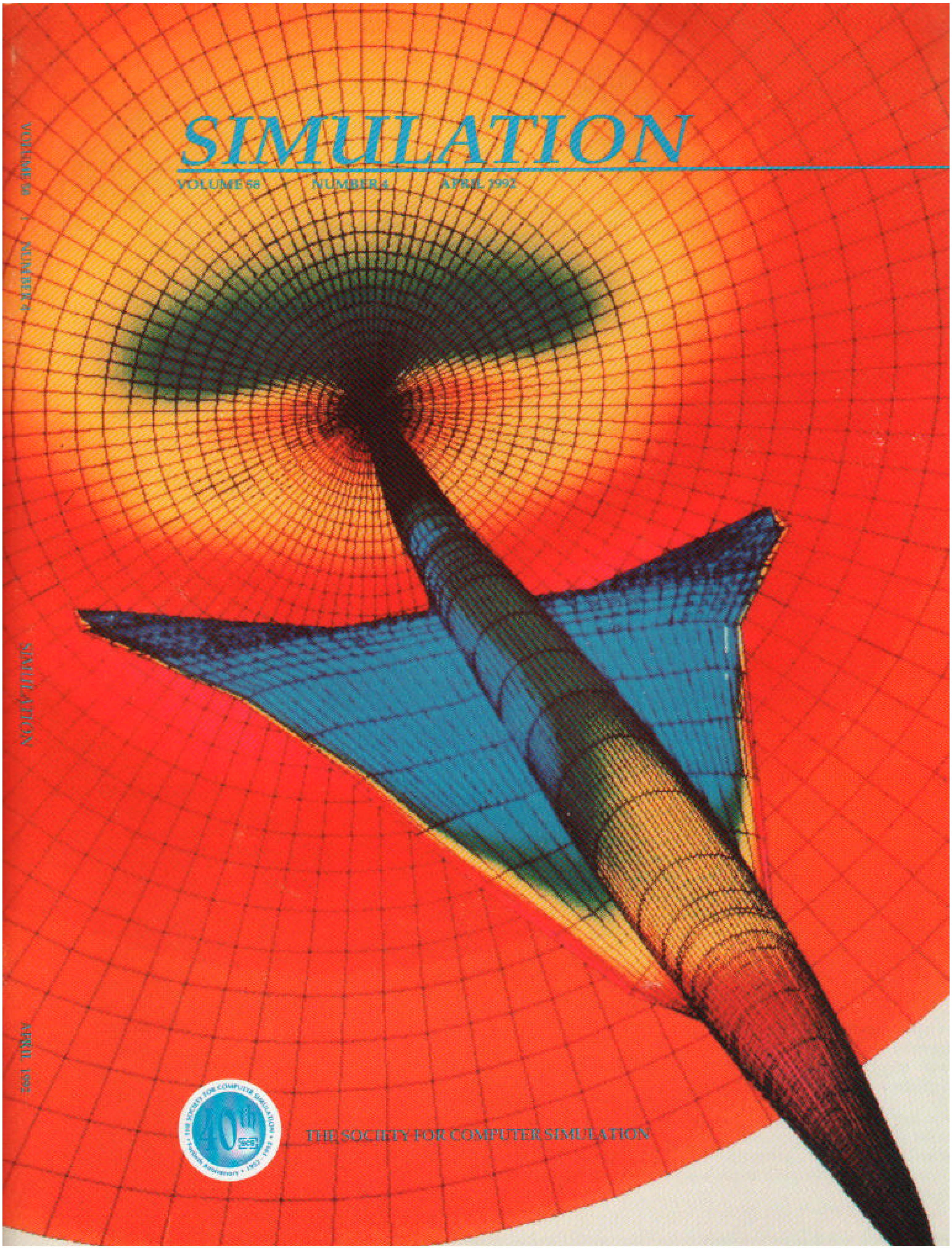
VOLUME 58 | NUMBER 4 SIMULATION APRIL 1992

SIMULATION

VOLUME 58 NUMBER 4 APRIL 1992



THE SOCIETY FOR COMPUTER SIMULATION

Publication and Subscription Information

Editorial & Circulation:
In U.S.A.: P.O. Box 17900, San Diego, California 92177. Telephone (619) 277-3888.
In Europe: Mr. Philippe Geril, University of Ghent, Dept. of Applied Mathematics & Biometrics, Coupure Links 653, 9000 Ghent, Belgium. Telephone 32.91.23.49.41

Subscriptions:
$112 per year, U.S. and U.S. Possessions.
End of year microfiche subscription, (12 issues) $110.

Single Copies:
$10.00 postpaid ($7.50 members).

Microfiches Available:
Vol. 1 (Fall 1963), $10; Vols. 2 through 49 (1964-1991), $75 per year (two vols. per year).

Membership:
Individual membership in The Society for Computer Simulation: $50 per year including subscription to *SIMULATION*. Individual membership not available to organizations.

Advertising:
In U.S.A.: Brian O'Neill, P.O. Box 17900, San Diego, California 92177. Telephone (619) 277-3888.
In Europe: Mr. Philippe Geril, University of Ghent, Dept. of Applied Mathematics & Biometrics, Coupure Links 653, 9000 Ghent, Belgium. Telephone 32.91.23.49.41

Publishing Information:
©1992 by Simulation Councils, Inc., P.O. Box 17900, San Diego, California 92177. All rights reserved. Published monthly. Second-class postage paid at San Diego, CA

***SIMULATION* Identification Statement**

SIMULATION (USPS 497020) ISSN 0037-5497/92
Published monthly by Simulation Councils, Inc., 4838 Ronson Court, Suite L, San Diego, California 92111.
Price: $112 per year
Second Class postage paid at San Diego, California.
Postmaster: Send address changes to *SIMULATION*, P.O. Box 17900, San Diego, California 92177

Abstracted and Indexed

SIMULATION is indexed and abstracted by
Applied Mechanics Reviews
Applied Science and Technology Index
Biostatistica
Computer and Information Systems
Computing Reviews
Current Contents
Data Processing Digest
Engineering Information, Inc.
Information Science Abstracts
Operations Research/Management Science
Quality Control and Applied Statistics
Science Abstracts
Sociological Abstracts

Change of Address

Send both old and new address to SCS, c/o Circulation Department, P.O. Box 17900, San Diego, California 92177. Please enclose old address label. Allow six weeks advance notice. Use subscription number on all correspondence.

Undelivered Copies

Please return to SCS, c/o Circulation Department, P.O. Box 17900, San Diego, California 92177.

SIMULATION

Volume 58: Number 4: April 1992

Technical papers

222 Simulating management's earnings-per-share forecasts
Alex B. Cameron

230 Hierarchical non-linear bond graphs: A unified methodology for modeling complex physical systems
François E. Cellier

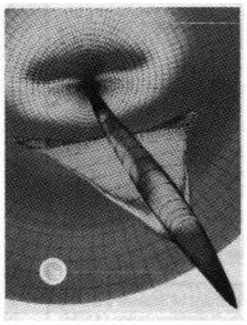
249 CSSL's and simulation of gas well behavior
Rudolf K. Stocker, P. Donald Ariel, John R. McRae, and Jaroslav Stanislav

258 A simulation technique for estimation in perturbed stochastic activity networks
V. G. Adlakha and H. Arsham

270 Development of design guidelines for local area CSMA/CD networks
Richard Cobb, Edward R. Mansfield, and Joseph M. Mellichamp

Departments

228 Book Review
229 Software Review
268 Industry News
269 AI and Simulation
280 Calendar of Events
282 Classified Ads
283 Simulation in the Service of Society
John McLeod



On the Cover . . .
Upper Surface View of a representative High Speed Civil Transport configuration under evaluation by McDonnell Douglas Corp. Shown is the predicted pressure field around the vehicle, using three-dimensional Computational Fluid Dynamics (CFD) analysis and a Cray XMP-18 supercomputer. High pressure regions on the wing leading edge and the fuselage nose are shown in red, and low pressure regions on the wing upper surface are in blue. Graphics were produced by ARTIS software, developed by McDonnell Douglas.

TECHNICAL ARTICLE

SIMULATION 58:4,230-248
© 1992, Simulation Councils, Inc.
ISSN 0037-5497/92 $3.00 + .10
Printed in the United States of America

Hierarchical non-linear bond graphs: a unified methodology for modeling complex physical systems

François E. Cellier
Department of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona 85721

Bond graphs have been around for a quarter of a century. While originally intended for modeling mechanical systems, they have meanwhile found widespread applications in many areas of physical system modeling. Bond graphs are a very appealing tool for modeling physical systems, because they represent the flow of power through a system. Since energy and mass are the only tradable goods in our physical universe, a bond graph model is more likely to reflect physical reality than a model derived by use of any other modeling methodology. However, bond graphs, like all graphical techniques, become unwieldy when applied to complex systems. Also, bond graphs were traditionally used to model predominantly linear systems. This paper introduces new concepts for modeling complex physical systems through hierarchical bond graphs which can include arbitrary non-linearities. It introduces a software tool that can be used to implement these hierarchical non-linear bond graphs. Finally, a new application area for bond graphs will be discussed. It will be demonstrated how these hierarchical non-linear bond graphs can be used to model chemical reaction kinetics and chemical thermodynamics together in very general terms also farther away from equilibrium than traditional approaches would permit.

Keywords: Bond graphs, model design, modeling software, physics, thermodynamics

Introduction

It has been known for a long time that the mathematics behind the dynamical behavior of systems from different disciplines of physical sciences have much in common. In particular, it has been recognized that mechanical systems can be treated by making use of *electro–mechanical analoga*, i.e. by treating masses, springs, and frictions as inductors, capacitors, and resistors.

This methodology was formalized in the early sixties through the introduction of so–called *bond graphs* [13]. A bond is a directed path that denotes the flow of power from one point of the system to another.

It turns out that, in *all* physical systems, power can be written as a product of two variables, one of which is an *across variable*, while the other is a *through variable*. Across variables around a node assume the same value, whereas through variables into a node add up to zero. The bond graph name of a node is a *0–junction*. In an electrical circuit, the potentials around a node assume all the same value, whereas the currents into a node add up to zero. Thus, if we identify the node of the electrical circuit with the 0–junction of the bond graph, the potentials will become across variables, while the currents will become through variables. The electrical power flowing through a circuit element i can be expressed as the product of the potential drop u_i and the current i_i:

$$P_{el} = u_i \bullet i_i \tag{1}$$

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. 1:

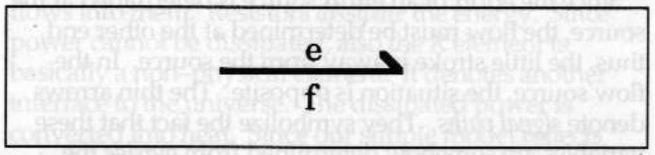


Figure 1. The bond

The bond graph literature is not systematic with respect to the bond graph conventions. The harpoon is sometimes shown to the left and sometimes to the right of the bond, and the effort variable is sometimes indicated on the side of the harpoon and sometimes away from the harpoon. This inconsistency can be explained by the fact that most bond graphers viewed the bond graph methodology as a pure modeling aid to be used with paper and pencil. The fact that a model represents always a *codified* form of knowledge occurred to them at best as an afterthought.

However, if we wish to use bond graphs as a tool to formalize a model and to formulate it as input to a computer program (as we shall do in this paper), we need to be more rigorous. For this purpose, we decided that the harpoon must sit always on the left of the bond, and the effort variable is always indicated on the side of the harpoon, while the flow variable is indicated on the side away from the harpoon.

The denomination of the effort and flow variables is arbitrary. If we had identified an electrical *mesh* with the 0–junction instead of the electrical *node*, the roles of the potentials and currents in the bond graph would have been interchanged, yet the power would still be the product of one effort variable and one flow variable.

Since electrical circuits, and all other physical systems, are described through a set of node (cutset) equations and a set of mesh (loop) equations, the bond graph must contain two types of junctions. The effort variables across a 0–junction assume the same value, whereas the flow variables into the 0–junction add up to zero. The flow variables across a 1–junction assume the same value, whereas the effort variables into the 1–junction add up to zero. The two function types are shown in Fig. 2.

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.

We are now ready to apply these concepts to a simple passive electrical circuit as shown in Fig. 3.

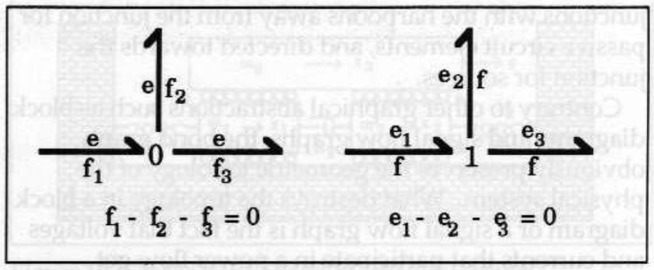


Figure 2. The two junction types

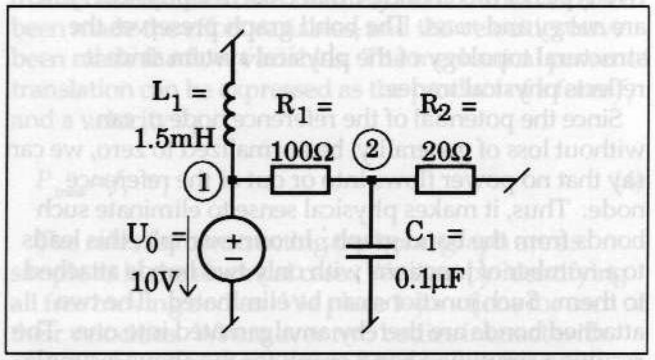


Figure 3. Circuit diagram of the passive electrical circuit

The bond graph for this circuit is shown in Fig. 4:

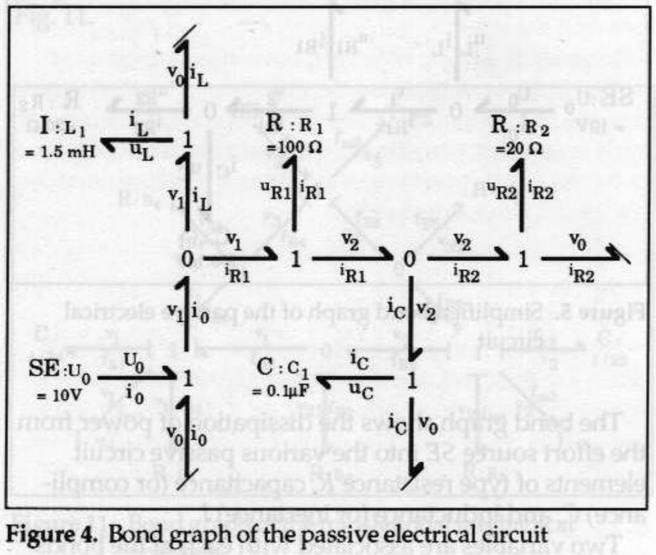


Figure 4. Bond graph of the passive electrical circuit

The rules for constructing a bond graph of an electrical circuit are very simple. We start by representing each circuit node by a 0–junction except for the reference node which is drawn like in the circuit diagram. We then represent each branch of the circuit diagram by a pair of bonds connecting two 0–junctions with a 1–junction between them. We let the harpoons point in the same direction that we picked for the branch currents. Finally, we attach the circuit elements to the 1–

junctions with the harpoons away from the junction for passive circuit elements, and directed towards the junction for sources.

Contrary to other graphical abstractions such as block diagrams and signal flow graphs, the bond graph obviously preserves the geometric topology of the physical system. What destroys the topology in a block diagram or a signal flow graph is the fact that voltages and currents that participate in a power flow get separated from each other. However, efforts and flows aren't tradable goods in a physical system. The only two types of merchandise that exist in a physical system are *energy* and *mass*. The bond graph preserves the structural topology of the physical system since it reflects physical trades.

Since the potential of the reference node v_0 can, without loss of generality, be normalized to zero, we can say that no power flows into or out of the reference node. Thus, it makes physical sense to eliminate such bonds from the bond graph. In our example, this leads to a number of junctions with only two bonds attached to them. Such junctions can be eliminated. The two attached bonds are thereby amalgamated into one. The resulting simplified bond graph for the above example is shown in Fig. 5.

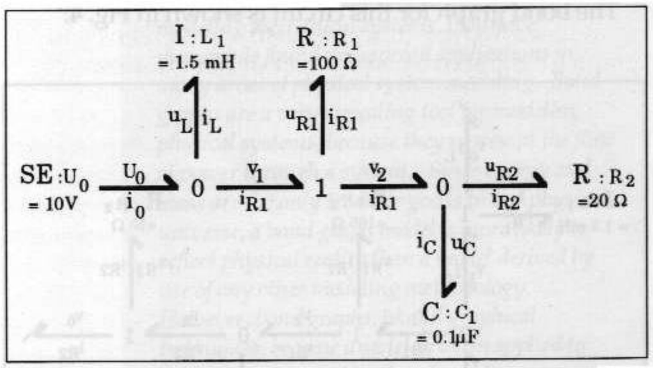


Figure 5. Simplified bond graph of the passive electrical circuit

The bond graph shows the dissipation of power from the effort source *SE* into the various passive circuit elements of type resistance *R*, capacitance (or compliance) *C*, and inductance (or inertance) *I*.

Two variables are associated with each of the bonds. Consequently, we need two equations in the resulting simulation program to evaluate them. It turns out that, in *all* bond graphs, one of these variables is always evaluated at each of the two ends of the bond. We can denote this fact by a little vertical stroke attached to one of the two ends of the bond. By convention, the stroke denotes the end where the flow variable is evaluated. In bond graph terms, this is called assigning a (computational) *causality* to the bond graph [10].

Sources have *mandated causalities* as shown in Fig. 6.

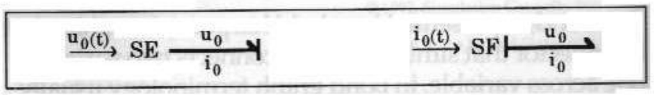


Figure 6. Mandated causalities for effort and flow sources

Since the effort of an effort source is determined at the source, the flow must be determined at the other end, thus, the little stroke is away from the source. In the flow source, the situation is opposite. The thin arrows denote *signal paths*. They symbolize the fact that these variables are somehow determined from *outside* the system. A source as drawn in the circuit example above is actually a non–physical element. Power cannot be generated, only transported and converted. However, a "system" never denotes the whole of the universe. It denotes a piece of the universe. Sources are interfaces between the system and the universe around it.

Capacitances and inductances have *desired causalities* as shown in Fig. 7.

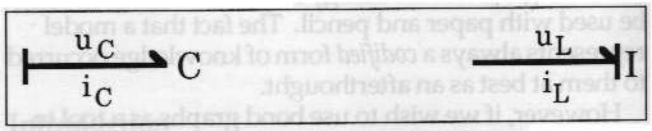


Figure 7. Desired causalities for capacitances and inductances

By declaring that we wish to compute the effort at the capacitance, we generate an equation of the type:

$$\frac{du_C}{dt} = \frac{1}{C} \cdot i_C \tag{2}$$

We thereby end up with a state equation in the state variable u_C which can be integrated. The reverse causality would have produced an integral equation for the variable i_C which would have forced us to compute the variable u_C somewhere else in the circuit by means of numerical differentiation. The situation for the inductance is analogous.

The resistance can assume either of two causalities as shown in Fig. 8.

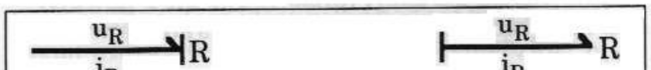


Figure 8. Possible causalities for resistors

The causality shown to the left leads to the equation:

$$i_R = \frac{1}{R} \cdot u_R \tag{3a}$$

whereas the causality shown to the right leads to the equation:

$$u_R = R \cdot i_R \tag{3b}$$

Capacitances and inductances *store* the energy that flows into them. Resistors *dissipate* the energy. Since power cannot be dissipated, also the *R* element is basically a non–physical element. It denotes another interface to the universe. The dissipated power is converted into heat. Since our simple model reflects only the electrical properties of reality, the generated heat is part of the *universe*, and not part of the *system*.

Since we add up the flows in a 0–junction, we can generate only one equation for the flows into a 0–junction. Consequently, only one of the bonds attached to a 0–junction can have its stroke at the side of the junction. Similarly, all bonds but one attached to a 1–junction must have their strokes at the side of the 1–junction.

These causality rules suffice to determine a unique causality for our simple passive electrical circuit. This is shown in Fig. 9.

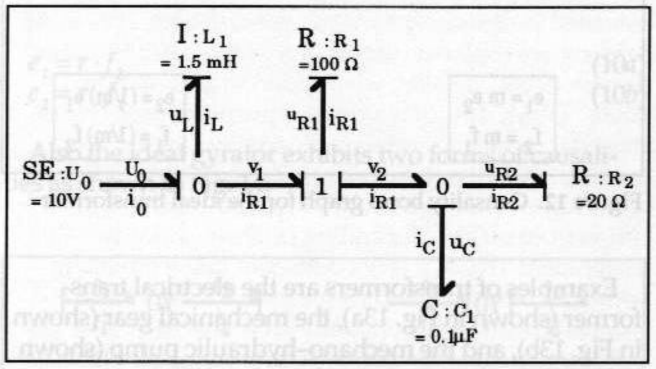


Figure 9. Causal bond graph of the passive electrical circuit

In this example, we were able to satisfy all causality constraints in a unique manner. This is the preferred situation. If not all mandated causality constraints can be satisfied, we are confronted with a *non–causal system*. This case occurs for example if we try to parallel connect two voltage sources with different voltage values. If we cannot satisfy all desired causality constraints, i.e. if we run into wrong causalities at either *C* or *I* elements, we are confronted with a *degenerate system* in which the true system order is lower than the number of energy storages makes us believe. If we have a choice in assigning causalities without offending any of the causality constraints, we have a system with one or several *algebraic loops*.

Mechanical systems can be modeled in a similar manner. Let us demonstrate this concept by means of the simple translational system shown in Fig. 10:

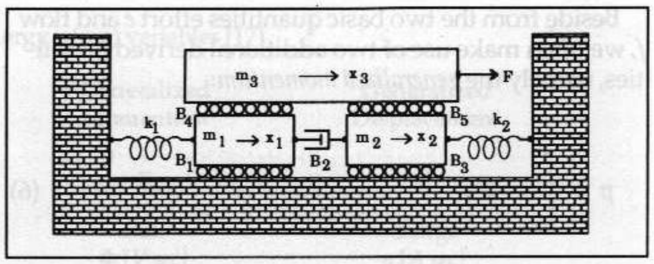


Figure 10. Simple translational mechanical system

In a translational mechanical system, the forces have been made the effort variables, and the velocities have been made the flow variables. The mechanical power of translation can be expressed as the product of a force f_i and a velocity v_i:

$$P_{trans} = f_i \cdot v_i \tag{4}$$

The rules for constructing the bond graph are as simple as in the electrical case. We start by identifying all free moving bodies. We place 1–junctions for each of their velocities. Where ever two bodies interact with each other, we connect their junctions with branches consisting of two bonds and one 0–junction in between, and attach all interacting elements to that 0–junction. Newton's law (or rather the d'Alembert principle) is formulated at the 1–junctions themselves. The bond graph for the above translational system is shown in Fig. 11.

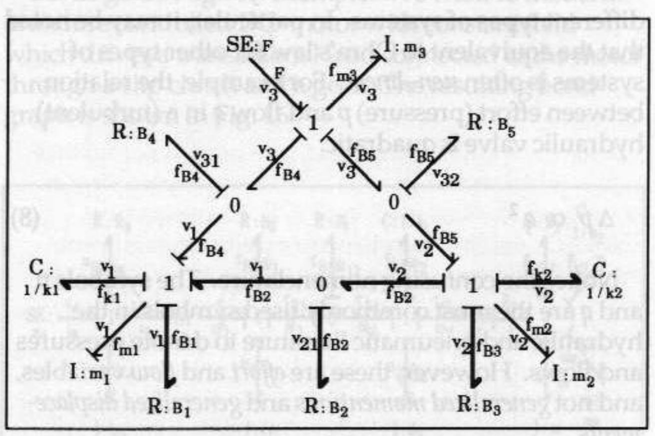


Figure 11. Bond graph of the translational mechanical problem

Rotational systems can be modeled in exactly the same way. Here, the effort variables are the torques τ_i, and the flow variables are the rotational velocities ω_i. The mechanical power of a rotation can be expressed as:

$$P_{rot} = \tau_i \cdot \omega_i \tag{5}$$

Beside from the two basic quantities effort *e* and flow *f*, we often make use of two additional derived quantities, namely the *generalized momentum*:

$$p = \int_0^t e \, d\tau \tag{6}$$

and the *generalized displacement*:

$$q = \int_0^t f \, d\tau \tag{7}$$

In electrical systems, the generalized momentum is the *flux* through a coil, and the generalized displacement is the *charge* in a capacitor. In translational mechanical systems, these are the *momentum* and the *displacement* (bond graphs were invented by mechanical engineers), and in rotational mechanical systems, they are the *angular momentum* and the *angular position*.

All these quantities are common to a large variety of other physical systems as well, as are the two Kirchhoff laws. Hydraulic, pneumatic, and acoustic systems operate similarly to the electrical and mechanical ones. In all these systems, the *pressure* is defined as the effort variable, while the *volume flow rate* is defined as the flow variable. The derived quantities are the *pressure momentum* and the *volume*.

The element laws, however, may look different for different types of systems. In particular, it may be noted that the equivalent to Ohm's law for other types of systems is often *non–linear*. For example, the relation between effort (pressure) *p* and flow *q* in a (turbulent) hydraulic valve is quadratic:

$$\Delta p \ \alpha \ q^2 \tag{8}$$

Notice the confusing nomenclature. The symbols *p* and *q* are the most commonly used symbols in the hydraulic and pneumatic literature to denote pressures and flows. However, these are *effort* and *flow* variables, and not *generalized momentums* and *generalized displacements*.

Table 1 presents a summary of the four generic variables for the most commonly used types of physical systems.

Until now, we have discussed different types of systems in isolation. However, one of the true strengths of the bond graph approach is the ease with which transitions from one form of system to another can be made, while ensuring that the energy (or power) conservation rules are satisfied. In an energy transducer (such as a transformer, or a DC–motor), the energy (or power) which is fed into the transducer is converted from one energy form to another, but it is never lost. Consequently, the energy that enters the transducer at one end must come out in one or more different form(s) at the other. A "loss–less" energy transducer may, for example, transform electrical energy into mechanical energy. In reality, every energy transducer "loses" some energy, but the energy does not really disappear – it is simply transformed into heat.

The above energy conservation law can be satisfied in exactly two ways in an "ideal" (i.e. loss–less) energy transducer. One such transducer is the *ideal transformer*. It is governed by the following set of relationships:

$$e_1 = m \cdot e_2 \tag{9a}$$

$$f_2 = m \cdot f_1 \tag{9b}$$

The ideal transformer is placed between two junctions. There are two types of causalities possible as shown in Fig. 12:

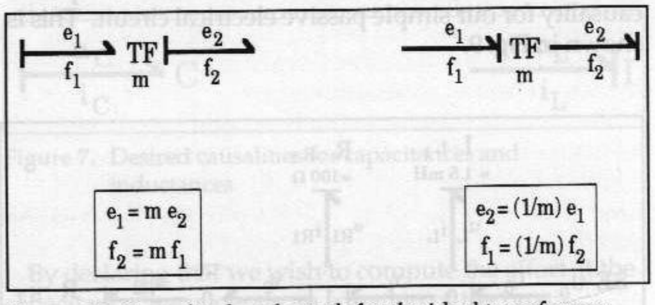


Figure 12. Causality bond graph for the ideal transformer

Examples of transformers are the electrical transformer (shown in Fig. 13a), the mechanical gear (shown in Fig. 13b), and the mechano–hydraulic pump (shown in Fig. 13c).

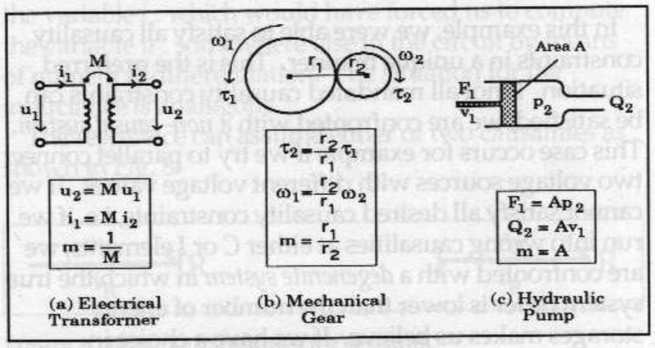


Figure 13. Examples of ideal transformers

The other type of energy transducer is the *ideal gyrator*. Its behavior is governed by the equations:

Table 1. Power – (e,f) and energy – (p,q) variables [17]

	Effort	Flow	Generalized Momentum	Generalized Displacement
	e	*f*	*p*	*q*
Electrical	voltage $u\,[V]$	current $i\,[A]$	flux $\Phi\,[V\,sec]$	charge $q\,[A\,sec]$
Translational	force $F\,[N]$	velocity $v\,[m\,sec^{-1}]$	momentum $I\,[N\,sec]$	displacement $x\,[m]$
Rotational	torque $T\,[N\,m]$	angular velocity $w\,[rad\,sec^{-1}]$	twist $T\,[N\,m\,sec]$	angle $\phi\,[rad]$
Hydraulic	pressure $p\,[N\,m^{-2}]$	volume flow $q\,[m^3 sec^{-1}]$	pressure momentum $\Gamma\,[N\,m^{-2}\,sec]$	volume $V\,[m^3]$
Chemical	chemical potential $\mu\,[J\,mole^{-1}]$	molar flow $\nu\,[mole\,sec^{-1}]$	–	number of moles $n\,[mole]$
Thermo–dynamical	temperature $T\,[°K]$	entropy flow $\frac{dS}{dt}\left[W\,°K^{-1}\right]$	–	entropy $S\,[J\,°K^{-1}]$

$$e_1 = r \cdot f_2 \qquad (10a)$$
$$e_2 = r \cdot f_1 \qquad (10b)$$

Also the ideal gyrator exhibits two forms of causalities as shown in Fig. 14:

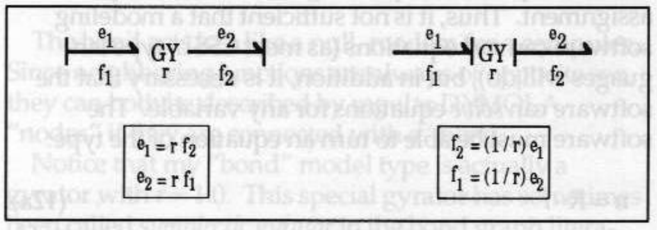


Figure 14. Causality bond graph for the ideal gyrator

Examples of gyrators are most electro–mechanical converters, such as the DC–motor shown in Fig. 15:

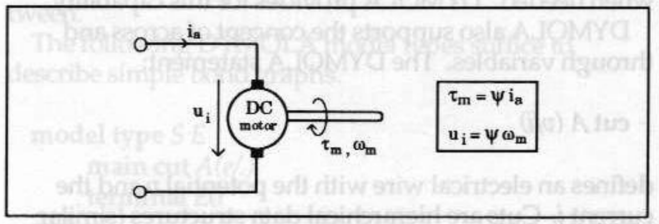


Figure 15. Example of an ideal gyrator

Notice that no real difference exists between the two transducer types [1,2]. If the effort and flow variables in the mechanical system were toggled, the DC–motor would in fact become a transformer.

Let us go through an example. We want to model an armature controlled DC–motor with constant field which drives a translational load connected to the motor through a slip clutch and a gear. The resulting bond graph is shown in Fig. 16.

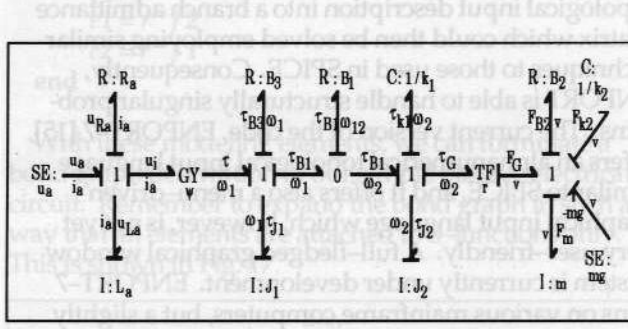


Figure 16. Bond graph of a DC–motor controlled mechanical system

The electrical power generated by the effort source is partly stored in the armature inductance L_a, and is partly dissipated in the armature resistance R_a. The remaining power is available for conversion into mechanical power. On the rotational side, the power is partly stored in the internal motor inertia J_1, and partly dissipated through friction B_3. The slip clutch reduces the angular velocity from ω_1 to ω_2. It contains the friction B_1. The remaining energy is partly stored in the external motor inertia J_2 which contains the primary cog of the gear. There is also a spring k_1 which prevents the axle from rotating too far. The gear itself is represented by a transformer. In this example, it converts the remaining rotational energy into translational energy. The load consists of a mass m, another spring k_2, a friction B_2, and the gravity force $m \cdot g$ (another source).

The bond graph shows clearly that the power available for conversion from the electrical side to the rotational mechanical side is:

$$P_{el} = u_i \cdot i_a \equiv P_{rot} = \tau \cdot \omega_1 \qquad (11)$$

The electrical power available for conversion is the product of the induced voltage u_i and the armature current i_a. This power is equal to the generated rotational power which can be described as the product of the motor torque τ and the angular velocity of the motor ω_1. This fact would not have been so easy to read out of a block diagram or a signal flow graph.

Notice the problem with the causality assignment. After satisfying all mandated causalities at the sources and junctions, we are confronted with a conflict. The translational inertance (the mass) m has the wrong causality. Thus, while this system exhibits six different energy storages, it is in fact a fifth order system.

Bond graph modeling in DYMOLA

The first bond graph simulation language written in the early seventies was ENPORT [14,15]. This software used an approach similar to SPICE. It did not request causalities to be specified, and it transformed the topological input description into a branch admittance matrix which could then be solved employing similar techniques to those used in SPICE. Consequently, ENPORT is able to handle structurally singular problems. The current version of the code, ENPORT–7 [15], offers an alphanumerical topological input language similar to SPICE, and it offers also a menu–driven graphical input language which, however, is not yet very user–friendly. A full–fledged graphical window system is currently under development. ENPORT–7 runs on various mainframe computers, but a slightly reduced version, ENPORT/PC, exists for IBM PC's and compatibles. ENPORT offers also a macro capability (somewhat comparable to the subcircuits in SPICE) which, however, does not provide for full hierarchical decomposition capabilities.

In the late seventies, another bond graph simulation language was developed at Twente University in the Netherlands, called THTSIM in Europe, and TUTSIM in the United States [17]. TUTSIM translates bond graphs into a state–space representation. The user is required to specify the causalities, and structurally singular systems cannot be handled. TUTSIM's simulation engine is somewhat poor in comparison with other state–space solvers such as ACSL. The same research group is currently prototyping a new bond graph modeling system, CAMAS [3], which runs on SUN's, has nice graphics capabilities, and is able to handle algebraic loops. CAMAS employs an object—oriented language (SIDOPS) for the model description. Once available, this might become a good product.

The third product on the market is CAMP [7,8], a preprocessor to ACSL [12] which translates bond graphs into ACSL programs. CAMP has the same limitations as TUTSIM. It does not handle algebraic loops or structural singularities, but it has the better simulation engine (ACSL). The input format is topological (as for the two other products). It is not truly flexible with respect to handling non–standard circuit elements. Non–linear elements need to be edited manually into the generated ACSL program which is inconvenient. A graphical front end exists meanwhile also for CAMP [9]. However as in the case of ENPORT–7, the graphics editor is menu–driven rather than window–operated.

With the exception of the unfinished CAMAS system, none of the above products is able to handle hierarchically structured models in a general fashion which is essential for the analysis of complex systems. For these reasons, we shall not discuss any of these programs in greater detail. Instead, we decided to go a different route. DYMOLA [6] is a general purpose hierarchical modular modeling software for continuous–time systems. As we noticed, elements –such as resistors– which allow different causalities to be applied lead to different equations depending on their causality assignment. Thus, it is not sufficient that a modeling software can *sort* equations (as most CSSL–type languages will do), but in addition, it is necessary that the software can *solve* equations for any variable. The software must be able to turn an equation of the type:

$$u = R \cdot i \qquad (12a)$$

into:

$$i = \frac{1}{R} \cdot u \qquad (12b)$$

when needed. DYMOLA provides for this capability.

DYMOLA also supports the concept of across and through variables. The DYMOLA statement:

cut *A (v/i)*

defines an electrical wire with the potential v and the current i. Cuts are hierarchical data structures (similar to PASCAL records) that enable the user to group individual wires into buses or cables, and cables into trunks. A cut is like a plug or a socket. It defines an interface to the outside world. The DYMOLA statement:

```
connect x:A at y:B
```

plugs the cut *A* of model *x* into the socket *B* of model *y*. Thereby, all the across variables (to the left of the slash separator) are set equal, and all the through variables (to the right of the slash operator) are summed up to zero. The DYMOLA preprocessor automatically generates the necessary coupling equations.

Nodes are a convenient means to organize connections. They act like your power distributor. You can plug several appliances into one such distributor. The above statement could also have been coded as:

```
node n
connect x:A at n
connect y:B at n
```

DYMOLA's *nodes* can be used as 0–junctions in a bond graph model. There is no DYMOLA equivalent for 1–junctions, but, as we explained before, 1–junctions are the same as 0–junctions with the effort and flow variables interchanged. Therefore, we created a model type "bond" which simply exchanges the effort and flow variables:

```
model type bond
    cut A (x/y) B (y/-x)
    main cut C [A B]
    main path P < A - B >
end
```

The bond acts just like a null–modem for a computer. Since neighboring junctions are always of opposite sex, they can both be described by regular DYMOLA "nodes" if they are connected with a "bond".

Notice that my "bond" model type is actually a gyrator with $r = 1.0$. This special gyrator has sometimes been called *symplectic gyrator* in the bond graph literature [1,2].

Since we don't want to maintain different types of *R*, *C*, *L*, *TF*, and *GY* elements, we add one additional rule: in DYMOLA, all elements (except for the bonds) can be attached to 0–junctions only. If they need to be attached to a 1–junction, we simply must place a bond in between.

The following DYMOLA model types suffice to describe simple bond graphs.

```
model type SE
    main cut A(e/.)
    terminal E0
end

model type SF
    main cut A(./-f)
    terminal F0
end

model type R
    main cut A(e/f)
    parameter R = 1.0
    R * f = e
end

model type C
    main cut A(e/f)
    parameter C = 1.0
    C*der(e) = f
end

model type I
    main cut A(e/f)
    parameter I = 1.0
    I*der(f) = e
end

model type TF
    cut A (e1/f1) B(e2/-f2)
    main cut C [A B]
    main path P < A - B >
    parameter m = 1.0
    e1 = m * e2
    f2 = m * f1
end

model type GY
    cut A (e1/f1) B(e2/-f2)
    main cut C [A B]
    main path P < A - B >
    parameter r = 1.0
    e1 = r * f2
    e2 = r * f1
end
```

With these modeling elements, we can formulate a bond graph description of our simple passive electrical circuit. Remember to expand the bond graph in such a way that all elements are attached to 0–junctions only. This is shown in Fig. 17.

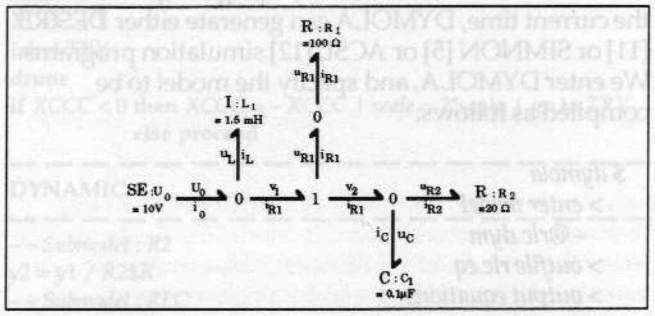


Figure 17. DYMOLA expanded bond graph of the passive circuit

The causalities were not marked down in Fig. 17 since DYMOLA is perfectly able to handle the causality assignment by itself .

The expanded bond graph can be immediately coded in DYMOLA as shown below:

```
@bond.bnd
@se.bnd
@r.bnd
@c.bnd
@i.bnd

model RLC

    submodel (S E) U0
    submodel (R) R1 (R = 100.0), R2 (R = 20.0)
    submodel (I) L1 (I = 1.5E – 3)
    submodel (C ) C 1 (C = 0.1E – 6)
    submodel (bond) B1, B2, B3
    node v1, ir1, vr1, v2
    input u
    output y1, y2

    connect U0 at    v1
    connect L1 at    v1
    connect R1 at    vr1
    connect R2 at    v2
    connect C1 at    v2
    connect B1 from v1 to ir1
    connect B2 from ir1 to v2
    connect B3 from ir1 to vr1

U0.E0 = u
y1 = C1.e
y2 = R2.f

end
```

The interpretation of this code is straightforward. DYMOLA's @ operator corresponds to the *include* statement of most programming languages. It includes the element definitions which were stored on separate files.

Let us see how the DYMOLA compiler preprocesses this code. DYMOLA is actually a *program generator*. At the current time, DYMOLA can generate either DESIRE [11] or SIMNON [5] or ACSL [12] simulation programs. We enter DYMOLA, and specify the model to be compiled as follows:

```
$ dymola
 > enter model
 – @rlc.dym
 > outfile rlc.eq
 > output equations
```

The *enter model* statement reads in the model and immediately expands the set of equations by the coupling equations. The *outfile* statement specifies the name of the output file, and the *output equations* statement writes the generated equations to the output file. The generated equations are shown in the next code segment.

```
U0       E0 = e
R1       R * f = e
C1       C * dere = f
L1       L * derf = e
R2       R * f = e
RLC      U0.E0 = u
         y1 = C1.e
         y2 = R2.f
         L1.e = B1. x
         U0.e = L1.e
         C1.e = B2.y
         R2.e = C1.e
         C1.f + R2.f = B2.x
         B2.x = B3.x
         B1.y = B2.x
         B3.y + B2.y = B1.x
         R1.e = B3.y
         R1.f = B3. x
```

The first eight equations were extracted from the models. The remaining equations are automatically generated coupling equations.

We can now execute the algorithm which assigns the causalities, i.e. which determines what variable to compute from each of the equations. In DYMOLA, this is achieved with the following set of instructions:

```
> partition
> outfile rlc.sor
> output sorted equations
```

which results in the following answer:

```
RLC      [U0.E0] = u
U0       E0 = [e]
RLC      U0.e = [L1.e]
         L1.e = [B1.x]
         C1.e = [B2.y]
         [B3.y] + B2.y = B1.x
         [R1.e] = B3.y
R1       R * [f] = e
RLC      R1.f = [B3.x]
         [B2.x] = B3.x
         [B1.y] = B2.x
         [R2.e] = C1.e
R2       R * [f] = e
RLC      [C1.f] + R2.f = B2.x
C1       C * [dere] = f
L1       L * [derf] = e
```

```
RLC        [y1] = C1.e
           [y2] = R2.f
```

This time, we decided to print out the sorted equations. The variables enclosed in "[ ]" are the variables for which each equation must be solved. This set of equations contains many trivial equations of the type *a* = *b*. DYMOLA is capable of throwing those out. This is accomplished through the following set of instructions:

```
> partition eliminate
> outfile rlc.sr2
> output sorted equations
```

which results in the following answer:

```
R2         R * [y2] = y1
RLC        [B3.y] + y1 = u
R1         R * [B3.x] = B3.y
RLC        [C1.f] + y2 = B3.x
C1         C * [dere] = f
L1         L * [derf] = u
```

which is a much reduced set of equivalent equations. The next step will be to actually perform the symbolic manipulation on the equations. In DYMOLA, this is done in the following way:

```
> outfile rlc.sov
> output solved equations
```

which results in the following answer:

```
R2         y2 = y1 / R
RLC        B3.y = u - y1
R1         B3.x = B3.y/R
RLC        C1.f = B3.x - y2
C1         dere = f/C
L1         derf = u/L
```

We are now ready to add the experiment description to the model. We can for instance use the following experiment:

```
cmodel
     simutime 2E - 5
     step 2E - 7
     commupoints 101
     input 1, u (independ, 10.0)

ctblock
     scale = 1
     XCCC = 1
     label TRY
     drunr
     if XCCC < 0 then XCCC = - XCCC |
     scale = 2 * scale | go to TRY else proceed
ctend

outblock
     OUT
     displ y1, y2
outend
end
```

This portion of code is specific for each of the target languages. The here shown version is the one required for DESIRE [11]. The *ctblock* set of statements instructs DESIRE to automatically scale the run-time display. XCCC is a DESIRE variable which is set to –1 whenever the DESIRE program is interrupted with an "overflow". This happens when one of the displayed variables hits either the top or the bottom of the displayed window. At this time, the plot is simply rescaled, and rerun with a new *drunr* statement. Since DESIRE is so fast, it is not worth the effort to store the results of the previous attempt, instead, we simply rerun the entire simulation.

The set of DYMOLA instructions:

```
> enter experiment
  - @rlc.ctl
> outfile rlc.des
> output desire program
```

tells DYMOLA to generate the following DESIRE program:

```
-----------------------------------------------
-- CONTINUOUS SYSTEM RLC
-----------------------------------------------
-- STATE y1 L1$f
-- DER dC1$e dL1$f
-- PARAMETERS and CONSTANTS:
R1$R = 100.0
C = 0.1E - 6
L = 1.5E - 3
R2$R = 20.0
-- INITIAL VALUES OF STATES:
y1 = 0
L1$f = 0
u = 10.0
-----------------------------------------------
TMAX = 2E - 5 | DT = 2E - 7 | NN = 101
scale = 1
XCCC = 1
label TRY
drunr
if XCCC < 0 then XCCC = - XCCC | scale = 2*scale | go to TRY
           else proceed
-----------------------------------------------
DYNAMIC
-----------------------------------------------
-- Submodel : R2
y2 = y1 / R2$R
-- Submodel : RLC
B3$y = u - y1
-- Submodel : R1
```

```
B3$x = B3$y/R1$R
-- Submodel : RLC
C1$f = B3$x - y2
-- Submodel : C1
d/dt y1 = C1$f/C
-- Submodel : L1
d/dt L1$f = u/L
```

```
OUT
dispt y1, y2
```

```
/ --
/ PIC 'rlc.PRC'
/ --
```

which can be executed at once using the following instructions:

```
   > stop
$ desire
   > load 'rlc.des'
   > run
```

which will immediately (within less than a second) produce the desired output variables u_C, and i_{R2} on the screen. Both DYMOLA [6] and DESIRE [11] are currently running alternatively on VAX/VMS, PC/MS–DOS or SUN/UNIX.

Thermodynamic bond graphs

Thermal conductive or convective flow of heat can be described by the heat equation:

$$\frac{\partial T}{\partial t} = \sigma \cdot \nabla^2 T \tag{13}$$

Heat flow in one space direction can be modeled by a simple *RC* chain as shown in Fig. 18.

Using the bond graph methodology, the *RC* chain can be represented as shown in Fig. 19.

This bond graph demonstrates that the approach is

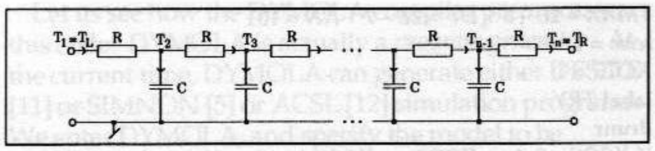


Figure 18. Electrical circuit analogon of a diffusion chain

not sound. When modeling electrical or mechanical phenomena, we could afford to model the dissipation of power through resistive elements. However, we have seen already that this is basically a non–physical concept. Power cannot be dissipated, only transported and converted. When modeling thermodynamic

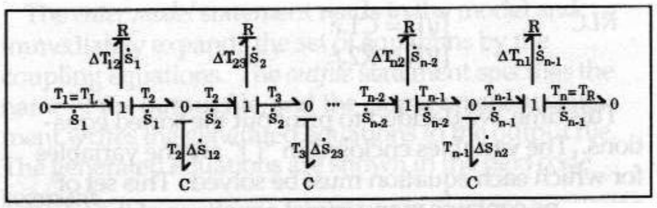


Figure 19. Causal bond graph of the diffusion chain

systems, the idea becomes absurd. What does it mean when we say that thermal power is being dissipated? This is obviously meaningless. Heat gets "absorbed" by the resistor, but it is immediately "generated" again by the same resistor.

To overcome this difficulty, Thoma introduced a new bond graph element called a *resistive source* [16]. The heat which is absorbed by the resistive element is routed through the resistive source, and is immediately re—introduced at the next junction. The modified bond graph is shown in Fig. 20.

The enhanced bond graph still represents the heat equation, but while the previously suggested bond graph models the temperature flow only, the enhanced

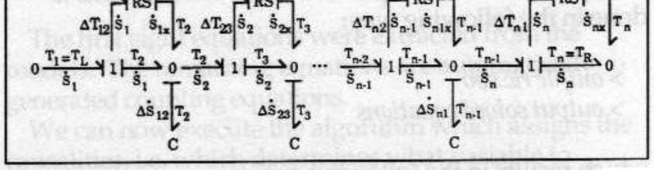


Figure 20. Corrected causal bond graph for the diffusion chain

bond graph models also the power flow correctly which can be written as:

$$P_{therm} = T \cdot \dot{S} \tag{14}$$

Thermal power is the product of the temperature T and the entropy flow $\dot{S}$.

In DYMOLA, the *RS* element can be represented as follows:

```
model type RS
    cut A(e1/f1) B(e2/-f2)
    main cut C [A B]
    main path P < A - B >
    parameter R = 1.0
    R * f1 = e1
    e1 * f1 = e2 * f2
end
```

There is however still one problem to be considered. Contrary to the electrical and mechanical systems, the *R* and *C* element of thermodynamics are not constant. The *R* element is proportional to the temperature:

$$R = \theta \cdot T \tag{15}$$

whereas the C element is inverse proportional to the temperature:

$$C = \frac{\gamma}{T} \tag{16}$$

Thermodynamic *R* and C elements are non–linear. To overcome this problem, earlier bond graph software systems introduced the *heat flow* as the flow variable instead of the *entropy flow*. This makes the *R* and *C* elements linear, but, at the same time, it makes it difficult to interface the so–called *thermodynamic bond graphs* with other bond graphs in a mixed energy system, since the product of temperature and heat flow does not represent power.

In DYMOLA, we don't have this problem, since we can model non–linearities easily. We simply introduce two new bond graph elements, the *modulated resistive source: mRS*, and the *"modulated" capacitance: mC*

```
model type mRS
    cut A(e1/f1) B (e2/ -f2)
    main cut C [A B]
    main path P < A - B >
    parameter theta = 1.0
    R = theta * e2
    R * f1 = e1
    e1 * f1 = e2 * f2
end

model type mC
    cut A( e/f )
    parameter gamma = 1.0
    C = gamma/e
    C*der(e) = f
end
```

Notice that, in the *mRS* model, θ must be multiplied by e_2 and not by e_1 since e_2 denotes an absolute temperature whereas e_1 denotes a temperature difference.

Notice further that the "modulation" of a capacitance is a rather dubious undertaking. How do we ensure that the "modulated" capacitance is still an energy storage element, and does not suddenly start to dissipate energy? This problem requires some further contemplation.

The energy stored in a capacitor (or inductor) is the integrated power that flows into that capacitor (or inductor), thus:

$$E(t) = \int_0^t P(\tau)\,d r = \int_0^t e(\tau) \cdot f(\tau)\,d r \tag{17}$$

whereby the energy for $t = 0.0$ has arbitrarily been normalized to zero. Using the formula for the general displacement (the charge) of the capacitor:

$$q(t) = \int_0^t f(\tau)\,d r \tag{18}$$

we can write:

$$E(t) = \int_0^t e(\tau) \cdot \dot{q}(\tau)\,d\tau = \int_0^q e(q)\,dq \tag{19}$$

Thus, in order for an element to behave like a capacitor, the effort *e* must be expressible as a (possibly non–linear) function of *q*:

$$e_C = \Phi_C(q_C) \tag{20}$$

Similarly, we can use the formula for the generalized momentum (the flux) of an inductor:

$$p(t) = \int_0^t e(\tau)\,d\tau \tag{21}$$

Therefore, in order for an element to behave like an inductor, the flow *f* must be expressible as a (possibly non–linear) function of *p*:

$$f_I = \Phi_I(p_I) \tag{22}$$

Let us check whether our "modulated" capacitance satisfies eqn. (20). We know that:

$$f_C(t) = C \cdot \dot{e}_C(t) = \frac{\gamma}{e_C(t)} \cdot \dot{e}_C(t) \tag{23}$$

and therefore:

$$q_C(t) = \int_0^t f_C(t)\,d\tau = \gamma \int_0^t \frac{\dot{e}_C(t)}{e_C(t)}\,d\tau = \gamma \Diamond \log(e_C) \tag{24}$$

The capacitive charge q_C is indeed a non–linear function of the effort e_C, and the capacitive nature of our "modulated" capacitance has thus been verified.

Let me now explain how the bond graph modeling concept can be used in a hierarchical fashion. For this purpose, we shall study a considerably more complex example: a solar heated house. The overall configuration is shown in Fig. 21.

One or several collectors act as black bodies which

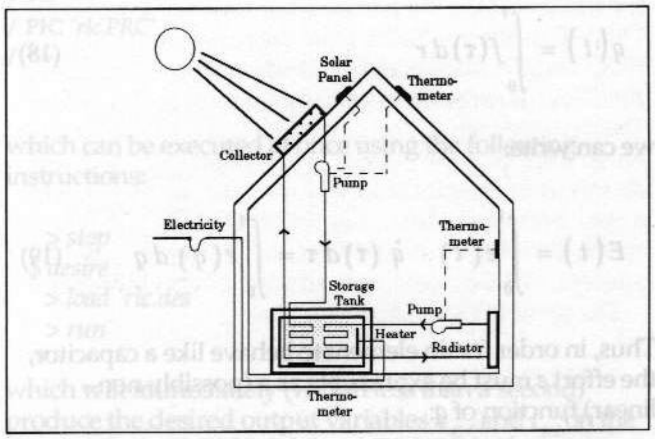


Figure 21. A solar heated house

absorb incoming solar radiation. Consequently, the temperature inside the collectors raises. The collectors can be filled with any material with a large heat capacity. Usually, it is simply air. Inside the collectors, there is a water pipe which meanders back and forth between the two ends of the collector to maximize the exposed pipe surface. We shall call this a "water spiral". A (mostly conductive) heat exchange takes place between the collector chamber and the water pipe, thereby heating the water in the pipe. A pump circulates the water from the collectors to the storage tank, thereby transporting the heat convectively from the collectors to the tank. We call this the "collector water loop". The water spirals in the various collectors can be either series connected, or they can be connected in parallel. The pump is usually driven by a solar panel. In the panel, the solar light is converted to electricity which drives the pump. Thereby, the pump circulates the water only while the sun is shining which is exactly what we want. In addition, a freeze protection device is often installed which also switches the pump on whenever the outside temperature falls below 5°C.

The storage tank is often realized simply as a large and well insulated water container (a water heater). However, we shall assume that a solid body storage tank is used together with another water spiral which deposits the heat in the storage tank just the same way as it was picked up in the collectors. Consequently, the water from the collector loop and from the heater loop never mix.

Inside the storage tank, there is a second water spiral which belongs to the "heater water loop", and which can pick up the heat from the storage tank. There is also an additional electrical heater installed which heats the storage tank electrically whenever the storage tank temperature falls below a critical value.

The heater water loop is driven by another pump which is switched on whenever the room temperature falls below 20°C during the day or 18°C during the night, and which is switched off whenever the room temperature raises beyond 22°C during the day or 20°C during the night.

In the house, we use one or several "radiators" (more water spirals) which, contrary to what their name suggests, exchange heat with the room in a partly conductive and partly convective manner.

Fig. 22 depicts the collector in more detail.

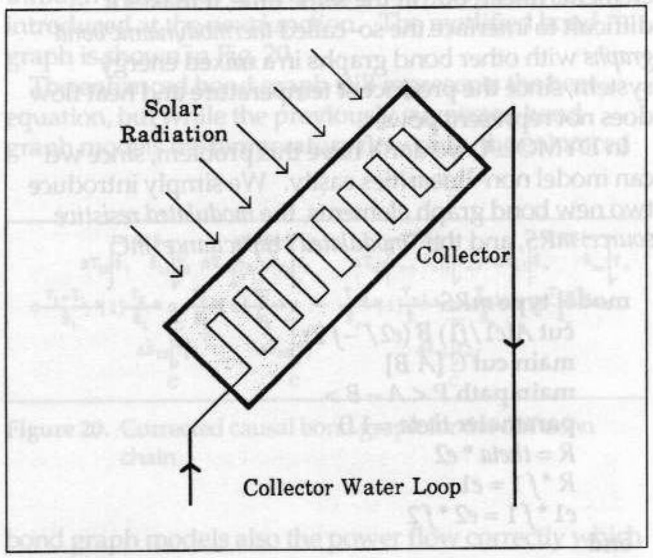


Figure 22. The solar collector

The water spiral is modeled through a series of one–dimensional cells. We want to model each such cell as shown in Fig. 23.

Each cell is described by a DYMOLA model type called *c1d.dym* which, from now on, can be used as an additional bond graph element. The correct causalities have been marked on the graph. The *mGS* element is a "modulated conductive source". It is modulated with

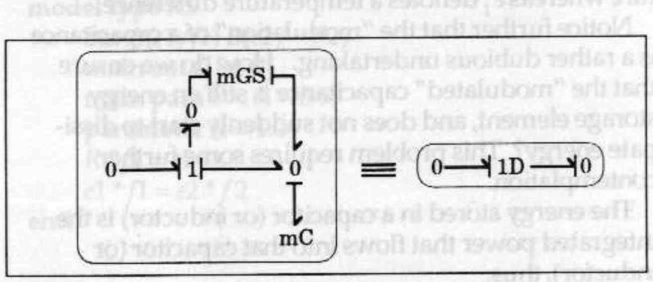


Figure 23. Bond graph of a one-dimensional cell

temperature (as always in thermal systems), but, in addition, it is also modulated with the water velocity in the pipe as shown in Fig. 24.

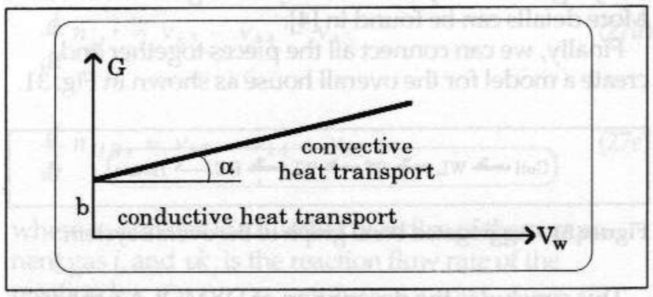


Figure 24. Modulated conductive source

Since the conductance changes linearly with the water velocity v_w, I preferred to model this element through its conductance rather than through its resistance. The c1*d* model references three submodels: a temperature modulated capacitance *mC*, a temperature and water velocity modulated conductive source *mGS*, and finally the regular *bond* submodel.

The exchange of heat across the border of two media is modeled by a *heat exchanger* as shown in Fig. 25.

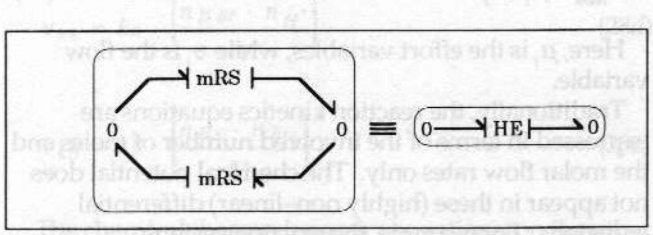


Figure 25. Bond graph of a heat exchanger

The heat exchanger is used here to model the transfer of heat from the collector chamber to the water spiral.

The water spiral is modeled through a series connection of several c1*d* elements with heat exchangers attached in between. The water spiral is shown in Fig. 26.

We decided to cut the spiral into three discrete links. Obviously, this is an approximation of a process with distributed parameters.

Notice that the newly introduced bond graph symbol representing the water spiral is a *3–port element*.

We need to model also the loss from the collector chamber to the environment. This loss is partly conductive and partly convective. The loss element (a 1–port element) is shown in Fig. 27.

The effort source denotes the outside temperature. The *mG* element denotes the heat dissipation to the environment. The dissipated heat is proportional to the difference in temperatures between the inside and the outside. *mG* is a modulated conductance similar to the *mGS* element found earlier, but this time, the secondary

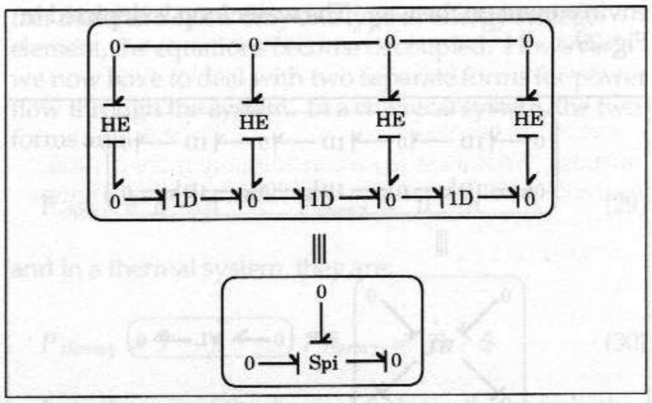


Figure 26. Bond graph of a water spiral

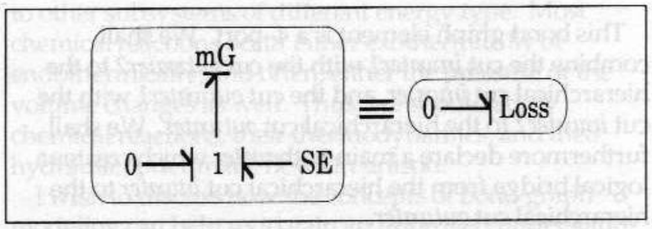


Figure 27. Bond graph of thermic loss

port (the environment) is not modeled, and the modulation is now with respect to the wind velocity v_{wind} rather than with respect to the water velocity v_w.

We are now ready to model the overall collector. It is shown in Fig. 28.

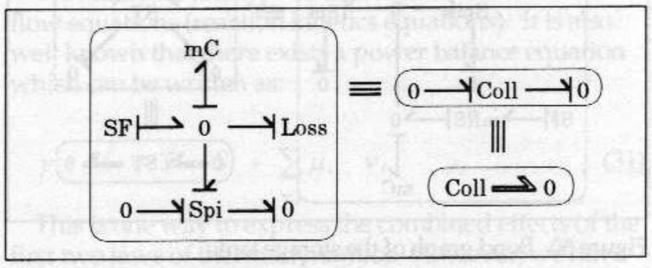


Figure 28. Bond graph of the collector

The *mC* element is the (temperature modulated) heat capacitance of the collector chamber. The *SF* element denotes the heat input from solar radiation.

We used the hierarchical cut concept of Dymola to combine the two cuts (i.e. bonds), *inwater* and *outwater*, into one hierarchical cut, *water*. This can be pictorially represented by a double bond. This *aggregated bond graph representation* has, of course, the disadvantage that causalities can no longer be depicted.

We shall now model the transport of heat from the collector to the storage tank, i.e. the collector water loop. We model each of the pipes by a series of one-dimensional cells, and we shall assume that the pipes are thermically well insulated, i.e., that no heat is lost to the environment on the way. The water loop is depicted in Fig. 29.

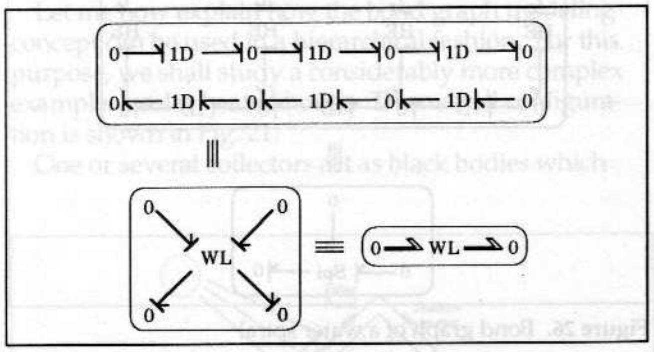


Figure 29. Bond graph of the water loop

This bond graph element is a 4–port. We shall combine the cut *inwater1* with the cut *outwater2* to the hierarchical cut *inwater*, and the cut *outwater1* with the cut *inwater2* to the hierarchical cut *outwater*. We shall furthermore declare a main path *water* which creates a logical bridge from the hierarchical cut *inwater* to the hierarchical cut *outwater*.

The storage tank contains two water spirals, one which belongs to the collector water loop, and one which belongs to the heater water loop. In addition, there has been installed an electrical resistance heater as a backup device. The storage tank is shown in Fig. 30.

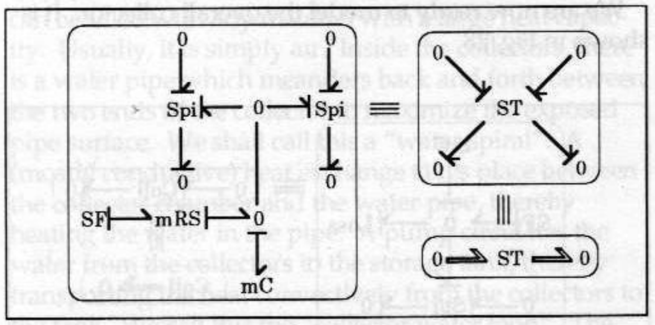


Figure 30. Bond graph of the storage tank

The *mC* element denotes the heat capacity of the storage tank. The flow source together with the *mRS* element denote the electrical backup heater. The primary side of the resistive source is electrical while the secondary side is thermic.

This is another 4–port. This time, we shall combine the cut *inwater1* with the cut *outwater1* to the hierarchical cut *inwater*, and the cut *outwater2* with the cut *inwater2* to the hierarchical cut *outwater*. We shall again declare a main path *water* which creates a logical bridge from the hierarchical cut *inwater* to the hierarchical cut *outwater*.

The heater water loop is modeled in exactly the same manner as the collector water loop.

The house itself is a little more tricky. We can model the house through a number of *three–dimensional cells* with loss elements attached to the walls, and heat exchangers denoting the heat input through the radiators. However, space limitations will prevent me from carrying this example all the way through to the end. More details can be found in [4].

Finally, we can connect all the pieces together and create a model for the overall house as shown in Fig. 31.

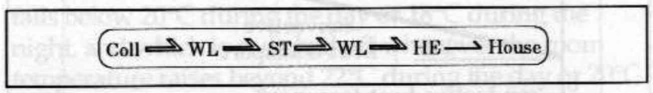


Figure 31. Aggregated bond graph of the overall system

This concludes the description of DYMOLA's modular hierarchical non–linear bond graph modeling capability.

Chemical reaction kinetics and chemical thermodynamics

Chemical power can be expressed as the product of the chemical potential μ_i of a chemical species i multiplied by its molar flow rate υ_i:

$$P_{chem} = \mu_i \cdot \upsilon_i \tag{25}$$

Here, μ_i is the effort variables, while υ_i is the flow variable.

Traditionally, the reaction kinetics equations are expressed in terms of the involved number of moles and the molar flow rates only. The chemical potential does not appear in these (highly non–linear) differential equations. For example, the reaction system:

$$Br_2 \xrightarrow{k_1} 2Br^\bullet \tag{26a}$$

$$2Br^\bullet \xrightarrow{k_2} Br_2 \tag{26b}$$

$$Br^\bullet + H_2 \xrightarrow{k_3} H\,Br + H^\bullet \tag{26c}$$

$$H\,Br + H^\bullet \xrightarrow{k_4} Br^\bullet + H_2 \tag{26d}$$

$$Br_2 + H^\bullet \xrightarrow{k_5} H\,Br + Br^\bullet \tag{26e}$$

can be described by the set of differential equations:

$$\frac{d}{dt} n_{Br_2} = -\nu_{k1} + \nu_{k2} - \nu_{k5} \tag{27a}$$

$$\frac{d}{dt} n_{Br^\bullet} = 2\nu_{k1} - 2\nu_{k2} - \nu_{k3} + \nu_{k4} + \nu_{k5} \tag{27b}$$

$$\frac{d}{dt} n_{H_2} = - v_{k3} + v_{k4} \tag{27c}$$

$$\frac{d}{dt} n_{H^\bullet} = v_{k3} - v_{k4} - v_{k5} \tag{27d}$$

$$\frac{d}{dt} n_{HBr} = v_{k3} - v_{k4} + v_{k5} \tag{27e}$$

where n_i denotes the number of moles of the component gas i, and vk_i is the reaction flow rate of the reaction k_i:

$$v_{k1} = k_1 \cdot n_{Br_2} \tag{28a}$$

$$v_{k2} = k_2 \cdot \left(\frac{n^2_{Br^\bullet}}{V} \right) \tag{28b}$$

$$v_{k3} = k_3 \cdot \left(\frac{nH_2 \cdot n_{Br^\bullet}}{V} \right) \tag{28c}$$

$$v_{k4} = k_4 \cdot \left(\frac{n_{HBr} \cdot n_{H^\bullet}}{V} \right) \tag{28d}$$

$$v_{k5} = k_5 \cdot \left(\frac{n_{H^\bullet} \cdot n_{Br_2}}{V} \right) \tag{28e}$$

The chemical thermodynamics equations, on the other hand, i.e. the equations which can be used to determine the chemical potential, aren't commonly handled by means of differential equations at all. Chemical thermodynamics is in fact a misnomer. Chemical thermostatics would be a more appropriate term. Thermodynamics, if at all, are only treated through small signal aberrations from the steady–state. This is because the *true* thermodynamics equations are still poorly understood.

Thus, the "kineticists" deal with mass flow only. They ignore the power flow altogether. The "thermodynamicists" deal with the energy balance under equilibrium conditions, and they actually ignore both the mass flow and the power flow.

How come that this decomposition is possible? For example, in electrical systems, we weren't able to decompose the dynamics into one set of equations involving currents only, and another set involving potentials exclusively. Why is it that both thermal and chemical systems exhibit this decomposition property? Remember that also the heat equation can be formulated either in terms of temperature or in terms of entropy alone. We discovered that this astounding property can be derived from the fact that neither thermal nor chemical systems have an *inertia* element. Due to the fact that there exists only one type of energy storage element, the equations become decoupled. However, we now have to deal with two separate forms for power flow through the system. In a chemical system, the two forms are:

$$P_{chem_1} = \mu \cdot \dot{n} \quad , \quad P_{chem_2} = \dot{\mu} \cdot n \tag{29}$$

and in a thermal system, they are:

$$P_{therm_1} = T \cdot S \quad , \quad P_{therm_2} = \dot{T} \cdot S \tag{30}$$

As in the case of the thermal systems, this seemingly convenient decomposition property has a considerable disadvantage. Such models cannot easily be connected to other subsystems of different energy type. Most chemical reactions occur either exothermically or endothermically, and often, either the pressure or the volume changes as well. Thus, we should study chemical reactions, their thermodynamics, and their hydraulic/pneumatic flows in unison.

I wish to discuss how the concepts of bond graph modeling can help us to gain an improved understanding of what is happening in these highly complicated systems. We start by noticing that our combined dynamics will contain six types of variables: (i) the temperature T, (ii) the entropy S, (iii) the hydraulic/pneumatic pressure p, (iv) the volume V, (v) the chemical potential μ, and (vi) the number of moles n. Consequently, we need six types of equations to describe the dynamics of this system in their entirety.

We have met one type of equation so far: the mass flow equations (reaction kinetics equations). It is also well known that there exists a power balance equation which can be written as:

$$p \cdot \dot{V} = T \cdot \dot{S} + \sum_{\forall i} \mu_i \cdot v_i \tag{31}$$

This is one way to express the combined effects of the first two laws of thermodynamics. However, we have discovered that chemical systems obey a second type of power balance equation as well:

$$\dot{p} \cdot V = \dot{T} \cdot S + \sum_{\forall i} \dot{\mu}_i \cdot n_i \tag{32}$$

Eqn. (32) is a generalization of the so–called Gibbs-Duhem equation. The detailed derivation of this formula is in [4].

Furthermore, we realize that energy is traded between the chemical system and its thermal and hydraulic/pneumatic environments. Thus, there must exist some sort of transformer with bonds into the thermal and hydraulic/pneumatic "worlds". Since we know that, for each bond, one variable is computed at each of the two ends of the bond, we realize that among the variables T and S, one must be computed on the chemical side, while the other is computed on the thermal side of the bond. Similarly for p and V. Thus, within the chemical system, we actually need only four different types of equations rather than six as previously assumed. The other two are dictated from the outside.

This fact is commonly reflected in chemical engineering by the assumption of either isothermic and isobaric or isothermic and isochoric conditions. The chemical engineer makes his life easier by holding one of the thermal variables and any one of the hydraulic/pneumatic variables at a constant value. However, these are just special cases of a more general truth.

Finally, we need one more equation. For this purpose, we shall use the so–called *equation of state*. Every system has one such equation. For example, an ideal gas reaction exhibits the following property:

$$p \cdot V = n \cdot R \cdot T \tag{33}$$

This gives us all equations needed to describe the dynamics of this system. Unfortunately, there is a problem with the last of these equations. The equation of state is actually a steady–state equation, i.e. it is only valid in a strict sense under equilibrium conditions. It should be replaced by a more general equation which is true also far from equilibrium, presumably a partial differential equation which assumes, as its steady–state condition, the various forms of the equation of state for different types of gases and fluids. Unfortunately, such an equation has not yet been found.

The current literature on chemical thermodynamics confounds wildly equations which describe structural (i.e. physical) properties of chemical reaction systems with other equations which are balance equations that are valid only under equilibrium conditions. The assumptions made are hardly ever explicitly stated. Even worse, the chemical thermodynamics literature is full of myths. Examples of such myths are given in my forthcoming book [4] in which I tried carefully to make all assumptions explicitly known.

We are now ready to set up a chemical bond graph. In this paper, we shall deal with the simplest of all cases only: the isothermic and isobaric reaction. In this case, the second power balance equation degenerates, and the chemical potentials become constants. Below, the isothermic and isobaric bond graph of the hydrogen/bromine reaction system is shown. The fourth (and least important) reaction k_4 has been eliminated to keep the bond graph planar. The bond graph is shown in Fig. 32.

The *CS* element is a newly introduced *capacitive source*. It models the mass storage. One *CS* element is used for each species. It computes the chemical potential of the component species. It is a source of chemical potential. It also stores the molar mass. In our simple situation, it can be modeled as follows:

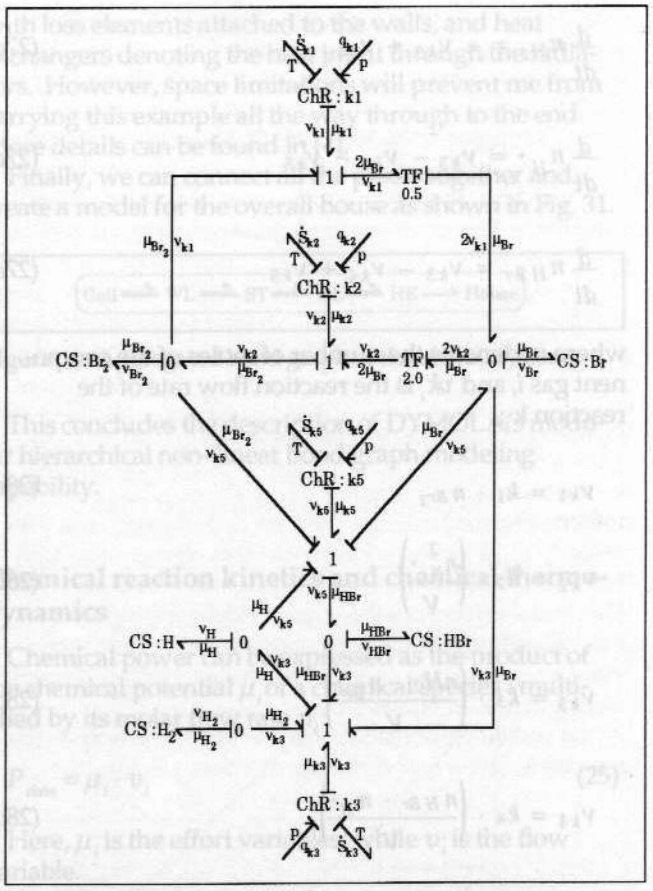


Figure 32. Bond graph of the isothermic and isobaric $H_2 - Br_2$ reaction

```
model type CS
        main cut chem(mu/nu)
        terminal n, mu0
        mu = mu0
        der(n) = nu
end
```

The capacitive source is attached to a 0–junction which models the chemical mass balance, i.e. it converts the reaction flow rates into component flow rates.

Each reaction is represented by a new bond graph element of type *chemical reactor*. A chemical reactor is a three–legged transformer. It contains the first power balance equation, the equations for the reaction flow rates, and the equation of state expressed in terms of *partial volume flows*, not decomposed into an individual component gas, but decomposed into an individual reaction. My book [4] provides the details of the calculations. The DYMOLA version of the third chemical reactor is as follows:

```
model type ChRk3
        main cut chem(muk3/–nuk3)
        cut therm(T/–Sdotk3), pneum(p/qk3)
        terminal nH2, nBr, V
```

```
    parameter R = 8.314
    local k3, nuek3
    k3 = (10 **11.43) * exp (-82400/(R * T))
    nuek3 = 0.0
    p * qk3 = T * Sdotk3 + muk3 * nuk3
    p * qk3 = nuek3 * R * T
    nuk3 = k3 * nH2 * nBr/V
end
```

The other reactor models can be defined equivalently. Due to the high degree of non–linearity and modulation, it is necessary to define one model type for each of the reactors. Each of the reactors is attached to a 1–junction which models the chemical energy balance, i.e. it converts the chemical potentials of the component species into "reaction potentials" which are expressions for the Gibbs free energy of the reaction. The reaction power:

$$P_{react} = \mu_{ki} \cdot \upsilon_{ki} \tag{34}$$

is the power that is available to be converted into either thermal or hydraulic/pneumatic power.

I discovered that there exists an interesting symmetry between the mass balance and the power balance equations:

$$\begin{pmatrix} \nu_{Br_2} \\ \nu_{Br^\bullet} \\ \nu_{H_2} \\ \nu_{H^\bullet} \\ \nu_{HBr} \end{pmatrix} = \begin{pmatrix} -1 & 1 & 0 & 0 & -1 \\ 2 & -2 & -1 & 1 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \nu_{k_1} \\ \nu_{k_2} \\ \nu_{k_3} \\ \nu_{k_4} \\ \nu_{k_5} \end{pmatrix} \tag{35a}$$

$$\begin{pmatrix} \mu_{k_1} \\ \mu_{k_2} \\ \mu_{k_3} \\ \mu_{k_4} \\ \mu_{k_5} \end{pmatrix} = \begin{pmatrix} -1 & 2 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 1 \\ 0 & 1 & 1 & -1 & -1 \\ -1 & 1 & 0 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mu_{Br_2} \\ \mu_{Br^\bullet} \\ \mu_{H_2} \\ \mu_{H^\bullet} \\ \mu_{HBr} \end{pmatrix} \tag{35b}$$

The two matrices are the transpose of each other. Moreover, they are *singular* (in many cases, they are not even square). Consequently, the component flow rates can be computed from the reaction flow rates, and the "reaction potentials" can be computed from the component potentials, but not vice versa. My book [4] provides an explanation for this symmetry.

I realize that space limitations forced me to reduce this section to a mere skeleton. My book describes the chemical reaction kinetics problem on more than 60 pages. However, I wanted to present at least the essence of these results in this paper.

Acknowledgments

This paper is an extended version of my keynote address previously presented during the 1990 European Simulation MultiConference in Nürnberg, FRG.

References

[1] Breedveld, P.C., (1982). "Thermodynamic Bond Graphs and the Problem of Thermal Inertance", *J. Franklin Institute*, **314** (1), pp. 15–40.

[2] Breedveld, P.C., (1984). *Physical Systems Theory in Terms of Bond Graphs*, Ph.D. Dissertation, Technical University Twente, Enschede, The Netherlands.

[3] Broenink, J.F., (1990). *Computer–Aided Physical–Systems Modeling and Simulation: A Bond–Graph Approach*, Ph.D. Dissertation, Universiteit Twente, Enschede, The Netherlands.

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

[5] Elmqvist, H., (1975). *SIMNON–An Interactive Simulation Program for Non–linear Systems–User's Manual*, Report CODEN: LUTFD2/(TFRT–7502), Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

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

[7] Granda, J.J., (1982). *Computer Aided Modeling Program (CAMP): A Bond Graph Preprocessor for Computer Aided Design and Simulation of Physical Systems Using Digital Simulation Languages*, Ph.D. Dissertation, Dept. of Mechanical Engineering, University of California, Davis.

[8] Granda, J.J., (1985). "Computer Generation of Physical System Differential Equations Using Bond Graphs", *J. Franklin Institute*, **319**(1/2), pp. 243–255.

[9] Granda, J.J., and F. Pourrahimi, (1985). "Computer Graphic Techniques for the Generation and Analysis of Physical System Models", in: *Artificial Intelligence, Graphics, and Simulation*, Proceedings of the Western Simulation MultiConference, (G. Birtwistle, ed.), SCS Publishing, pp. 70–75.

[10] Karnopp, D.C., and R.C. Rosenberg, (1974). *System Dynamics; A Unified Approach*, John Wiley, New York.

[11] Korn, G.A., (1989). *Interactive Dynamic–System Simulation*, McGraw–Hill, New York.

[12] Mitchell, E.E.L., and J.S. Gauthier, (1986). *ACSL: Advanced Continuous Simulation Language–User Guide/ Reference Manual*, Mitchell & Gauthier Assoc., 73 Junction Square, Concord, MA 01742.

[13] Paynter, H.M., (1961). *Analysis and Design of Engineering Systems*, M.I.T. Press, Cambridge, MA.

[14] Rosenberg, R.C., (1974). *A User's Guide to ENPORT–4*, John Wiley, New York.

[15] RosenCode Associates, Inc. (1989), *The ENPORT Reference Manual*, RosenCode Associates, Inc., 200 N. Capitol Bldg., Lansing, MI 48933.

[16] Thoma, J.U., (1975). "Entropy and Mass Flow for Energy Conversion", *J. Franklin Institute*, **299**(2), pp. 89–96.

[17] van Dixhoorn, J.J., (1982). "Bond Graphs and the Challenge of a Unified Modelling Theory of Physical Systems", in: *Progress in Modelling and Simulation*, (F.E. Cellier, ed.), Academic Press, London, pp. 207–245.

FRANÇOIS E. CELLIER received his B.S. degree in Electrical Engineering from the Swiss Federal Institute of Technology (ETH) Zürich in 1972, his M.S. degree in Automatic Control in 1973, and his Ph.D. degree in Technical Sciences in 1979, all from the same university. Following his Ph.D., Dr. Cellier worked as a Lecturer at ETH Zürich. He joined the University of Arizona in 1984 as an Associate Professor. Dr. Cellier's main scientific interests concern modeling and simulation methodology, and the design of advanced software systems for simulation, computer–aided modeling, and computer–aided design. He has designed and implemented the GASP–V simulation package, and he was the designer of the COSY simulation language a modified version of which under the name of SYSMOD has meanwhile become a standard by the British Ministry of Defence. Dr.Cellier has authored or co–authored more than forty technical publications, and he has edited two books. He served as a chairman of the National Organizing Committee (NOC) of the Simulation'75 conference, and as a chairman of the International Program Committee (IPC) of the Simulation'77 and Simulation'80 conferences, and he has also participated in several other NOC's and IPC's. He is associate editor of several simulation related journals, and he served as vice–chairman of two committees on standardization of simulation and modeling software.

CALL FOR PAPERS

1993 INTERNATIONAL CONFERENCE ON BOND GRAPH MODELING AND SIMULATION (ICBGM '93)

JANUARY 17–20 1993, HYATT REGENCY (LA JOLLA) SAN DIEGO, CALIFORNIA
PART OF THE 1993 SCS WESTERN MULTICONFERENCE ON COMPUTER SIMULATION

Sponsored by SCS The Society for Computer Simulation International

The 1993 International Conference on Bond Graph Modeling and Simulation brings together research papers, presentations, panel sessions, tutorials, workshops, seminars, industrial applications, and software demonstrations that use Bond Graph modeling methods. There will be also featured plenary speakers and some special invited presentations.

This conference will focus on Bond Graph modeling techniques for dynamic systems. Theoretical principles for electrical, mechanical, hydraulic, pneumatic, and control system applications will be presented. Leading industrial users of the method in automotive, aircraft, fluid power, kinematics, multibody systems, and social and biological systems have been invited. Research papers are welcome in the following categories of presentation

SESSIONS:

. Tutorials	. Mechanical Systems and Robotics
. Panel Discussions	. Electrical and Power Systems
. Software and Tools	. Control Systems
. Bond Graph Theory	. Thermal and Chemical Systems
. Advanced Bond Graph Methodology	. Biomechanics and Prosthetics
. Bond Graphs and Block Diagrams	. Ecological Systems
. Computer Graphics and B.G. Modeling	. Social and Economic Systems
. Qualitative Modeling Using Bond Graphs	. Industrial Applications (Large, Nonlinear Models)

DEADLINES:

. May 31, 1992	Short Abstracts due (two copies).
. June 30, 1992	Notification of abstract acceptance to authors.
. July 31, 1992	Full papers due (five copies).
. August 31, 1992	Notification of paper acceptance to authors.
. September 30, 1992	Camera Ready Copies due.

Abstracts and full papers can be submitted alternatively to the General Chairman or the Program Chairman. They must contain original contributions that have not been previously published in the literature. In addition to regular papers about 12 pages (double-spaced), the program committee may accept additional papers, either as short papers (about 6 pages double-spaced), or proposals for tuturials. Please add a cover letter stating your name, affiliation, mail address, telephone number, FAX number, and EMail address. The short abstracts will be reviewed by the two chairmen for general suitability to the conference. Each full paper will be reviewed by at least three members of the International Program Committee.

General Chairman
Jose J. Granda
Department of Mechanical Engineering
California State University, Sacramento
Sacramento, Calif. 95819
Phone: (916) 278-5711
FAX: (916) 278-5949
EMail: GrandaJJ@ECS.CSUS.Edu

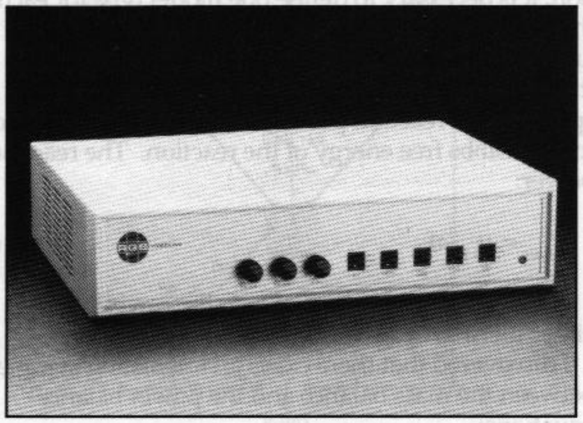
Program Chairman
Francois E. Cellier
Dept. of Electr. and Computer Engr.
University of Arizona
Tucson, Ariz. 85721
Phone: (602) 621-6192
FAX: (602) 621-8076
EMail: Cellier@ECE.Arizona.Edu

International Program Committee

Peter Breedveld, Tech. University of Twente, Netherlands	Francis Lorenz, Lorenz Consulting, Liege, Belgium
Marisol Delgado, Universidad Simon Bolivar, Venezuela	Donald Margolis . University of California, Davis , U.S.A.
Hallvard Engja, Norwegian Institute of Technology, Norway	Fernando Mejia, Universidad Nacional de Colombia, Colombia
Peter Gawthrop, University of Glasgow, Scotland , U.K.	Henry Paynter, formerly at M.I.T., U.S.A.
Dean Karnopp , University of California, Davis, U.S.A.	Jean Thoma, Thoma Consulting, Zug, Switzerland
Lennart Ljung, Linkoping University, Sweden	Tong Zhou, California State University, Sacramento, U.S.A.

New Model

Convert Computer Graphics to Television



RGB / Videolink®
Model 1450AX
with Auto-sync

The first popularly priced full range scan converter for video taping, video projection and video teleconferencing

- **Adjustment free auto-locking to all workstations, PCs and Mac IIs**
- **Horizontal scan range 21 to 80 kHz**
- **Full broadcast quality encoder and sync generator**
- **Flicker free output**
- **Genlock**
- **Linear keyer for overlaying live video with computer graphics**
- **Zoom EGA to fill video screen**
- **Full 24 bit color processing**
- **Wide range of outputs; Composite (NTSC or PAL), RGB, S-Video, Betacam / MII**
- **Manufactured in the USA**

Other models available

RGB
SPECTRUM
2550 Ninth Street Berkeley, CA 94710
TEL: (415) 848-0180 FAX: (415) 848-0971