HyBrSim—a modelling and simulation environment for hybrid bond graphs

P J Mosterman
Institute of Robotics and Mechatronics, DLR Oberpfaffenhofen, D-82230 Wessling, Germany

Abstract: Bond graphs are a powerful formalism to model continuous dynamics of physical systems. *Hybrid bond graphs* introduce an ideal switching element, the *controlled junction*, to approximate continuous behaviour that is too complex for numerical analysis (e.g. because of non-linearities or steep gradients). HyBrSim is a tool for hybrid bond graph modelling and simulation implemented in Java and is documented in this paper. It performs event detection and location based on a bisectional search, handles run-time causality changes, including derivative causality, performs physically consistent (re-)initialization and supports two types of event iteration because of dynamic coupling. It exports hybrid bond graph models in Java and C/C++ code that includes discontinuities as switched equations (i.e. pre-enumeration is not required).

Keywords: hybrid systems, physical system modelling, bond graphs, hybrid bond graphs, hydraulic actuators

NOTATION

C	storage of generalized displacement
e	effort
f	flow
GY	gyrator
I	storage of generalized momentum
p	generalized momentum
q	generalized displacement
R	dissipation
Se	effort source
Sf	flow source
TF	transformer
0	generalized Kirchhoff current law
1	generalized Kirchhoff voltage law

1 INTRODUCTION

Physical systems can be modelled by sets of ordinary differential equations (ODEs) possibly supplemented by algebraic constraints [differential and algebraic equations (DAEs)]. Often these are composed from local constituent equations of primitive elements and constraints imposed by a network structure that connects them [1, 2].

The MS was received on 7 December 2000 and was accepted after revision for publication on 15 May 2001.

I08800 © IMechE 2002

1.1 Stiff systems

Continuous physical system models may contain non-linearities and steep gradients that complicate numerical simulation. Integration methods such as the backward difference formula (BDF) [3, 4] address these problems by reducing the simulation step size, at the cost of increased computational complexity.

To illustrate, consider the hydraulic cylinder in Fig. 1. To move the piston with inertia m_p, the intake valve with fluid inertia I_{in} and flow resistance R_{in} is adjusted to control oil flow into the cylinder. The elasticity C_{oil} and viscosity R_{oil} of the oil in the chamber generate a pressure that opens a relief valve with inertia and resistance, I_{rel} and R_{rel} respectively, when the pressure exceeds a predefined safety threshold.

Initially, the relief valve may be closed and when the

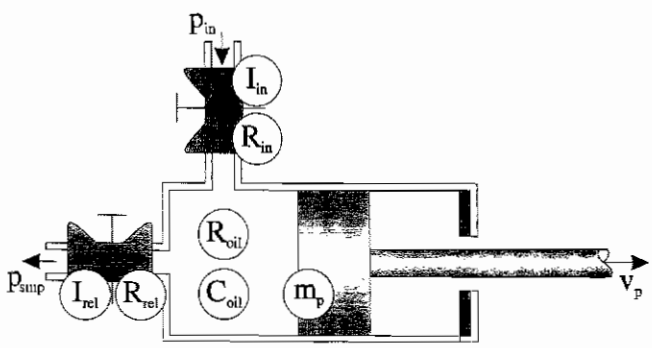


Fig. 1 Hydraulic cylinder with relief valve

intake valve closes completely, there is no flow of oil into the chamber, causing the piston velocity to become zero quickly. In a continuous model, a small leakage flow of the valve interacts with the fast dynamics induced by C_{oil} and R_{oil}, which leads to high-frequency low-amplitude oscillations requiring a small simulation step size for a considerable time interval.

For the particular problem, efficient variable step-size numerical integration may not be possible or such solvers may not be available or suitable for the task at hand (e.g. real-time simulation).

1.2 Discontinuities

Stiff systems because of steep gradients can be simplified by abstracting the fast continuous transients into discontinuous changes and the models assume a mixed continuous–discrete hybrid nature [5]. Hybrid systems typically operate in piecewise continuous modes, modelled by ODEs and DAEs. Mode changes are most conveniently modelled by activating and deactivating constituent equations of model components which may change computational causality. For example, when the intake valve in Fig. 1 closes, its constituent equation changes from enforcing a zero pressure drop across it to enforcing zero flow. These causal changes may result in dynamic (i.e. run-time) changes in the state vector.

Two situations can be classified:

1. State variables become dependent on exogeneous variables.
2. State variables become algebraically related.

In Fig. 1, when the intake valve closes and the relief valve is closed as well, if the oil elasticity and viscosity are abstracted away, the piston velocity is forced to zero, and, therefore, its momentum is no longer a state variable (first issue). In this mode, the required pressure build-up in the cylinder may be such that the relief valve opens. Now, the state variables that correspond to I_{rel} and m_p, i.e. the fluid momentum p_{rel} and the momentum p_p, become dependent (algebraically related) and the initial piston momentum has to be distributed so that their values are mutually consistent (second issue).

The closing of the intake valve and the opening of the relief valve follow each other with no continuous differential-equation behaviour in between. In general, (de)activating blocks of equations may result in a sequence of consecutive mode changes that has to converge before continuous behaviour resumes [5].

1.3 HyBrSim

Hybrid bond graph simulator HyBrSim is a modelling and simulation environment to handle hybrid behaviours, implemented in Java. Instead of generating a global system of equations, HyBrSim attempts to propagate known variable values (input and state) through the model topology at each evaluation, i.e. it is interpretive. The advantage of the interpretive approach lies in the flexible treatment of variable structure, i.e. hybrid, models for which HyBrSim has been developed specifically [6]. It does not focus on sophisticated handling of pure continuous and discrete behaviours.

Section 2 reviews the bond graph modelling and simulation approach and identifies the support provided by HyBrSim. Section 3 discusses the hybrid bond graph modelling approach, general hybrid dynamic systems effects and how these are facilitated by HyBrSim. Section 4 describes the export filter to C/C++ and Java and Section 5 presents conclusions and future work.

2 BOND GRAPH MODELLING AND SIMULATION

In HyBrSim, dynamic behaviour of physical systems is modelled by bond graphs [7, 8].

2.1 Bond graph modelling with HyBrSim

Bond graphs model the exchange of energy, *power*, between idealized physical processes, which allows for multi-domain modelling (e.g. electrical, mechanical and hydraulic).

2.1.1 The power domain

Each power connection, *bond*, contains two conjugate variables, effort e and flow f, the product of which constitutes power, that correspond to an intensive variable (e.g. pressure and voltage) and rate of change in an extensive variable (e.g. volume and charge) respectively [9]. There are nine primitive bond graph elements, listed in Table 1, that represent lumped ideal behaviour and exchange energy through ports.

Irreversible processes. The ideal irreversible processes R dissipate energy and generate entropy. For example, in Fig. 1 the intake valve is modelled to have a dissipative effect, R_{in}. If the generated entropy does not affect dynamic behaviour, this port (at present not supported by HyBrSim) is not shown.

Reversible processes. The ideal reversible processes C and I store energy without dissipation and indicate storage of flow and effort respectively, with initial values of generalized displacement q_0 $(= Ce_0)$ and generalized momentum p_0 $(= If_0)$. For example, in Fig. 1 the oil elasticity C_{oil} stores oil compression. These variables capture the *state* of the system. In general, one storage element can communicate energy to many different domains; however, at present, HyBrSim supports only one-port storage.

I08800 © IMechE 2002

Table 1 Bond graph elements

Process	Identification	Properties	Relation
Irreversible	R	Resistance R	$e = Rf$
Reversible	C	Capacitance C	$f = C\frac{de}{dt}$
		Initial value q_0	$e = \frac{1}{C}\int f\,dt + \frac{q_0}{C}$
	I	Inertia I	$e = I\frac{df}{dt}$
		Initial value p_0	$f = \frac{1}{I}\int e\,dt + \frac{p_0}{I}$
Context	Se	Amplitude E	$e = F$
	Sf	Amplitude F	$f = F$
Normal distribution	0		$e_i = e_0$ $\sum_i f_i = 0$
	1		$f_i = f_0$ $\sum_i e_i = 0$
Weighted distribntion	TF	Transformation ratio n	$e_{in} = ne_{out}$ $f_{in} = nf_{out}$
	GY	transformation ratio r	$e_{out} = rf_{in}$ $e_{in} = rf_{out}$

The context. The dynamics of the system environment are not modelled. Instead, interaction is modelled by ideal sources that can be of effort and flow type, i.e. Se and Sf respectively. In Fig. 1, the interaction is by the hydraulic pressure p_{in} and sump pressure p_{smp}, both of Se type.

The junction structure. A junction structure that consists of normal and weighted elements distributes power. The normal elements are of common effort type (e.g. the Kirchhoff current law), 0-junctions, or common flow type (e.g. the Kirchhoff voltage law), 1-junctions. For example, in Fig. 1 the oil pressure in the chamber is the same on the intake path, relief path and the piston and can therefore be modelled by a 0-junction.

The weighted distribution elements are of TF, the transformer type and GY, the gyrator type. The intake valve in Fig. 1 can be modelled by a TF element to modulate the hydraulic power supplied to the cylinder.

Graphical appearance. A HyBrSim bond graph model of the cylinder in Fig. 1 is shown in Fig. 2. The bond graph elements are rectangles with the element type on the left and their name on the right [**10**]. Power bonds (depicted by a harpoon) connect elements that exchange energy, e.g. in Fig. 2 the connection between the Se element and the TF element. In addition, bond graph models may contain modulation effects based on connections that carry no energy. Such a connection may correspond to an individual effort or flow variable that is tapped from a power bond by an active bond (depicted by an arrow). In Fig. 2 an active bond taps the velocity of the piston, m_p, and feeds its value into the displacement element that is part of the signal domain.

2.1.2 *The signal domain*

A number of mathematical operations are available that operate on *signals*, i.e. connections that carry no power (of which active bonds are a subset). In HyBrSim, the signal part of a model is shown in blue (grey in monochrome depiction).

In Fig. 2 the signal part is used to model a PID control law for the piston displacement. This is a geometric state that is used as input to the control part, together with a desired set point. The error between the two is integrated once to implement integral control and the velocity is used directly to add a derivative component. A set of

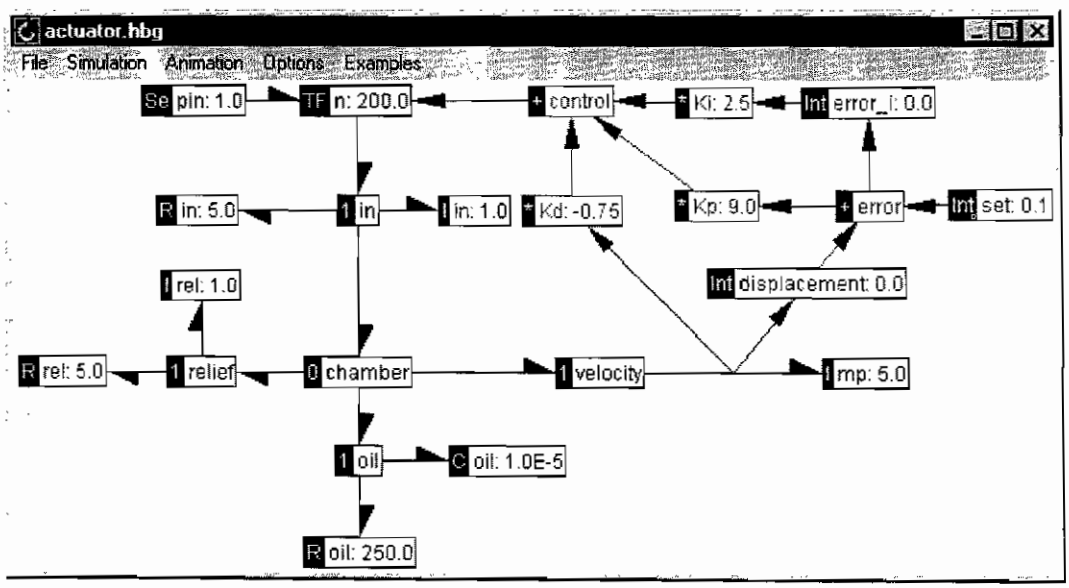


Fig. 2 Bond graph of cylinder with proportional–integral–derivative (PID) controller without the valves being modelled

proportional, integral and derivative gains, K_p, K_i and K_d) respectively, computes the controller output used to modulate the physical process, in this case the hydraulic power supplied to the cylinder. These block diagram elements allow multiple-input s_i and multiple-output s_o signals. Their functionality is given in Table 2.

2.1.3 *User interface*

The HYBRSIM user interface consists of a work space and a bond graph and block diagram toolbox. Connections are made by first clicking the source and then the destination element. The source element determines the type of connection. If it is a bond graph element, a power connection is made. If it is a block diagram element, a signal connection is made. A connection from a block diagram to a bond graph element is allowed in the case of a modulated element, Se, Sf, TF and GY, and automatically interpreted as such.

Right clicking on an element brings up a pop-up menu with entries that include the element name and its properties. In most cases, element properties include a parameter p and initial value x_0 (Tables 1 and 2). For the summing element it contains a drop-down list with all ingoing signals and their sign, p_i, i.e. whether they are to be added or subtracted.

2.2 Simulation with HYBRSIM

Continuous simulation applies a distributed token passing approach and is based on the forward Euler method, $x(t_{k+1}) = x(t_k) + \dot{x}(t_k)\,\Delta T$, where $x(t_k)$ is the state at time t_k, $\dot{x}(t_k)$ its time derivative and ΔT the integration step size. Each port of a bond graph and block diagram element has an attribute that contains the current value of the token at that port [11]. Note that focus of HYBRSIM is on the interaction between continuous and discrete models and, therefore, although simple, the forward Euler integration method is chosen as an initial means to generate continuous behaviours.

Table 2 Bond diagram elements

Element	ID	Properties	Relation
Clock	t	p, x_0	If time $< p$, then $s_o = x_0$; else $s_o =$ time $+ x_0 - p$
Integrator	int	p, x_0	$s_o = p \int \sum_i s_i \, dt$
Step	—	p, x_0	If $\sum_i s_i > p$, then $s_o = 1$; else $s_o = x_0$
Sum	+	p_i	$s_o = \sum_i p_i s_i$
Multiplier	*	p	$s_o = p * \prod_i p_i s_i$
Inverter	inv	p	$s_o = \dfrac{1}{p * \sum_i s_i}$
Square root	sqrt	p	$s_o = \sqrt{\lvert p * \sum_i s_i \rvert}$
sin	sin	p	$s_o = p * \sin(\sum_i s_i)$
cos	cos	p	$s_o = p * \cos(\sum_i s_i)$

2.2.1 *The method*

Firstly, causality is assigned to the power elements based on a sequential causality assignment procedure (SCAP) algorithm [12]. Next, an execution order is determined such that, at a given evaluation k, all inpnt values of an element are known when it is called to compute its output. Modulated elements are handled by introducing pseudo-state behaviour, i.e. the modulation factor is one evaluation (during continuous integration this equals one integration step) delayed. The initial value is user specified.

The execution order is determined by firstly propagating the values of all clock elements so that time modulation is not delayed. Next, the values of sources and storage elements in integral causality are propagated. The values of sources are user provided or, in the case of modulation, given by the pseudo-state value. The values of storage elements in integral causality are computed from their state. Storage elements in derivative causality have no propagation roots and compute a numerical difference approximation of the time derivative variables, $(1/p)[(x_k - x_{k-1})/\Delta T]$, where p is the parameter and x_k the state with x_0 as its initial value. Note that this implies that the first time step is off. Furthermore, zero-order causal paths between resistances cannot be handled. These issues are addressed by a recently implemented approach that is beyond the scope of this paper.

After all effort and flow variables have assigned values, these are propagated into the block diagram model part via active bonds. Together with the integrator elements that propagate their stored value and the values of clock elements, all block diagram variable values are computed.

2.2.2 *Derivative causality*

In the case of derivative causality, there is dependence between storage elements and sources and straightforward propagation of their values does not apply. Instead, algebraic constraints on state variables determine their values. In the case of dependence on sources only, the stored energy can be computed directly. If dependence on other storage elements exists, state values have to be found that are consistent with the algebraic constraints. Two critical issues must be solved:

1. A given set of values has to be made consistent with the algebraic constraints.
2. It must be ensured that one integration step of dynamic behaviour remains consistent with the algebraic constraints.

To achieve issue 1, HYBRSIM implements the conservation-of-state principle [13] as an iteration process between algebraically related storage elements. This circumvents the need to generate the conservation equations from a bond graph [13] and to solve these

I08800 © IMechE 2002

explicitly, either numerically or symbolically, as required in other work [**14**, **15**]. HyBrSim does facilitate equation generation (for a class of bond graphs even in explicit solved form) but the direct implemention of the conservation principle by balancing the exchange of extensity between storage elements seamlessly fits the token propagation simulation approach.

Figure 3 illustrates the iterative process. Here, I_2 is in derivative causality* because of the algebraic dependence between I_1 and I_2 through their common flow. Therefore the stored momentum p_1 and p_2 has to be consistent with

$$\frac{p_1}{I_1} = \frac{p_2}{I_2} \tag{1}$$

Suppose that at a point in time, t_k, $p_1 = 2$ and $p_2 = 1$. These values may be user specified if t_k is the simulation start time or the result of a mode switch at t_k in the case of a hybrid model. For $I_1 = I_2 = 1$, this is inconsistent with equation (1) and iteration first takes part of the stored momentum out of I_1, determined by a convergence factor $\eta = 0.4$, and transfers it into I_2. This leads to $p_1 = 1.6$ and $p_2 = 1.4$ (Table 3). For these momentum values, equation (1) is still not satisfied, and again a (now smaller) part of stored momentum is transferred from I_1 to I_2. This results in $p_1 = 1.52$ and $p_2 = 1.48$. This process continues until $f_1 = f_2$, within some preset numerical margin.

An important observation is that this allows all storage elements to be initialized. Other simulation tools typically allow only initialization of the reduced state, which may lead to differences in the simulation results. For example, if the I elements in Fig. 3 are replaced by C elements, standard simulation packages allow the initialization of only one of them and, depending on the value of E, the state of the other is computed while neglecting to take into account the constraint that only displacement can be added to one if it is taken out of the other.

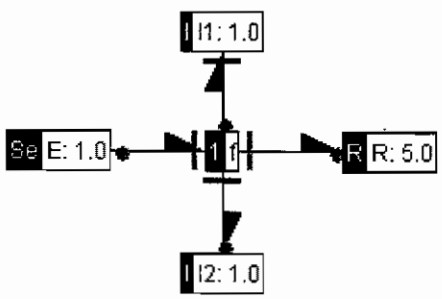


Fig. 3 Bond graph with derivative causality

Table 3 Conservation-of-state iteration, $\eta = 0.4$

Iteration	p_1	p_2^+	p_2	$\eta(p_2^- - p_2)$
0	2	2	1	0.4
1	1.6	1.6	1.4	0.08
2	1.52	1.52	1.48	0.016
3	1.504	1.504	1.496	0.0032
4	1.5008	1.5008	1.4992	0.000 64

* Note that the effort causality is indicated by a perpendicular stroke at the end of a power bond and flow causality by a circle at the other end.

Once consistent values are found, it has to be ensured that future behaviour remains consistent, i.e. for the model in Fig. 3 that equation (1) remains satisfied. In an equation-based system, this is achieved by computing the gradient of behaviour while accounting for the dependence. From equation (1) and

$$\dot{p}_1 = E - \dot{p}_2 - \frac{p_1}{I_1} R \tag{2}$$

it follows, after differentiation of equation (1), that $\dot{p}_1 = [1/(I_1 + I_2)](I_1 E - R p_1) = -3.25$ and one $\Delta T = 0.1$ time step gives $p(t_{k+1}) = p_1(t_k) + \dot{p}_1(t_k)\,\Delta T = 1.175$.

HyBrSim takes a two-step approach. Firstly, a gradient is allowed that results in state values that are inconsistent with the algebraic constraints. Secondly, the iteration approach is applied to find consistent values again. For the two inertias, after consistent values $p_1 = p_2 = 1.5$ are found, the gradient of p_1 is determined while disregarding the influence of I_2, i.e. $e_2 = 0$, which yields $e_1 = -6.5$, and one simulation step $\Delta T = 0.1$ is taken that only updates the state of I_1, $p_1 = 0.85$. This results in values for p_1 and p_2 that violate the algebraic constraint in equation (1) and iteration computes $p_1 = p_2 = 1.175$ as well.

2.2.3 *User interface*

Effort, flow and signal values can be plotted after simulation. These variables are coded by colouring the effort stroke, the flow circle and the fill colour of the signal arrowhead, corresponding to the colours of the traces in a plot. This plot has a small set of rudimentary display features such as data points on–off, autoscale, user-selected maxima and minima for x and y axes separately and a mouse-driven zooming feature. To allow the use of sophisticated plotting features such as those provided by Matlab [**16**], data can be written to a file in standard ASCII format, which includes the evaluation step k and the time stamp t_k for each of the sets of data points†.

In addition, power along each bond can be animated (logarithmic or linear) with positive power based at the harpoon destination and negative power at the source. Figure 4 shows the power distribution in the hydraulic actuator in Fig. 1 at $t = 0.2$ s. The relief valve is considered to be closed and, therefore, not shown and the PID control part is not shown because it does not distribute power. At the point in time represented by Fig. 4, a small fraction of the power supplied by the p_{in} source is dissipated and some is returned by I_{in} (power is based

† All plots in this paper are generated by MATLAB.

I08800 © IMechE 2002

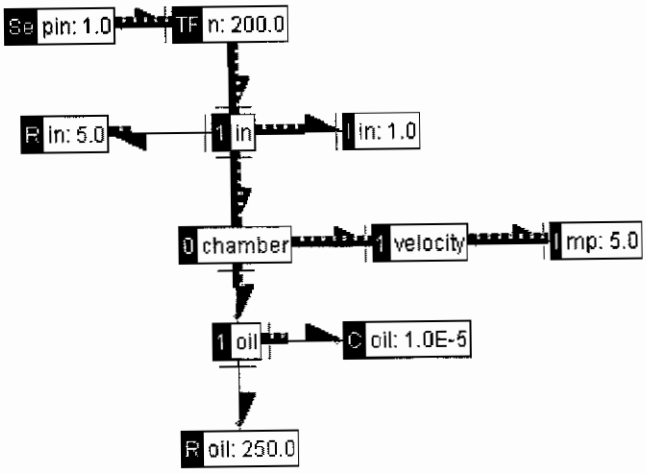


Fig. 4 Logarithmic power distribution at $t = 0.2$ s

at the harpoon source and, therefore, negative). The oil dynamics represented by R_{oil} and C_{oil} have almost reached steady state and hardly affect the power balance (recall the logarithmic scale). After the initial transient, the oil dynamics reach an internal steady state, and the parameters R_{oil} and C_{oil} do not consume any more power. Therefore, to investigate low-frequency behaviour these elements can be removed, e.g. by a singular perturbation related approach for bond graphs [**17**]. In general, such power analyses can be used to aid in model reduction by removing elements with low power consumption [**18**]. Animation can be paused, restarted and continued from final values when the simulation end time was reached.

3 HYBRID BOND GRAPHS

Piecewise linearization of non-linearities and removal of steep gradients may lead to hybrid models with continuous behaviour that is interspersed with discrete mode changes [**19**].

3.1 Hybrid bond graph modelling

Hybrid bond graphs endow the bond graph modelling formalism with a finite state machine (FSM) model part that communicates by means of a controlled junction [**20**].

3.1.1 From discrete to continuous

The controlled junction operates in one of two states to model switching behaviour systematically. When it is *on*, it acts as a normal junction and, when it is *off*, a 0-junction acts as a zero-value effort source and a 1-junction as a zero-value flow source (Fig. 5). This implements ideal switching behaviour (see, for example, references [**20**] to [**22**]) and changes causality on one port when the junction changes its state. Computationally, this is similar to the use of a specific switching element and, therefore, the HYBRSIM implementation to handle hybrid phenomena also applies to *switched bond graphs* [**23**]. Note that dissipation may still occur when junctions change their state, e.g. in the case of a perfectly non-elastic collision.

In the bond graph model in Fig. 2, if the valves are modelled as ideal switches, the corresponding 1-junctions (*in* and *relief*) become controlled junctions with inertial and dissipative effects explicitly modelled.

3.1.2 The discrete event model

The discrete event model part is implemented by local FSMs, one for each controlled junction, that map each of their states on to the *on* and *off* states. The graphical representation is a state transition diagram that is associated with the junction property. For example, Fig. 6 shows the FSM for the relief valve mechanism in Fig. 1. When the net pressure p_{net} crosses a threshold value p_{th}, $p_{net} = p_{cyl} - p_{th} < 0.0$ the relief valve opens, i.e. the corresponding controlled junction comes on. Note that the threshold value can also be modelled in the FSM. For example, the relief valve closes again when the pressure has subsided and crosses another threshold ($p_{net} > 25.0$).

This leads to the behaviour in Fig. 7. During a control manoeuvre, the intake valve closes inadvertently at $t = 0.2$ s. Shortly thereafter, the oil parameters have built up a pressure in the cylinder chamber that exceeds the safety threshold. Consequently, the relief valve opens for a short duration (see Fig. 7b) until the pressure has subsided and then closes again. This leads again to too high a pressure in the cylinder and the same procedure repeats, after which the piston velocity has reduced to a value that can safely decay to zero. This 'stuttering' is typical of relief valve behaviour.

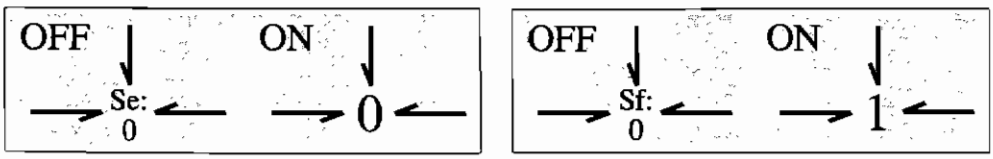
Each FSM associated with a controlled junction has an initial state, indicated by a shaded background, and an active state that is highlighted. States can be added and removed but are always of the *on* and *off* type.

Because FSM switching effects are included locally, no global analysis of the modes of behaviour is required. Although this avoids pre-enumeration (which can be prohibitively complex because of the state explosion), it requires run-time facilities to determine the new global mode dynamically. This includes causal analysis which may lead to run-time changes in the complexity of the underlying system of equations (i.e. derivative causality may emerge).

3.1.3 From continuous to discrete

Block diagram signals and active bonds that may cause a controlled junction to change its state are connected to this junction and show up in the state transition diagram as *signal ports*. Signal ports constitute *crossing functions* that compare the value of the corresponding

I08800 © IMechE 2002



(a) 0-junction. (b) 1-junction.

Fig. 5 The controlled junction types

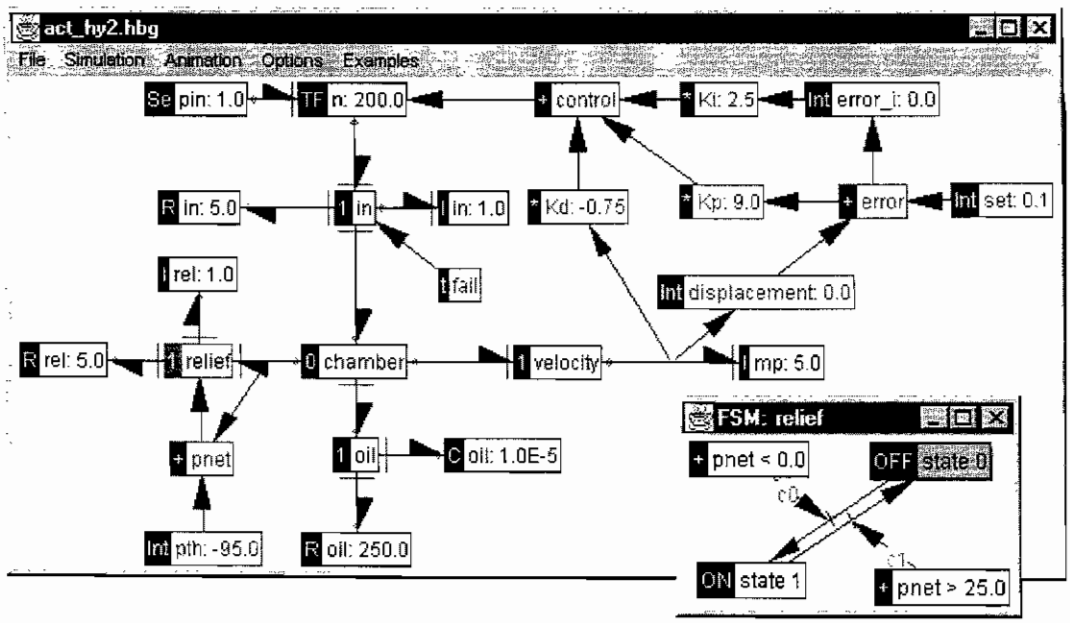
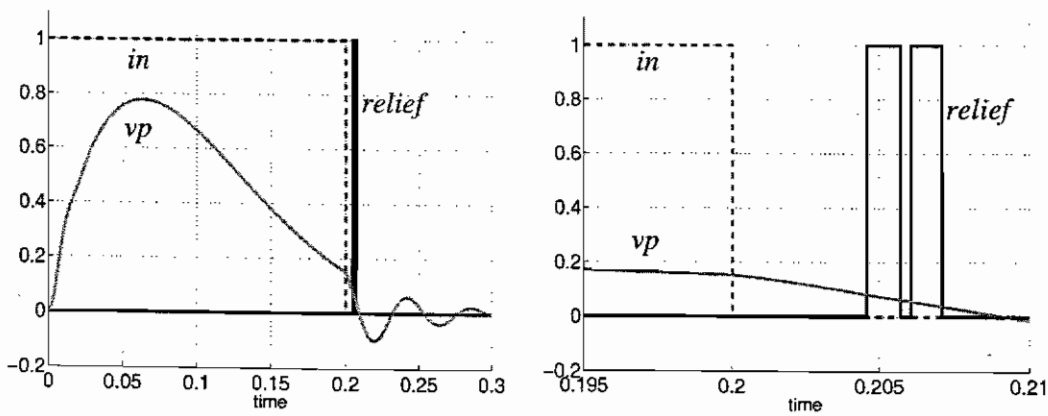


Fig. 6 Hybrid bond graph model of actuator with the valves modelled by controlled junctions



(a) Aborted control manoeuver. (b) Zoomed in on switching behaviour.

Fig. 7 Simulation of the model in Fig. 6

variable in the bond graph, x, with a threshold δ. Two inquiries are allowed: (a) the new truth value of the crossing function can be requested and (b) it can be queried whether a crossing or change in the sign of the crossing function z (-1, 0 and 1 for below, at and above the threshold respectively), takes place, indicated by setting the inquiry Boolean variable cross to true. The first option is used to find the new state of all controlled junctions and the second to halt continuous simulation when a discrete event is generated.

The comparison can be of the types listed in Table 4 and results in a Boolean output (true and false) that can be connected to transitions between states. Several signal ports may be connected to one transition to form a

I08800 © IMechE 2002

Table 4 Crossing variable x and its threshold δ comparisons

Request	Return value
$x<\delta$	$x<\delta-\varepsilon \wedge (\neg\text{cross} \vee z>-1)$
$x\leqslant\delta$	$x\leqslant\delta+\varepsilon \wedge (\neg\text{cross} \vee z>0)$
$x=\delta$	$(\text{cross} \wedge z>0 \wedge x\leqslant\delta+\varepsilon) \vee$ $(\text{cross} \wedge z<0 \wedge x\geqslant\delta-\varepsilon) \vee$ $(\neg\text{cross} \wedge \lvert x-\delta\rvert\leqslant\varepsilon)$
$x\geqslant\delta$	$x\geqslant\delta-\varepsilon \wedge (\neg\text{cross} \vee z<0)$
$x>\delta$	$x>\delta+\varepsilon \wedge (\neg\text{cross} \vee z<1)$

logical conjunction. A signal port with output true generates a discrete event that may enable a transition and, when it does, force it to occur immediately (i.e. the FSM implements 'must fire' semantics).

3.2 Hybrid bond graph simulation

Simulation of hybrid systems has to deal with a number of idiosyncrasies [**6**].

3.2.1 Event detection and location

When continuous variables exceed threshold values, as specified in the signal ports, HYBRSIM uses a bisectional search to find when the first event (in general there are multiple events) in the ΔT interval occurs. If an event is detected, the step size is reduced from ΔT to δt_m, where δt_m is computed on the basis of whether an event occurs, $\sigma=1$, or not, $\sigma=0$, iu the interval δt_i as follows:

$$\delta t_{i+1}=\delta t_i+\Delta t_i(1-\sigma)$$
$$\Delta t_{i+1}=\tfrac{1}{2}\Delta t_i \qquad (3)$$

The initial values for this iteration are $\delta t_0=0$ and $\Delta t_0=\Delta T$, and the iteration terminates after a fixed number of *a priori* prescribed steps, m. This method is robust and guaranteed to find the first event with a pre-specified accuracy, provided that the crossing function does not have an even number of roots on the δt_i intervals. Because HYBRSIM requires the user to determine the step size of the Euler method, ΔT, this is the user's responsibility.

3.2.2 Reinitialization

During event iteration, algebraic dependences of storage elements may arise that discontinuously change the state values using the iteration procedure in Section 2.2.2.

3.2.3 Event iteration

When an event occurs and a FSM changes its state, a further transition from this new state may be immediately enabled, requiring a new evaluation of the FSM and causing a discrete iteration phase. Furthermore, in the case of *dynamic coupling*, when a discrete state change causes a junction to switch between its *on* and *off* state (not each state change is necessarily one between *on* and *off*), effort and flow variables in the bond graph may change their values and, when these values are propagated into the signal ports, they may enable further state transitions. Therefore, before continuous simulation can resume, iteration between the discrete model parts and between the discrete and continuous model parts is necessary to find first a consistent, stable, continuous and discrete state.

A priori and a posteriori values. Algebraic constraints may be activated and deactivated during one sequence of discrete state changes and change the state values of storage elements. In the bond graph model it may or may not be desired to return to the original values before the discrete state changes [**5**]. A check box specifies whether a signal port generates events based on *a priori* or *a posteriori* values of x around a discontinuity, shown by a $-$ or $+$ sign respectively on the left in the signal port (see the *relief* FSM in Fig. 6). In case of *a posteriori* conditions, the new model variable values computed as described in Section 2.2.1* are adopted by the signal port. In the case of *a priori* switching conditions, the values are only adopted after the system state is updated and so discontinuous changes are effected. The application of this is illustrated next.

Mythical modes. Consider the situation where the intake valve closes while $v_p>0$, as discussed previously. The oil viscosity R_{oil} immediately causes a large pressure in the cylinder chamber and this may cause the relief valve to open without any noticeable change in piston velocity. If the oil viscosity R_{oil} and elasticity C_{oil} are abstracted away (simply removed from the bond graph in Fig. 6), the piston velocity becomes zero in the mode where both valves are closed. If the required (now infinitely) large pressure causes the relief valve to open, the velocity would remain at zero, which differs from the behaviour of the more detailed model. Instead, the state values before the sequence of switches started by closing the intake valve should be transferred to the mode where the relief valve is opened. The intermediate mode where both valves are closed is called a *mythical mode* [**5**, **24**]. In this example, if the opening of the relief valve is based on *a posteriori* values, the state vector is not updated yet when it is inferred that the valve opens and, therefore, the state is transferred correctly.

Pinnacles. When the intake valve closes, the viscous pressure may not suffice to open the relief valve. Instead, the elasticity may further build up pressure and there is

* If two consecutive evaluations occur at the same point in time, i.e. $t_k=t_{k-1}$, the difference approximation to compute efforts and flows of elements in derivative causality is formed by using $\xi \ll \Delta T$.

108800 © IMechE 2002

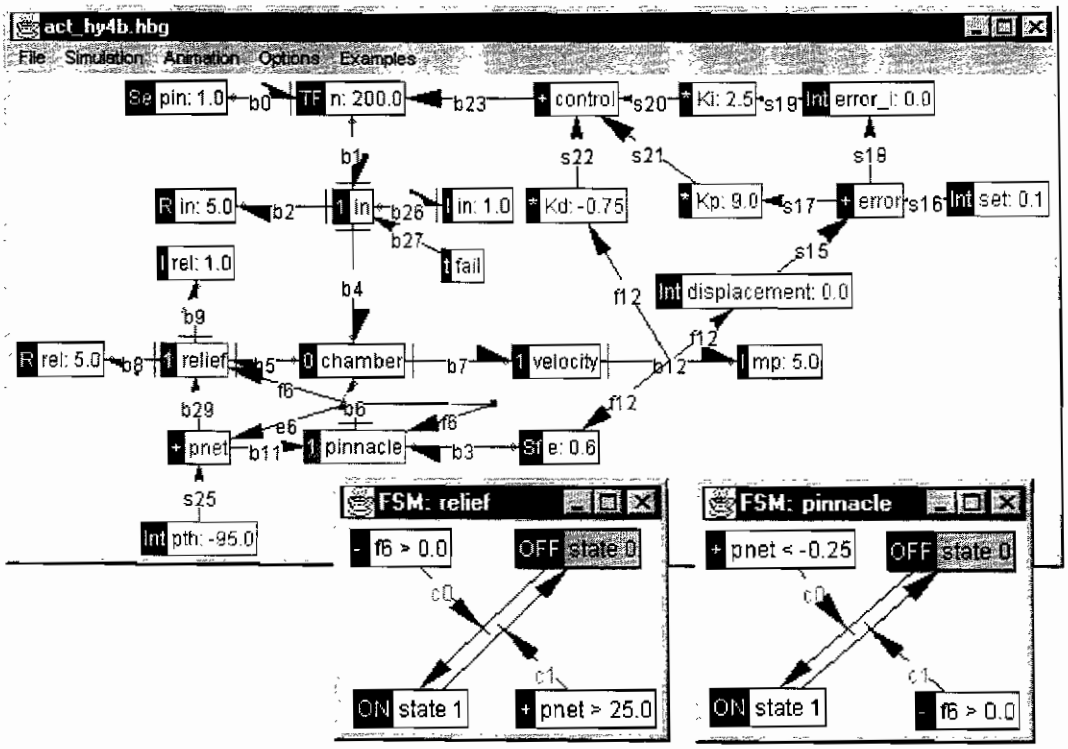


Fig. 8 Actuator with explicit state jump

a significant change in elevator velocity before the relief valve opens (see Fig. 7b). If the oil parameters are abstracted away, a coefficient of restitution, ε, is used that depends on the dissipation of the original parameters, to compute the change in elevator velocity, $v_p = \varepsilon v_p^-$, where v_p^- is the value immediately before switching started. In Fig. 8 this is implemented by the Sf element with $\varepsilon = 0.6$.*

The algebraic equation is activated using *a posteriori* values and deactivated using *a priori* values. The relief valve is opened on the basis of *a priori* values to ensure that the computed velocity change is effected. The stuttering behaviour shown in Fig. 7 now occurs instantaneously at the same point in time, because the oil parameters are not present any longer. A sequence of activations of the algebraic restitution constraint and opening of the relief valve reduces the piston velocity before it can be safely set to zero, shown in Fig. 9 by the data points with decreasing velocity at the switching time. Note that the mode where both valves are closed has a mythical incarnation; otherwise, the velocity transferred to the mode where *pinnacle* is *on* is zero, and no stuttering behaviour takes place.

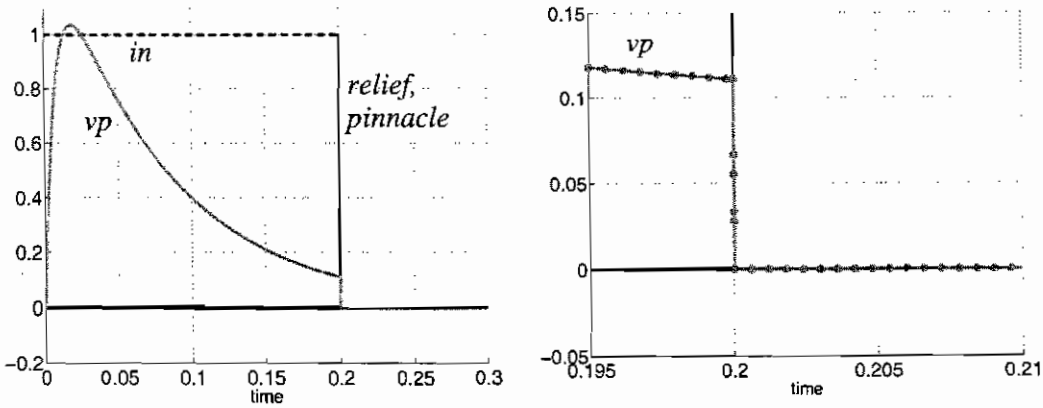
Also note that in Fig. 9b the data points are evenly spaced during continuous integration (distance ΔT), but there is a shorter time step to reach $t = 0.2$ s, because of root finding, described in Section 3.2.1.

3.2.4 *Impulse comparison*

In the model of the hydraulic cylinder in Fig. 8, when both valves are closed and the piston has non-zero velocity, a pressure spike occurs that takes the form of a Dirac pulse and has an infinite magnitude. Therefore, the threshold pressure is always exceeded, regardless of the piston velocity. In the more detailed model whether the pressure threshold is exceeded depends on the piston velocity [**19**]. This can be included in a first-order approximation by comparing impulse areas, i.e. the change in piston velocity.

HyBrSim explicitly compares impulsive variables based on their areas. The threshold values as specified in the signal ports are interpreted as areas as well. Therefore, in the cylinder model in Fig. 8, because the pressure threshold is given in the signal port, the impulse area is tested. If, however, the threshold is given by use of a constant value on the work space, this is interpreted as a normal non-impulsive variable and, therefore, disregarded when impulses occur. In Fig. 8 the int element is used for the continuous behaviour threshold and the threshold value in the signal port for the impulse area comparison. Therefore, if the instantaneous change in velocity is less than -0.25, the relief valve opens and, when during continuous behaviour the pressure falls

* Note the unique labels of connections and that causal conflicts between sources and junctions that are *off* are not terminal.



(a) Aborted control manoeuver. (b) Zoomed in on switching behaviour.

Fig. 9 Simulation of the model in Fig. 8

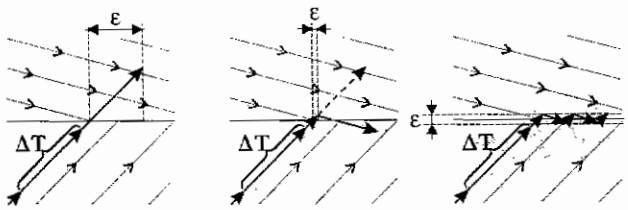


Fig. 10 Chattering behaviour

below the combined threshold value of −95.25, the relief valve opens as well.

3.2.5 *Further issues*

Other phenomena in hybrid dynamic systems behaviour that at present cannot be handled by HYBRSIM are (a) chattering and (b) aborted projections.

Chattering. In hybrid dynamic system behaviour, after a mode switch from an initial mode is completed and a stable and converged new discrete and continuous state is found, an infinitesimal small time step may lead to a mode change back to the initial mode.* The next infinitesimal time step may again switch to the mode reached from the initial one. This means the system has reached a *switching surface* in phase space where the gradients of behaviour point towards the surface in both modes (Fig. 10). If this occurs, simulation reduces the step size to its smallest possible value repeatedly and, therefore, simulation becomes (many times prohibitively) slow. At present, in HYBRSIM this can only be circumvented by disabling root finding, which causes a larger error. A more sophisticated algorithm has been developed in other work [**25**].

Aborted projections. The dependence of a storage element may cause a discontinuous change in its state variable. Before this change is completed, a further mode change may be induced by an intermediate value and this intermediate value should be transferred to the new mode instead. This can be implemented by, for example, a bisectional algorithm.

* Note that this differs from an immediate switeh back, in which ease no stable diserete state exists.

4 EXPORT FILTERS

HYBRSIM can export a hybrid bond graph model as an explicit ODE or implicit DAE in Java or C/C++ to generate model behaviours using sophisticated numerical solvers.

4.1 Explicit equations

Explicit equations take the form $\dot{x} = f(x, u, t)$, with x the state variables, u input and t time and are generated by a straightforward graph traversal procedure after the execution order of the bond graph model is established, provided that no derivative causality exists. Otherwise, the iteration procedure described in Section 2.2.2 is included as an additional method, executed after the system of equations is evaluated. For example, for the model in Fig. 3, the variables `I1.f` and `I2.f` are algebraically related as `I1.f=I2.f`. To facilitate the iteration process, this is generated as a fixed point constraint where the new value of `I2.f`, namely `I2.nf`, has to equal `I1.f`. Applying the conservation constraint leads to the changes `I1.df` and `I2.df` of `I1.f` and `I2.f` respectively, and the iteration is repeated until these changes are within some preset numerical margin. The generated method for the model in Fig. 3 is

```
void 2i::iterate() {
     double I2.nf = I1.f;
     I1.df = (-I2.I * (I2.nf - I2.f)) /
     I1.I;
     I2.df = I2n - I2.f;
```


I08800 © IMechE 2002

and iteration conforms to Table 3, with the convergence factor being set by the calling method.

The drawback of an explicit solution is that it is mode dependent because the change in state of a controlled junction may affect the computational causality. Therefore, the system of equations, f_α, has to be derived for each global mode α. At present, a global causal analysis is performed for permutations of up to three controlled junction states.

4.2 Implicit equations

A DAE system with equations of the form $0 = f(\dot{x}, x, u, t)$ is more flexible. Numerical solvers such as the differential–algebraic system solver (DASSL) [4] provide the $\dot{x}$, x, u and t arguments and require a vector f of return values.

In this form, modulation need not be treated as a pseudo-state and the iteration process is not required because the algebraic constraints can be included in the implicit formulation. Although this means that the system of equations becomes more difficult to handle, i.e. it is of a higher *index* [26], solvers such as DASSL are typically able to handle this complexity provided that consistent initial values are available.

An additional advantage of the implicit formulation is that controlled junction state changes can be included by switched equations and, therefore, global analysis is not required. To this end, the equations for each controlled 0-junction are formulated as follows:

$$0 = L \sum_i p_i f_i + (1 - L)e \tag{4}$$

where $L \in \{0, 1\}$ is a mode selection variable that is 1 if the junction is *on* and 0 if it is *off*. The variables $p_i \in \{-1, 1\}$ indicate the orientation of each power bond. Furthermore, equations

$$0 = e_i - e \tag{5}$$

are added for each power port i. In the case when $L = 1$, the standard equations for a 0-junction in implicit form are active. When $L = 0$, equation (4) results in $e = 0$ and, combined with equation (5), the efforts on all ports are zero. The formulation for controlled 1-junctions is the dual of this.

For the model in Fig. 3, with a switch to turn the 1-junction *off* when the effort of R exceeds 0.5, the equations are

```
E.e = E.in;
f.L = (f.state == True) ? 1.0 : 0.0;
I1.res = -I1.f + f.f;
I1.e = I1.derf * I1.I;
R.f = f.f;
R.e = R.f * R.R;
I2.res = -I2.f + f.f;
I2.e = I2.derf * I2.I;
f.res = (-E.e + I1.e + R.e + I2.e) * f.L + (f.L - 1) * f.f;
```

with

$$u = [E.in], \qquad x = \begin{bmatrix} I1.f \\ I2.f \\ f.f \end{bmatrix}, \qquad f = \begin{bmatrix} I1.res \\ I2.res \\ f.res \end{bmatrix} \tag{6}$$

and *derf* the time derivative of the variable f. The required initial values can be computed by (a) using a dedicated routine explicitly generated by HYBRSIM but which suffers from the combinatorial restriction (the computations differ per mode) and (b) using a more general (and computationally more expensive) decomposition method that relies on the Weierstrass normal form [27].

5 CONCLUSIONS

Bond graphs are a powerful formalism to model the continuous behaviour of physical systems in different domains. In many cases the dynamics contain non-linearities or steep gradients that may be best handled by a discontinuous approximation. Hybrid bond graphs facilitate this by supporting a junction that is controlled by a finite state machine to operate as either a normal junction or a zero value source.

HYBRSIM is a hybrid bond graph modelling and simulation tool that is specifically developed to handle phenomena in the mixed continuous–discrete systems realm. It performs event detection and location based on a bisectional search, handles run-time causality changes, including derivative causality, performs physically consistent (re-)initialization and supports two types of event iteration because of dynamic coupling. Continuous behaviour is handled by a simple forward Euler integration scheme. Therefore, hybrid bond graph models can be exported as Java and C/C++ code to be used by sophisticated numerical solvers where discontinuities are included as switched equations (i.e. pre-enumeration is not required).

ACKNOWLEDGEMENTS

The graphical appearance of the HYBRSIM elements is based on a bond graph simulator by Róbert Bajzát [28]. The author is supported by a grant from the Deutsche Forschungsgemeinschaft Schwerpunktprogramm KONDISK.

REFERENCES

1 **Andersson, M.** Object-oriented modeling and simulation of hybrid systems. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1994.

2 **Cellier, F. E., Elmqvist, H.** and **Otter, M.** Modelling from

I08800 © IMechE 2002

physical principles. In *The Control Handbook* (Ed. W. S. Levine), 1996, pp. 99–107 (CRC Press, Boca Raton, Florida).

3 **Bujakiewicz, P.** Maximum weighted matching for high index differential algebraic equations. PhD thesis, Technische Universiteit te Delft, Delft, The Netherlands, 1994.

4 **Petzold, L. R.** A description of DASSL: a differeutial/ algebraic system solver. Technical Report SAND82-8637, Sandia National Laboratorics, Livermore, California, 1982.

5 **Mostermau, P. J.** Hybrid dynamic systems: a hybrid bond graph modelling paradigm and its application in diagnosis. PhD thesis, Vanderbilt University, Nashville, Tennessee, 1997.

6 **Mosterman, P. J.** An overview of hybrid simulation phenomena and their support by simulation packages. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, Vol. 1569 (Eds F. W. Vaandrager and J. H. van Schuppen), March 1999, pp. 164–177 (Springer-Verlag, Berlin).

7 **Karnopp, D. C., Margolis, D. L.** and **Rosenberg, R. C.** *Systems Dynamics: A Unified Approach*, 2nd edition, 1990 (John Wiley, New York).

8 **Paynter, H. M.** *Analysis and Design of Engineering Systems*, 1961 (MIT Press, Cambridge, Massachusetts).

9 **Falk, G.** and **Ruppel, W.** *Energie und Entropie: Eine Einführung in die Thermodynamik*, 1976 (Springer-Verlag, Berlin).

10 **Bajzát, R.** Building geometric information into and getting out of mechanical systems described by bond graphs. In *MicroCAD'96*, University of Miskolc, 1996.

11 **Mosterman, P. J.** and **Biswas, G.** A Java implementation of an environment for hybrid modelling and simulation of physical systems. In Proceedings of the International Conference on *Bond Graph Modeling and Simulation* (*ICBGM'99*), Simulation Series, Vol. 31, San Francisco, California, January 1999, pp. 157–162 (Society for Computer Simulation, San Diego, California).

12 **van Dijk, J.** On the role of bond graph causality in modelling mechatronic systems. PhD dissertation, University of Twente, The Netherlands, 1994.

13 **Mosterman, P. J.** and **Biswas, G.** Formal specifications from hybrid bond graph models. In Qualitative Reasoning Workshop, Cortona, Italy, June 1997, pp. 131–142.

14 **Edström, K.** Switched bond graphs: simulation and analysis. PhD thesis, Linköping University, Linköping, Sweden, 1999.

15 **Mosterman, P. J.** State space projection onto linear DAE manifolds using conservation principles. Technical Report R262-98, Institute of Robotics and System Dynamics, DLR Oberpfaffenhofen, Wessling, Germany, 1998.

16 *MATLAB, The Language of Technical Computing*, May 1997 (The MathWorks, Natick, Massachusetts).

17 **Dauphin-Tanguy, G., Borne, P.** and **Lebrun, M.** Order reduction of multi-time scale systems using bond graphs, the reciprocal system and the singular perturbation method. *J. Franklin Inst.*, 1985, **319**(1–2), 157–171.

18 **Minten, W., Vranckx, S., De Moor, B.** and **Vandewalle, J.** Bondlab, a MATLAB based GUI for bond graph modelling. *J. A*, 1997, **38**(3), 11–15.

19 **Mosterman, P. J.** and **Biswas, G.** Towards procedures for systematically deriving hybrid models of complex systems. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science (Eds N. Lynch and B. Krogh), 2000, pp. 324–337 (Springer-Verlag, Berlin).

20 **Mosterman, P. J.** and **Biswas, G.** A theory of discontinuities in dynamic physical systems. *J. Franklin Inst.*, January 1998, **335B**(3), 401–439.

21 **Kassakian, J. G., Schlecht, M. F.** and **Verghese, G. C.** *Principles of Power Electronics*, 1991 (Addison-Wesley, Reading, Massachusetts).

22 **Strömberg, J.-E., Top, J.** and **Söderman, U.** Variable causality in bond graphs caused by discrete effects. In Proceedings of the International Conference on *Bond Graph Modeling and Simulation* (*ICBGM'93*), Simulation Series, Vol. 25, La Jolla, California, January 1993, pp. 115–119 (Society for Computer Simulation, San Diego, California).

23 **Söderman, U.** and **Strömberg, J.-E.** Switched bond graphs: towards systematic composition of computational models. In Proceedings of the 1995 International Conference on *Bond Graph Modeling and Simulation* (*ICBGM'95*), Simulation Series, Vol. 27, No. 1, Las Vegas, Nevada, January 1995, pp. 73–79 (Society for Computer Simulation, San Diego, California).

24 **Nishida, T.** and **Doshita, S.** Reasoning about discontinuous change. In Proceedings of *AAAI-87*, Seattle, Washington, 1987, pp. 643–648.

25 **Mosterman, P. J., Zhao, F.** and **Biswas, G.** Sliding mode model semantics and simulation for hybrid systems. In *Hybrid Systems V*, Lecture Notes in Computer Science (Eds P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode and S. Sastry), 1999, pp. 218–237 (Springer-Verlag, Berlin).

26 **Gear, C. W.** Differential–algebraic equation index transformations. *SIAM J. Scient. Statist. Computing*, 1988, **9**(1), 39–47.

27 **Mosterman, P. J.** Implicit modeling and simulation of discontinuities in physical system models. In Proceedings of the 4th International Conference on *Automation of Mixed Processes: Hybrid Dynamic Systems* (Eds S. Engell, S. Kowalewski and J. Zaytoon), 2000, pp. 35–40.

28 **Bajzát, R.** http://gold.uni-miskolc.hu/~iitbajzi/bond/.

I08800 © IMechE 2002