International Conference on Simulation in Engineering Education

Proceedings of the 1992 SCS Western Simulation Multiconference
on Simulation in Engineering Education
20-22 January 1992
Newport Beach, California

Edited by
Hamid Vakilzadian
Department of Electrical Engineering
University of Nebraska–Lincoln

Simulation Series
Volume 24
Number 2

Sponsored by:
The Society for Computer Simulation

in participation with
American Society for Engineering Education
IEEE Education Society

IEEE EDUCATION SOCIETY

Bond Graphs — The Right Choice for Educating Students in Modeling Continuous–Time Physical Systems

François E. Cellier

Department of Electrical & Computer Engineering
University of Arizona
Tucson, Arizona 85721

EMail: Cellier@ECE.Arizona.Edu

Abstract

This paper describes a modeling technique that, better than alternative approaches, teaches the student to develop from the start *valid* models of physical continuous–time processes. Several examples of state–space models are presented that look very plausible, give seemingly plausible results, and yet are physically wrong. These examples may serve to illustrate the potential dangers behind state-space descriptions used as a modeling tool. It is our conviction that *model validation* should be integrated with *model building*, and should not be an afterthought. The bond graph modeling technique enables us to describe physical systems in terms that are much closer to physical reality than state–space modeling. Thereby many of the standard pitfalls in making models are avoided right from the beginning. Bad (i.e., non–physical) models have no chance of being created in the first place.

1 Introduction

Modeling has become extremely easy. Modern CSSL–type languages allow the user to specify complex dynamic processes in a state–space format very close to a pure mathematical description, a format that is very convenient to use. It is possible to master the basics of a modern CSSL–type language within less than 30 minutes.

The use of CSSL–type languages is so simple that it is almost impossible to make syntactic mistakes. The application programs will run at once and, within a matter of minutes, elegant–looking multi–colored graphs will appear on the user's screen. Unfortunately, the simplicity of their utilization is at the same time the greatest vice of all CSSL–type languages. It is very tough to convince a student (and most practicing engineers for that matter) that a program that doesn't display any error messages, that produces results with 14 digits accuracy out of which not a single one is zero, and that generates elegant-looking and somewhat plausible curves can still be wrong.

Model validation should not be done after the fact, i.e., for the purpose of verifying that a once constructed model is indeed meaningful. It is our experience that students (and most practicing engineers) won't take the time to verify even the simplest facts about their models once they were able to generate nice–looking graphs. They immediately fall in love with their graphs ... and true love, as we all know, is blind.

Here is the most common mistake that I find in my students' programs. For a simple mechanical system containing a mass, some friction, and a spring, we start out by formulating Newton's law:

$$m \cdot a = f - k \cdot x - b \cdot v \tag{1}$$

which can be quickly rewritten as:

$$m \cdot \ddot{x} = f - k \cdot x - b \cdot \dot{x} \tag{2}$$

and by choosing the outputs of the integrators as state–variables:

$$\dot{x}_1 = x_2 \tag{3}$$

$$\dot{x}_2 = -\frac{k}{m} \cdot x_1 - \frac{b}{m} \cdot x_2 + f \tag{4}$$

which can be coded without problem using any CSSL-type language and will lead to the aforementioned elegant–looking graphs. Unfortunately, the model is incorrect since the student forgot to divide the input f by the mass m. This error would be easy enough to detect — all the student would need to do is to check for

dimensional consistency across each equation. On the left–hand side of Eq.(4), we have an acceleration measured in m/s^2, but the last term on the right–hind side is a force measured in N or $kg \cdot m/s^2$. Obviously, those two terms are not compatible with each other. However, it is my experience that my students won't check for this. They already got their sugar cube. Looking at the graphs is so gratifying, and there are other pressing homework assignments waiting to be completed, thus, why bother.

Yet, dimensional consistency checking won't do the trick in all cases. Sometimes the problems are much more subtle, as the next example shall demonstrate.

2 Crash Rockets or the Modern–day Flying Dutchman

Figure 1 shows a force diagram for a lunar lander module.

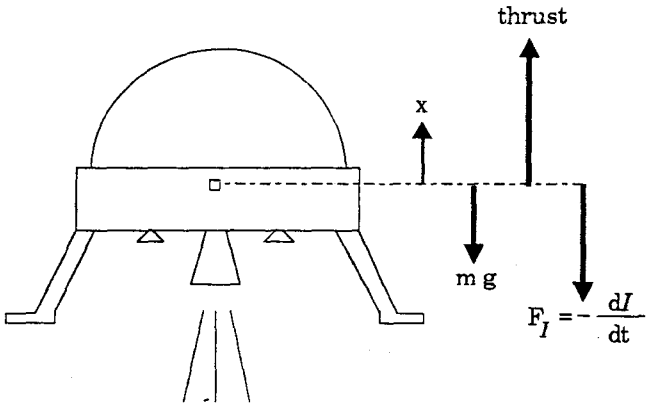

Figure 1. Lunar landing module.

Since the mass of the rocket changes with time, we are inclined to believe that, for this system, Newton's law can be written as:

$$\frac{d(m \cdot v)}{dt} = \frac{dm}{dt} \cdot v + m \cdot \frac{dv}{dt} = thrust - m \cdot g \qquad (5)$$

which can be rewritten as:

$$m \cdot a = thrust - m \cdot g - \frac{dm}{dt} \cdot v \qquad (6)$$

Let us try to validate this model. A good validation technique is the following: We make simplifying assumptions until the problem is reduced to a much simpler problem for which we can check the plausibility of the results obtained. Applied to our rocket: We shall assume that our space craft is far away from any planetary mass. Consequently, we may ignore the gravity term. Moreover, we shall assume that the thrust is always nonnegative, i.e.:

$$thrust \geq 0.0 \qquad (7)$$

Thus, we obtain the following set of equations:

$$a = \frac{1}{m} \cdot (thrust - \dot{m} \cdot v) \qquad (8)$$

$$\dot{m} = -c \cdot thrust \qquad (9)$$

where Eq.(8) is the simplified Newton equation and Eq.(9) is the simplified fuel consumption equation. If we now plug Eq.(9) into Eq.(8), we find:

$$a = \frac{1}{m} \cdot thrust \cdot (1.0 + c \cdot v) \qquad (10)$$

If we assume that we travel initially with a constant velocity of

$$v = -\frac{1}{c} \qquad (11)$$

backward through space, the last factor of Eq.(10) cancels out and we shall never again be able to accelerate or decelerate our space craft. What a fate!

Quite obviously, something has gone awry. The problem is the following: Newton's law is not truly a "law of physics." It is a derived law, i.e., a "law of mathematics." The real "law of physics" states that the total momentum of a closed system must be conserved, or more generally:

$$\mathcal{I}(t + \Delta t) = \mathcal{I}(t) + \Delta\mathcal{I}(t \rightarrow t + \Delta t) \qquad (12)$$

The total momentum $\mathcal{I}$ of a system at time $t + \Delta t$ equals the total momentum at time t plus the (positive or negative) momentum added to (subtracted from) the system between time t and time $t + \Delta t$. Let us apply this law to our space craft:

$$(m - \Delta m) \cdot (v + \Delta v) + \Delta m \cdot v = m \cdot v + thrust \cdot \Delta t \qquad (13)$$

The first term on the left–hand side of Eq.(13) denotes the momentum of the space craft at time $t + \Delta t$. The second term denotes the momentum of the cloud of exhaust at the same time. The first term on the right–hand side of Eq.(13) denotes the momentum of the space craft at time t, and the second term denotes the added momentum due to the drive of the space craft. Notice that we must somehow include the exhaust. Either we consider the cloud of exhaust a part of our system by adding it to the left–hand side of Eq.(13) or we must consider that the exhaust leaves the system

between time t and time $t + \Delta t$ and subtract this term from the right–hand side of Eq.(13).

Neglecting terms in Eq.(13) that are of second order small, we find:

$$m \cdot \Delta v = thrust \cdot \Delta t \qquad (14)$$

or by dividing through Δt and by letting Δt go to zero:

$$m \cdot \frac{dv}{dt} = thrust \qquad (15)$$

Thus, we must use the more familiar form of Newton's law, although the mass of the space craft is undeniably changing with time. Initially, we had simply forgotten to take the cloud of exhaust into account. We could have arrived at the same conclusion by considering the total *kinetic energy* of the system instead of its momentum, since the energy must also be conserved, but the momentum was easier to use in this example.

What has gone wrong in the above problem? Contrary to the previous example in which we introduced a flagrant (though quite common) mathematical manipulation mistake, the last example was mathematically sound throughout. Dimensional consistency checking would not have revealed at any time during the modeling process that there were problems with the model. State–space models describe the change over time of state variables. Unfortunately, physics doesn't know anything about change over time of state variables. All that physics knows about is that this universe of ours is a harsh universe of bartering ... and the only types of merchandise that are up for sale are energy and mass. Mass can be transported from point A to point B, and energy can be either transported from point A to point B or converted from form X to form Y. Beside from the above transactions, nothing goes in this universe of ours. The change over time of a state variable is only a *side effect* of such a bartering deal taking place.

State-space models don't obey any physical code of honor. They don't understand physics at all. They are happy to accept any garbage deal that Johnny Would–Be Modeler chooses to formulate, and they are willing to trade it for slick aesthetically–looking multi-colored graphs.

3 Bond Graphs – Representing the Power Flow Through a System

Figure 2 shows a very simple electrical circuit.

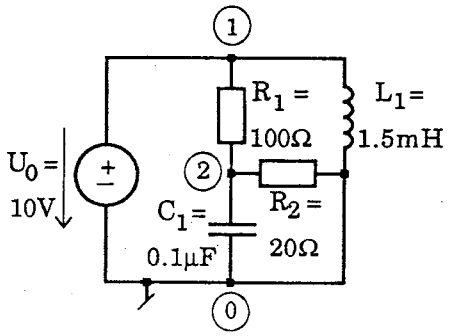

Figure 2. Simple electrical circuit.

A bond graph representation for this circuit is shown in Fig.3.

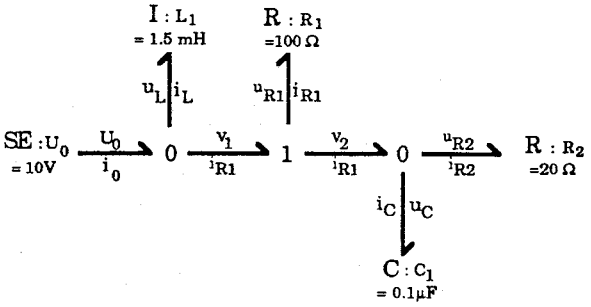

Figure 3. Bond graph of the electrical circuit.

A bond, represented by a bold harpoon, is nothing but a connector that simultaneously connects two variables, one across variable, in bond graph terminology usually referred to as the "effort" e, and one through variable, called the "flow" f. The bond is shown in Fig.4.

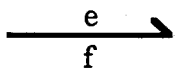

Figure 4. The bond.

Bonds connect either to system elements, such as a resistor **R**, which in bond graph terminology is a *single port element* (since both variables u_R and i_R are connected simultaneously), or to other bonds in a *junction*. Two different types of junctions exist, the so–called 0–junction, and the so–called 1–junction. In a 0–junction,

all effort variables are equal while all flow variables add up to zero. A 0–junction is thus equivalent to a node in an electric circuit diagram. In a 1–junction, all flow variables are equal while all effort variables add up to zero. The two junction types are shown in Fig.5.

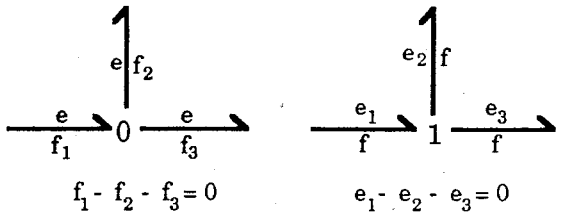

Figure 5. The two junction types.

The 0–junction thus represents Kirchhoff's current law, while the 1–junction represents Kirchhoff's voltage law. If a bond connects two junctions, one will always be of the 0–junction type, while the other is of the 1–junction type, i.e., in a bond graph, 0–junctions and 1–junctions toggle among each other. Neighboring junctions of the same gender can be combined into one.

The directions of the harpoons in Fig.3 were purposefully chosen such that they indicate the *direction of power flow.* In any physical system, power can be represented as the product of two variables, one of the across–type and the other of the through–type, thus:

$$P = e \cdot f \tag{16}$$

The bond graph shows clearly how the power is generated in the voltage source and then spreads through the circuit and gets absorbed by the passive components. Capacitors and inductors *store* the electrical power, i.e., as long as the signs of both voltage across and current through the element is the same, power flows into the element and is stored there. When the signs of voltage across and current through the element are opposite, the previously stored energy flows out of the element back into the circuit. Resistors *dissipate* the electrical power. Since the voltage across and the current through a resistor have always the same signs, power always flows into the resistor, never back out. The resistor is in fact *not* a one–port element at all. It is a transformer of electrical (or other) energy into thermal energy (i.e., heat). Consequently, the resistor should be represented by a two–port as shown in Fig.6,

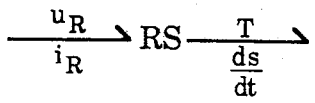

Figure 6. Enhanced bond graph of a resistor.

where the primary side can be either electrical, mechanical, hydraulic, pneumatic, or even thermal, whereas the secondary side is always thermal. The (electrical) power $u_R \cdot i_R$ flowing into the resistor equals the thermal power $T \cdot \dot{S}$ flowing out of the resistor. The representation of Fig.3 is a *simplification*, in that we chose not to model the thermal properties of the circuit, only its electrical properties.

Similarly, the effort source **SE** is a non–physical element since electrical power (or any other form of power for that matter) cannot be *generated.* The power has to come from somewhere (in our case out of a battery or out of a wall socket); however, we chose not to represent that portion of reality in our model.

4 Magic Heat Flow or How to Do Away With the Greenhouse Effect

Let us discuss the flow of heat through an ideally insulated rod. The temperature distribution along the rod can be modeled by the heat equation:

$$\frac{\partial T}{\partial t} = \sigma \frac{\partial^2 T}{\partial x^2} \tag{17}$$

which is a partial differential equation (PDE) in the two independent variables t (time) and x (space). One way to approximately solve this PDE is by discretizing the space axis x while leaving the time axis t continuous. We can approximate the second derivative in space through:

$$\frac{\partial^2 T(t, x_k)}{\partial x^2} \approx \frac{T(t, x_{k+1}) - 2T(t, x_k) + T(t, x_{k-1})}{\Delta x^2} \tag{18}$$

where x_k denotes any particular value x and $x_{k\pm1}$ are abbreviations for $x \pm \Delta x$. By applying this transformation, the PDE is reduced to a set of ordinary differential equations (ODEs) of the type:

$$\frac{dT_k(t)}{dt} = \frac{\sigma}{\Delta x^2}[T_{k+1}(t) - 2T_k(t) + T_{k-1}(t)] \tag{19}$$

where $T_k(t)$ denotes the temperature T at $x = x_k$ as a function of time. Equation (19) represents a state–space model of the heat equation. The advocated technique is referred to as the *method of lines.*

It is well known that the state–space model of Eq.(19) can be accurately described by the electrical analogon shown in Fig.7.

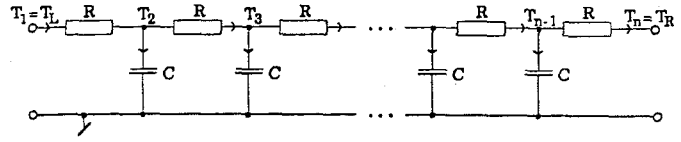

Figure 7. Electrical analogon of a diffusion chain.

A bond graph representation of this RC–chain is shown in Fig.8.

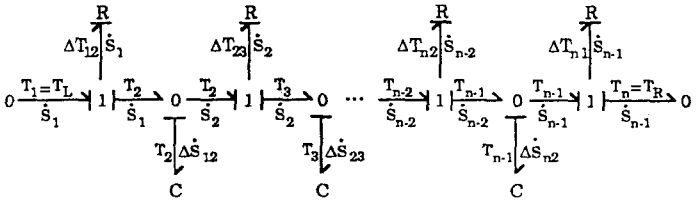

Figure 8. Bond graph of the diffusion chain.

However, the bond graph shown in Fig.8 still exhibits a problem. As in the electrical case, we seem to have resistances that dissipate heat and thereby lose energy. This is a rather dubious concept. Since we are dealing with thermic variables throughout, we have a problem. Where is the dissipated heat flowing to? In the case of the electrical circuit, it made somehow sense to ignore the thermal subsystem. However, in the case of the heat flow problem, we can't possibly do this. We must represent the secondary sides of the resistors in the model. Since the dissipated power cannot vanish, we simply reintroduce it right away at the next node, as shown in Fig.9.

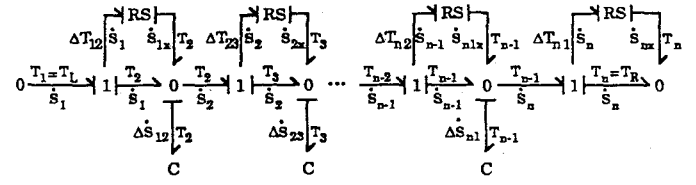

Figure 9. Corrected bond graph of the chain.

Notice that our modified bond graph is no longer exactly equivalent to the electrical circuit analogon. While the electrical circuit was able to represent the temperature distribution correctly, it failed to represent the power flow adequately. The bond graph representation helped to discover and eliminate the flaw in the model.

5 Experiences and Conclusions

Although bond graphs have been around for almost 30 years, I did not teach bond graphs in my modeling class until two years ago. I always liked bond graphs, but didn't find the technique very useful until I had decent software available that allowed me to code bond graphs elegantly and efficiently for use in a simulation program. While bond graph software has been around for some time also, I didn't like the available tools. They all lacked flexibility. DYMOLA [1,2,3] is clearly the software of choice.

I found that my students appreciate the bond graph methodology a lot. They grasp the concepts easily and quickly, they are highly motivated, and they work energetically on their homework assignments. I also found that the average student's understanding of physics has increased dramatically since the introduction of bond graphs to my class and since the introduction of the new text book [1]. My students are *much* less gullible than they used to be. They truly make efforts to come up with physically correct models. They understand much better than before the difference between a running simulation program and a valid model. The questions they raise in class prove that they have a much deeper understanding of what is going on.

References

[1] F. E. Cellier (1991). *Continuous System Modeling*, Springer–Verlag, New York, 755 p.

[2] F. E. Cellier (1992). "Hierarchical Non–linear Bond Graphs: A Unified Methodology for Modeling Complex Physical Systems," Simulation, February issue, in press.

[3] H. Elmqvist (1978). *A Structured Model Language for Large Continuous Systems*, Ph.D. Dissertation, Report CODEN: LUTFD2/(TRFT–1015), Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, 226 p.