
Unified aggregate component modelling of dynamic mechatronic systems using Bond Graphs

Stephen. J. Dickinson

Engineering Department, Lancaster University, Lancaster, UK

E-mail: s.dickinson@lancs.ac.uk

Abstract Traditional dynamic modelling of multidisciplinary systems involves separate sets of analysis skills for each domain. Bond Graphs embrace commonality, providing a concise unified means of system representation and a systematic analysis method. This paper introduces Bond Graphs, benefits of acausal component models, and software for automated equation generation and simulation.

Keywords Bond Graphs; component modelling; dynamic systems; mechatronics

From the smallest nano-robot to the largest hydro-electric power plant, assessment of dynamic performance is vital in order to optimise and verify designs, and to reduce the probability of costly mistakes. A most challenging task in this area is the creation of dynamic system models: mathematical representations of systems which are simple enough to understand, compute and analyse, and yet at the same time are detailed enough to capture all essential dynamic aspects within particular operational envelopes. Bond Graph notation and methodology address three key areas which are fundamental to the understanding, analysis, simulation and hence teaching of physical dynamic mechatronic systems. First, that of a highly concise unified graphical notation that seamlessly crosses boundaries between engineering disciplines. Secondly, the ability to represent complete non-causal sub-models which greatly simplifies the construction of aggregate models; and thirdly, simple analytical methods which allow relationships and system equations to be systematically deduced at a glance. Bond Graphs also lend themselves well to automated transformation to and from other notations such as circuit schematics or block diagrams, and also the fully automated generation of system equations.

Ideally, the creation of a dynamic system model is best treated as a process consisting of two distinct phases. The first, and most challenging, phase involves making important decisions on the high level model structure: which physical elements to include, which to leave out, and which may be combined. A single-discipline example of this would be the creation of a circuit schematic where decisions such as whether or not to include parasitic elements, such as capacitance between PCB tracks, would be made depending on their significance within the envelope of operation, mainly frequency range. The development of a dynamic model is generally an iterative process where basic assumptions may be changed many times; the use of a rapid prototype-type environment where high level changes are instantly reflected in system equations can therefore yield great benefits.

The problem of incompatibility between sub-models due to conflicting assign-

ment of causality is addressed well by Bond Graphs,¹ as will be seen later in this paper. When dealing with mixed discipline mechatronic systems the ability to connect sub-models from different disciplines is obviously essential, however the possibility of merging schematics from such multiple disciplines can be problematic, especially when attempting to automate the generation of composite system equations. An approach commonly adopted is to generate independent sets of equations for each sub-system, and then merge them to produce the final aggregate system model. This however can only be done if the causality of each sub-model is suitably assigned, and it therefore becomes necessary to manually fix causalities for each sub-system interconnect prior to the generation of sub-system equations. The problem is compounded where component libraries are employed, as alternative sets of equations must be maintained for each sub-model.

This paper introduces Bond Graph notation and methodology mainly through examples and by comparison with other well-known techniques. Important issues regarding the assignment of causality will be discussed, particularly within the context of developing component model libraries. Finally, a teaching lab session example will be described where students use Bond Graph software to simulate systems directly from the Bond Graph representation.

Bond Graph concepts and notation

Before proceeding to Bond Graphs, it is useful first to consider how a set of first-order differential equations may be obtained for the system shown in Fig. 1 using 'traditional' techniques.

There are several ways of going about this task;² ways which may on the surface seem dissimilar and vary widely in rigour. However, whether done formally or informally, each method must at some point assume a particular set of causal assignments which will dictate an equation structure such as that shown graphically by the block diagram³ of Fig. 2.

Having generated the block diagram, sets of first-order state space equations may then be obtained by reading directly from the block diagram. This system has two

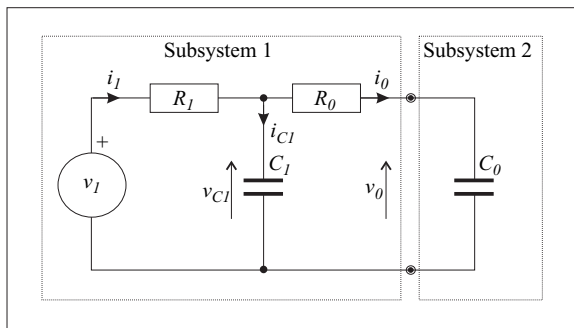


Fig. 1 A schematic of an electrical RC network.

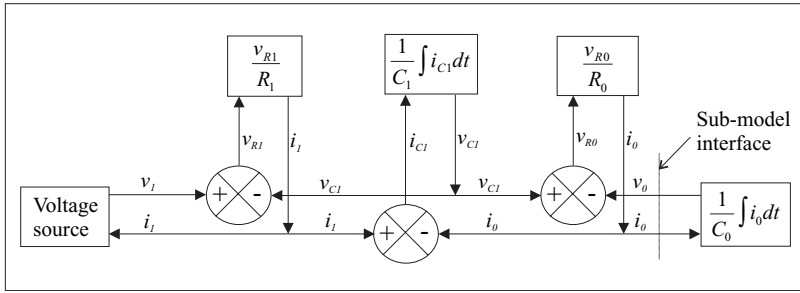


Fig. 2 Block diagram representation of RC system equations.

state variables, therefore it can be represented by a set of two such equations which are:

$$\frac{dq_1}{dt} = V_1 \frac{1}{R_1} - q_1 \left(\frac{1}{R_1 C_1} + \frac{1}{R_0 C_1} \right) + q_0 \frac{1}{R_0 C_0}, \text{ and} \tag{1}$$

$$\frac{dq_0}{dt} = q_1 \frac{1}{R_2 C_1} - q_0 \frac{1}{R_2 C_0} \tag{2}$$

where q_1 and q_0 are the state variables associated with C_1 and C_0 respectively.

The issue of causality can be developed from Fig. 2. If we first consider the voltage source, by definition an ideal voltage source will determine (or cause) the voltage on whatever it is connected to, which in this case is the rest of the RC circuit, as shown by the v_1 signal line arrow pointing away from the voltage source in Fig. 2. Conversely, it follows that any current flowing in the voltage source, in this case i_1 , is determined by the rest of the circuit. This is shown by the i_1 arrow pointing into the voltage source. In the case of a capacitor, the decision as to causal direction is not quite so cut and dry, as a capacitor can equally well be treated as a differentiator of voltage, or an integrator of current. In Bond Graph ‘language’ these two causal options are referred to as ‘differential causality’ and ‘integral causality’ respectively. It is however preferable to avoid any differentiating elements within the final model, as the resulting equations will generally be problematic. Integral causality has therefore been assumed for both capacitors in the example, and this is shown by the current lines pointing into the capacitor node and the voltage lines pointing out, and of course, the node indicating the appropriate integral function. There is no preference to resistor causality except that the assignment must be compatible with the adjoining system. In this example it has been assumed that the system is causing the voltage on both resistors, and conversely the resistors are therefore causing the current. Another important point to note regarding the block diagram is that voltage lines and current lines have been arranged in pairs. This is by no means accidental, as the instantaneous product of each pair yields the instantaneous power. Each pair therefore represents a single path for power to flow, a power flow which is positive when the product of the two is positive. An important aspect that is not however

apparent from this diagram is the direction of positive power, an issue which, as will be seen, is addressed by Bond Graph notation.

The Bond Graph shown in Fig. 3 represents the RC circuit shown in Fig. 1, and despite its sparse appearance it contains all of the information in the block diagram of Fig. 2. The general layout of this block diagram and the corresponding Bond Graph have deliberately been kept the same so that their similarities may be easily seen. Essentially, a Bond Graph is a true graph in the mathematical sense and concisely captures all of the essential features of a dynamic system as a set of nodes linked by bonds representing paths for power flow. There are nine basic types of Bond Graph node as summarised in Table 1.

In order to be generic across engineering domains, standard Bond Graph notation defines two co-variables: ‘Effort’ to denote quantities such as voltage, pressure and force; and ‘Flow’ to denote quantities such as current, flow and velocity, their product being power. The idealised voltage source of Fig. 1 is therefore represented in Fig. 3 by an idealised ‘Source of Effort’ node, denoted by the ‘Se’ node type and here having an Effort quantity of v_1 . Bond Graph nodes are connected by single lines (the ‘bonds’) which each represent a path for power flow. Each bond represents both an Effort and Flow quantity in a way that is semantically identical to that shown explicitly in the block diagram of Fig. 2. A bond’s Effort and Flow are always opposite to each other, and their causal direction may be indicated by a single perpendicular stroke (known as a causal stroke) drawn at one end of the bond. The causal stroke may be thought of as showing the arrowhead of the Effort variable, and as the Flow variable is always in the opposite direction, then its direction of this may also be inferred.

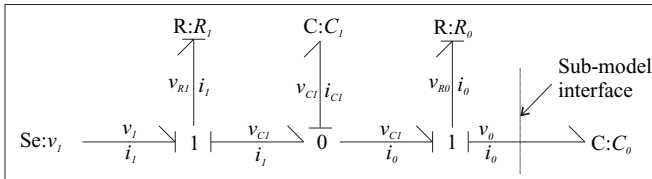


Fig. 3 Bond Graph representation of the RC circuit.

TABLE 1 Summary of Bond Graph node types

Bond Graph node type	Electrical system	Mechanical system
‘Se’ (Source of effort)	Voltage source	Force source
‘Sf’ (Source of flow)	Current source	Velocity source
‘R’ (Energy dissipater)	Resistor	Damper
C (Energy storage)	Capacitor	Spring
I (Energy storage)	Inductor	Mass
‘0’ (Common effort)	Equal voltage relationship	Equal force relationship
‘1’ (Common flow)	Equal current relationship	Equal velocity relationship
‘TF’ (Transformer)	e.g. transformer	e.g. lever, gearbox
‘GY’ (Gyrator)		e.g. d.c. motor or loudspeaker

In the Bond Graph example shown in Fig. 3, the causal stroke on the Se: v_1 bond indicates that the Se node is determining the Effort on the '1' type node, and hence the '1' type node is causing the effort on the Se node. The placement of the causal stroke also implies the form of node equation to use, for example consider the case of a resistor. If the connected circuit is determining (i.e. causing) the voltage v across a resistor then according to Ohm's Law the current i flowing through the resistor is $i = v/r$. Conversely, if the connected circuit is determining the current i through a resistor, then the voltage across the resistor is $v = ir$. It can be seen therefore that the choice of equation to use for a resistor is entirely dependent on whether the external circuit is causing the voltage on, or causing the current through the resistor. In other words, in Bond Graph terminology, the choice is dependent on causality, i.e. an 'R' type node may be $\text{Flow} = \text{Effort}/R$ or $\text{Effort} = \text{Flow} \times R$ for a near or far causal stroke respectively.

Bond Graph notation defines two types of junction node, namely the '0' type and the '1' type of Table 1. The purpose of each junction type can best be appreciated by comparing the Bond Graph of Fig. 3 with the block diagram of Fig. 2. In essence, a '0' type node is used to indicate that all connected bonds have an equal Effort value as defined by exactly one bond, and that the bond which defines the Effort will in turn have its Flow determined as the sum of Flows on all other connected bonds (Kirchhoff's current law). Conversely, a '1' type node is used to indicate that all connected bonds have an equal Flow value as defined by exactly one bond, and that the bond which defines the Flow will in turn have its Effort determined as the sum of Efforts on all other connected bonds (Kirchhoff's voltage law).

The power directions around a junction node have significance in that they determine the sign of the inputs to the nodal summing junction. If the power direction of a particular input to the summing junction is the same as the output from the junction, then the input is non-inverted, otherwise it is inverted. In order to illustrate this, consider the left-hand '1' type node shown in Fig. 3. Being a '1' type node it follows that all flow variables of connected bonds have the same value, and as indicated by the causal strokes, R_1 is determining that value. The node's causality also means that the Effort v_{R1} is the output from the summing junction, and by examining the power directions that v_1 is added and v_{c1} is subtracted, i.e. $v_{R1} = v_1 - v_{c1}$. In a little more detail, if we wish to find the equation for the Effort on R_1 (i.e. v_{R1}), then we must traverse the connected bond away from causal stroke. This leads us to the '1' type node, which we know from the previous discussion serves to sum Efforts. It is important to note at this point that this first bond was traversed in the opposite direction to the power direction half arrow. This '1' type node has two Effort input bonds v_1 and v_{c1} , and as for all '1' type nodes, it has a single Effort output bond. The value of v_{R1} is therefore the sum of the positive or negative values of v_1 and v_{c1} . On inspection it can be seen that the sign of the v_1 component is positive because the power direction of the v_1 bond is in the same direction to that of the v_{R1} bond. Similarly the sign of the v_{c1} component is negative because the power direction of the v_{c1} bond is opposite to that of the v_{R1} bond. The equation $v_{R1} = v_1 - v_{c1}$ is thereby obtained.

Sub-models and model libraries

An important benefit with Bond Graph notation is the ability to construct acausal Bond Graphs, i.e. Bond Graphs with no assigned causality. When building systems from library components the assignment of causality is best performed after the component sub-models have been pieced together. Pre-assignment of causality to sub-models can greatly restrict their compatibility for connection with other sub-models.

Many methods used for the representation of sub-models, such as Block Diagrams or even state equations, have no such intermediate acausal representation, as the diagram hangs together on causality-loaded symbols such as summing junctions. The use of Bond Graphs allows the most difficult task, that of creating the structural sub-model, to be carried out and stored in acausal form. Having designed a system using acausal Bond Graph sub-models, it is then a trivial task to assign causality to the whole system using routine Bond Graph methodology. This point can be seen by comparing the circuit schematic and its associated block diagram and Bond Graph shown in Figs 1, 2 and 3 with those of Figs 4, 5 and 6 respectively.

The only thing that has changed between the circuit schematic shown in Fig. 1 and that of Fig. 4 is that sub-system 2 has been replaced by sub-system 3. However,

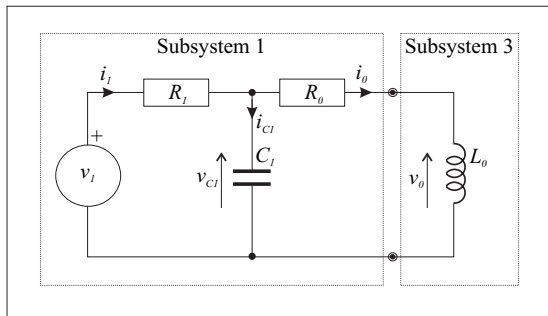


Fig. 4 A schematic of an electrical RLC network.

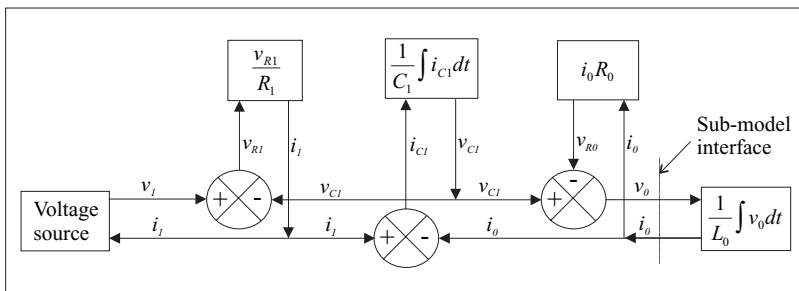


Fig. 5 Block diagram representation of RLC system equations.

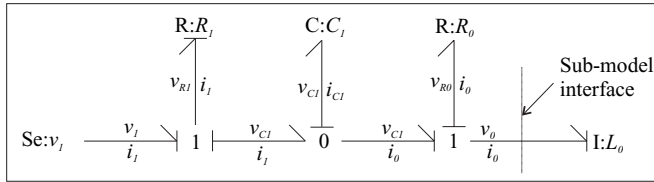


Fig. 6 Bond Graph representation of the RLC circuit.

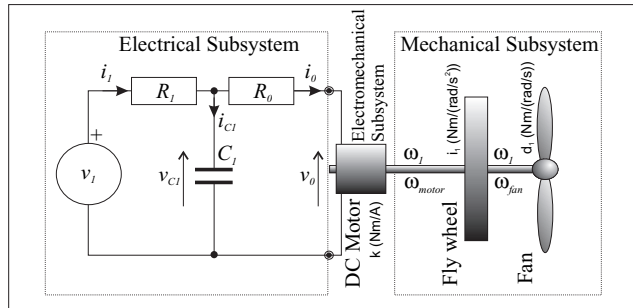


Fig. 7 A mechatronic system schematic.

in order to have integral causality in sub-system 3, it is necessary to assign an opposite causality to that of sub-system 2 making the causality assigned to sub-system 1 in Fig. 2 incompatible with that of sub-system 3. Figure 5 shows an alternative causality for sub-system 1, which is compatible with sub-system 3. The Bond Graph in Fig. 6 illustrates how easy it is to modify the causality of individual Bond Graph bonds without major reworking.

A mechatronic example

Having now introduced the basic concepts of Bond Graphs, their application to a system having both electrical and mechanical elements as shown in Fig. 7 is now considered. The primary electrical sub-system is sub-system 1 from Figs. 1 and 4, and is connected to a sub-system which happens to be a d.c. motor. The d.c. motor acts as an interface between the electrical domain and the mechanical domain, allowing power to be transferred not only from the electrical to the mechanical domain but also vice versa. In this particular case, the motor is assumed to be ideal in that the instantaneous input and output power are always equal, the mechanical torque produced is proportional to the electrical current, and the electrical voltage is proportional to the mechanical speed, where k is the constant of proportionality. The motor's shaft is attached to the third mechanical sub-system consisting of a fly wheel, representing a mechanical inertia, and a fan representing a damper, and it is assumed that there is no spring in the shaft.

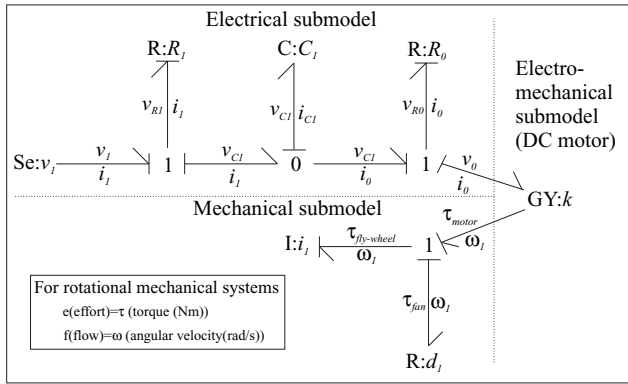


Fig. 8 Bond Graph representation of the mechatronic system.

The Bond Graph is constructed, first as a set of independent acausal sub-models, which are then joined together to make the complete acausal aggregate Bond Graph and then finally the causality of the whole system is defined using a few simple rules. The electrical sub-model is the same as before and therefore needs no more explanation. The d.c. motor sub-model is modelled as a Gyrator (GY) type Bond Graph node. A GY type node has two ports (i.e. it has two connected bonds) and its purpose is to swap over the Effort and Flow variables between one port and the other while applying a scaling constant to both. A GY type node does not dissipate power, therefore the instantaneous power on both sides is always equal. For the Mechanical sub-model it can be seen that the angular velocity (flow) is the same on the input shaft, the fan and the fly-wheel as represented by the ‘1’ type node linking all three together. Figure 8 shows all three sub-models joined together and the final causality assigned.

Automated equation formulation and simulation

A range of Bond Graph support software is available that allows Bond Graph models to be entered and processed for a variety of purposes such simulation and dynamic analysis. One such program is called Quest and was developed by the author of this paper at Lancaster University, and this has been successfully used in teaching labs. Essentially, the Quest program accepts Bond Graph models and sub-models in a textual form and performs a range of operations including automatic assignment of causality, derivation of system equations and time domain simulation.

Teaching example – Three-bogie train simulation

In this example the students are asked to model and simulate a three-bogie train consisting of three masses linked by springs and dampers as shown in Fig. 9. Just to make things a little more interesting, the input to the system is in the form of current into a d.c. motor. Values for all components are specified and it is required to perform

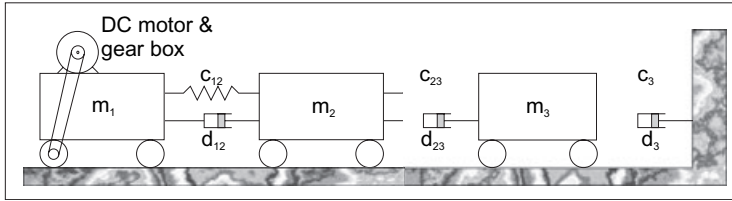


Fig. 9 Train system to be modelled.

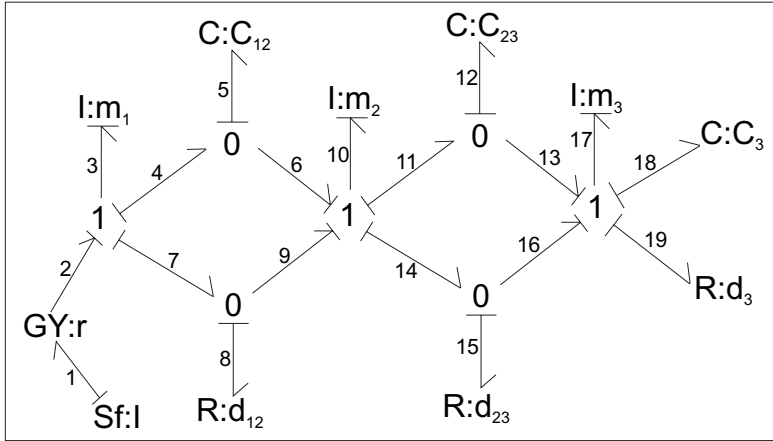


Fig. 10 Bond Graph of train system.

a simulation for a step input at time = zero. The main steps involved in the task are outlined below, and potential problem areas such as choice of simulator configuration parameters are removed by recommending values which are known to work. As the lab session progresses, solutions to each stage are handed out so that all students can at least appreciate the overall process and can successfully generate a working simulation and plots.

The first step is to produce a Bond Graph representation of the system shown in Fig. 9. Students are hopefully by this stage familiar with the necessary techniques through lectures and example sheets. There are several possible Bond Graph representations of this system and one such solution is shown in Fig. 10 which is briefly explained as follows:

This system has one source of energy input which is the current ‘I’ into the d.c. motor and is represented by a ‘Sf’ (flow source) type Bond Graph node. The d.c. motor is a particularly interesting type of device because the effort and flow variables swap between the electrical domain and the mechanical domain. In other words, an input current determines the torque output, and the angular velocity on the output determines the terminal voltage. As we are assuming in this case an ideal motor, gearing and transformation from rotational to linear movement, it is possible

to lump these three parts into one equivalent part, the lumped part being a linear d.c. motor. This lumped motor's input current will determine the linear mechanical force and the linear mechanical velocity will determine the motor's terminal voltage. This relationship is represented by a 'GY:r' (Gyrator) type Bond Graph node where the r parameter allows a ratio to be specified, so allowing conversion of units and gear ratios to be dealt with. Like an ideal transformer or lever, there is no power loss, therefore the product of Effort and Flow variables on each side is always equal. The output from this 'GY' node is a force, and this is applied to a '1' (common flow) type node and, as can be seen from the Bond Graph of Fig. 10, the output from the Gyrator (or linear motor) has the same velocity as for mass m_1 , one end of the spring C_{12} and one end of the damper d_{12} . This '1' type node also indicates that the forces on all connected nodes always sum to zero. Moving now to the spring C_{12} , we see that it is connected to a '0' type node, indicating that all connected bonds have the same effort and conversely that the flows sum to zero. On reflection this seems reasonable enough, as we would expect the force on either end of the spring to be equal, and also for the rate of change in spring length to be the difference between the two velocities of either end. At first glance the arrangement of C_3 looks entirely different, however as one end of this spring is connected to the fixed frame of reference and will therefore always have a velocity of zero, associated bonds can be removed with no effect on the model. In effect we are saying that the rate of change of the length of C_3 is entirely determined by the velocity of m_3 .

Having constructed the Bond Graph (with or without causality defined), the next stage is to enter the structure using Quest's Model Description Language (MDL), as shown in Fig. 11. This may be performed within the Quest environment by creating a new MDL document and entering the text.

The Quest MDL code is fairly straightforward and is best understood by comparing it with the Bond Graph shown in Fig. 10. A model or sub-model starts with the keyword 'bond_graph' and ends with the keyword 'end'. In between the start and end, there are various simple lists which define variables, nodes and structure, power directions and relationships in the form of equations. Various types of variables exist including 'Input', 'Output' and 'Variable'. Of particular importance, output variables facilitate the recording of values over time, whereas normal variables discard historic data. State variables are generated automatically whenever a variable is assigned to a storage-type Bond Graph node, i.e. a C or an I type node, and any of these may be used in equations. State variables adopt the name of the components parameter variable prefixed with 'sv_'. In the example shown in Fig. 11, the variable s_{12} is associated with a C-type node, therefore a state variable called sv_s_{12} is automatically created. The 'node' list specifies all of the Bond Graph nodes including node types, associated variables and connected bond numbers.

The next step is to validate and simulate the Bond Graph model. This is done by creating a new 'Experiment' document which is a dialogue-type form as shown in Fig. 12. The MDL 'File Name' is then selected using a file open dialogue and the appropriate 'Model Name' from within the file is selected using a pull down menu. The structure and syntax of the model can then be checked by pressing the 'Validate' button, and any detected errors open up the MDL editor and highlight the

```

bond_graph(train)

  Inputs:
    current_ip=10; {Amps}

  Outputs:
    p1, p2, p3;    {Mass displacement in metres}

  Variables:
    motor_k=1000,
    m1=1000,      m2=1000,      m3=1000,    {Kg}
    s12=0.00001, s23=0.00001,  s3=0.00001, {m/n}
    d12=100000,  d23=1000,    d3=10000;  {ns/m}

  nodes:
    Sf:current_ip->[1],      GY:motor_k->[1,2],
    n1->[2,3,4,7],          I:m1->[3],
    n0->[4,5,6],            n0->[7,8,9],
    n1->[6,9,10,11,14],     C:s12->[5],
    R:d12->[8],              n1->[13,16,17,18,19],
    I:m2->[10],              n0->[11,12,13],
    n0->[14,15,16],         C:s23->[12],
    R:d23->[15],            I:m3->[17],
    C:s3->[18],              R:d3->[19];

  power_directions:
    bond(1) [Sf->GY],      bond(2) [GY->n1],
    bond(3) [n1->I],        bond(4) [n1->n0],
    bond(5) [n0->C],        bond(6) [n0->n1],
    bond(7) [n1->n0],       bond(8) [n0->R],
    bond(9) [n0->n1],       bond(10) [n1->I],
    bond(11) [n1->n0],      bond(12) [n0->C],
    bond(13) [n0->n1],      bond(14) [n1->n0],
    bond(15) [n1->R],       bond(16) [n0->n1],
    bond(17) [n1->I],       bond(18) [n1->C],
    bond(19) [n1->R];

  equations:
    p3=sv_s3,
    p2=sv_s23+sv_s3,
    p1=sv_s12+sv_s23+sv_s3;

end.

```

Fig. 11 Train Bond Graph in Quest MDL form.

errors. Simulation is then a matter of selecting start and finish times, time units and the numeric integration method, together with associated settings such as the minimum step size, and finally pressing the default simulation button. Other buttons control optimisation capability and are outside the scope of this paper.

Having performed the simulation, results are then viewed by creating a new 'Plot' document as shown in Fig. 13 showing the displacement of the three masses over time. A plot document connects up to model output data channels, which are updated whenever the simulator is rerun.

Conclusion

Bond Graphs are a highly concise and methodological system for modelling and analysing dynamic systems. They cross the boundaries of engineering disciplines

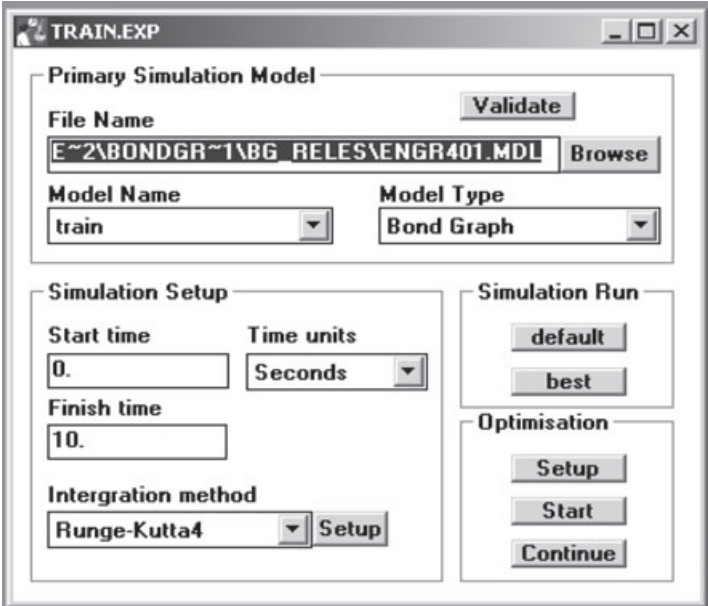


Fig. 12 Quest Experiment document (screen shot).

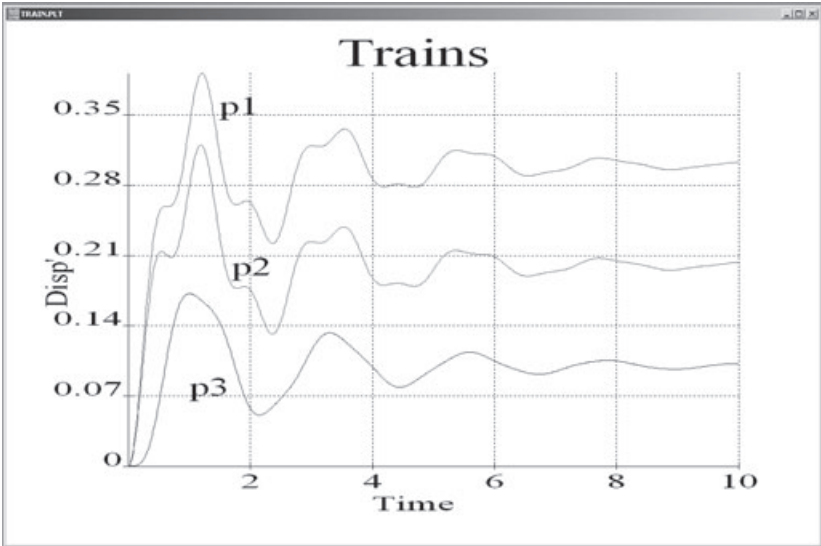


Fig. 13 Train simulation results (screen shot).

seamlessly and allow the engineer to see important dynamics uncluttered by irrelevant or repeated information. The highly polished methodology promotes excellent practice and clear thinking, highlighting obvious important analytical points which often go unrealised, making the technique ideal for teaching and professional use at all engineering levels. There is a small learning curve associated with Bond Graphs as the notation will inevitably seem strange and unnatural at first. However following an initial investment of time and effort the payback is great (no pain, no gain). It has been found at Lancaster University that students are able to use them for solving problems after only a few days' teaching and practice. Bond Graph notation and methodology lend themselves to implementation in software and a range of software tools is available including tools for graphical input, analysis, automated generation of equations and simulation. One such tool has been developed by the author of this paper at Lancaster University which facilitates the automatic generation of state space system equations, simulation and graph plotting.

References

- 1 D. C. Karnopp, D. L. Margolis and R. C. Rosenberg, *System Dynamics, A Unified Approach*, 2nd edn (Wiley, Chichester, 1990).
- 2 R. E. Ziemer, W. H. Tranter and D. R. Fannin, *Signals and Systems: Continuous and Discrete*, 2nd edn (Macmillan, Basingstoke, 1989).
- 3 G. F. Franklin, J. D. Powell and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 2nd edn (Addison Wesley, New York, 1991).