
    error('Resistance must be greater than zero')
  end
end
```

Figure 13. Setup section of Simscape component file, containing the declaration of the relationship between the variables and nodes in the Simscape language. This section also leverages MATLAB functions to check a parameter value and inform the user if it is out of range.

Useful models of physical components will have parameters that correspond to physical quantities, ideally the kinds of quantities that will be found on industry data sheets. These parameters should also be defined in the component definition file so that they can be integrated with the component equations. The parameters of a DC motor implemented in the Simscape language are shown below:

```
parameters
  Kt = {0.0637 'N*m/A'}; % Torque constant
  Ke = {0.0637 'V/(rad/s)'}; % Back EMF Constant
  Rwind = {0.048 'Ohm'}; % Winding Resistance
  Lwind = {1600e-6 'H'}; % Winding Inductance
  J = {0 'g*cm^2'}; % Motor Inertia
  B = {1e-8 'N*m/(rad/s)'}; % Motor Damping
end
```

Figure 14. Declaration of the component parameters for a DC Motor in the Simscape language

In each case, the units for the parameter are defined along with a realistic default value, in order to provide the user a realistic starting point.

Implementing the equations is obviously a critical part of defining the component. As has been explained, in order to be reusable the equations need to be expressed as a mathematical relationships that are valid for the component and are independent of what other components it is connected to. And, in order to be able to properly express DAEs, these expressions must be able to represent simultaneous sets of equations and not simply input/output relationships or assignment. In the Simscape language, the operator "==" is used to represent a mathematical relationship that is not based on assignment. The equations for a DC motor based on the through and across variables defined above are shown below:

```
equation
    w == theta.der;
    v == Ke*w + i*Rwind + Lwind*i.der;  % Motor equations
    tq == -Kt*i + B*w + J*w.der;
end
```

Figure 15. Declaration of component equations for a DC Motor in the Simscape language

These mathematical relationships are evaluated simultaneously at each step of the simulation. The techniques used to formulate the equations for the system is explained in the following section.

4 Simscape Simulations

The process of converting the physical network diagram into a set of DAEs that can be integrated is shown in the diagram below:

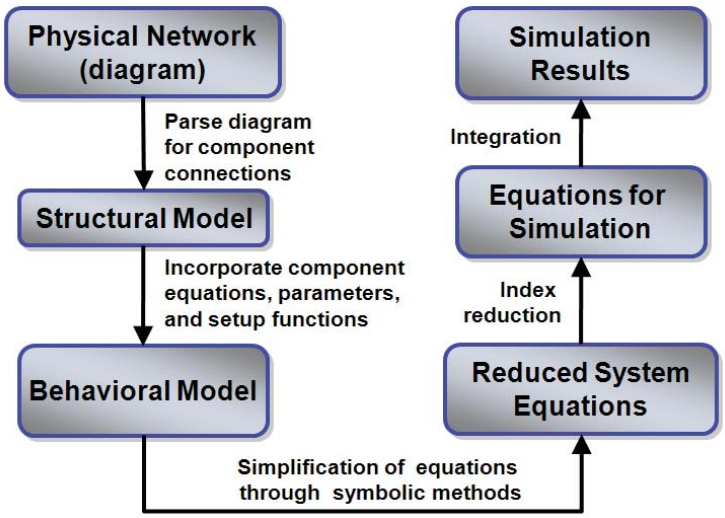


Figure 16. Flowchart describing the steps of converting a physical network diagram into system equations.

Once the physical network diagram is constructed in the diagram editor, the first step Simscape must do is to analyze the components in the network and their connections in order to convert it into what is known as a structural model. The structural model is the framework of the physical network. The equations, parameters, and initialization for the components is added to this framework to create a behavioral model of the system. The behavioral model represents the set of DAEs symbolically at the equation level and has all of the information necessary to describe the system. Further steps are necessary to ensure a robust and quick simulation.

The system of equations contained in the behavioral model are analyzed and reduced via symbolic simplification methods. The index of the system of equations is reduced after all possible simplifications have been made. This is done by first identifying the higher index constraint equations and then differentiating these equations in time. With this technique, Simscape can reduce many common classes of index-2 problems to index-1 and index-0 problems. Once the index has been reduced, the resulting system is presented to the integrator as a DAE or ODE depending on the type of integrator. The integrator integrates the system to generate the simulation results.

It is important to note that the equation formulation described here results in a set of equations that can be solved simultaneously. Because of this, there is no need to use an algebraic loop solver to integrate these equations or to add dynamics to your system to break algebraic loops. This is a significant advantage over an input-output approach to modeling physical systems, for the resulting simulation is a more efficient, more robust, and more exact solution.

5 Conclusion

As more engineers rely upon Model-Based Design to improve their development process, the model of the physical system (plant model) is becoming increasingly important. Engineers are learning that it is possible to combine in one environment methods best suited for modeling physical systems (physical networks) and control systems (input-output methods). The Simscape language is an option that allows engineers to model the physical system while leveraging their MATLAB and Simulink experience and legacy models, as well as permitting system level optimization.

6 References

1. S. Popinchalk, J. Glass, R. Shenoy, and R. Aberg, “Working in Teams: Modeling and Control Design within a Single Software Environment,” AIAA Modeling and Simulation Technologies Conference and Exhibit [CD-ROM], Hilton Head, SC, 2007.

2. C. Moler, “The Growth of MATLAB® and The MathWorks over Two Decades,” MATLAB Digest, Boston, January 2006.

3. L. W. Nage, D. O. Pederson, SPICE (Simulation Program with Integrated Circuit Emphasis), Memorandum No. ERL-M382, University of California, Berkeley, Apr. 1973.

4. Christen, E. and Bakalar, K., ”VHDL-AMS-a hardware description language for analog and mixed-signal applications”, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Volume 46, Issue 10 , Oct. 1999.

5. Simscape User's Guide, The MathWorks, Natick, MA, October 2008.

6. H. Elmqvist, S. E. Mattsson, and M. Otter, “Modelica: The new object-oriented modeling language,” presented at The 12th European Simulation Multiconference, Manchester, UK, 1998