

HYBRSIM – A Modeling and Simulation Environment for Hybrid Bond Graphs

Pieter J. Mosterman*

Institute of Robotics and Mechatronics

DLR Oberpfaffenhofen

D-82230 Wessling

Pieter.J.Mosterman@dlr.de

December 1, 2000

Keywords

hybrid systems, physical system modeling, bond graphs, hybrid bond graphs, hydraulic actuators

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 behavior that is too complex for numerical analysis (e.g., because of nonlinearities or steep gradients). HYBRSIM is a tool for hybrid bond graph modeling and simulation implemented in Java and 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).

1 Introduction

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

* Pieter J. Mosterman is supported by a grant from the DFG Schwerpunktprogramm KONDISK.

1.1 Stiff Systems

Continuous physical system models may contain nonlinearities 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.

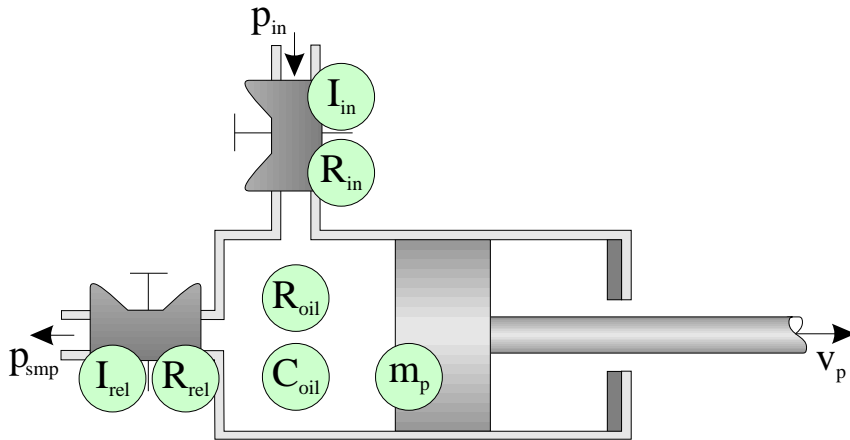


Figure 1: Hydraulic cylinder with relief valve.

Initially, the relief valve may be closed and when the intake valve closes completely, there is no flow of oil into the chamber, causing the piston velocity to quickly become 0. 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 become of a mixed continuous/discrete, *hybrid*, nature [5]. Hybrid systems typically operate in piecewise continuous *modes*, modeled by ODEs and DAEs. Mode changes are most conveniently modeled

by activating and deactivating constituent equations of model components which may change causality. For example, when the intake valve in Fig. 1 closes, its constituent equation changes from enforcing a 0 pressure drop across it to enforcing 0 flow. These causal changes may result in dynamic (i.e., run-time) changes in the state vector.

Two situations can be classified;

- state variables become dependent on exogeneous variables, and
- 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 0, 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, behavior in between. In general, (de)activating blocks of equations may result in a sequence of consecutive mode changes that has to converge before continuous behavior resumes [5].

1.3 HYBRSIM

HYBRSIM (*Hybrid Bond gRaph Simulator*) is a modeling and simulation environment to handle hybrid behaviors, 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 is developed specifically [6]. It does not focus on sophisticated handling of pure continuous and discrete behaviors.

Section 2 reviews the bond graph modeling and simulation approach and identifies the support provided by HYBRSIM. Section 3 discusses the hybrid bond graph modeling 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 Modeling and Simulation

In HYBRSIM, dynamic behavior of physical systems is modeled by bond graphs [7, 8].

Table 1: Bond graph elements.

process	ID	properties	relation
irreversible	R	resistance, R	$e = Rf$
reversible	C	capacitance, C initial value, q_0	$f = C \frac{de}{dt}$ $e = \frac{1}{C} \int f dt + \frac{q_0}{C}$
	I	inertia, I initial value, p_0	$e = I \frac{df}{dt}$ $f = \frac{1}{I} \int e dt + \frac{p_0}{I}$
context	Se	amplitude, E	$e = E$
	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 distribution	TF	transformation ratio, n	$e_{in} = n e_{out}$ $f_{in} = n f_{out}$
	GY	transformation ratio, r	$e_{out} = r f_{in}$ $e_{in} = r f_{out}$

2.1 Bond Graph Modeling With HYBRSIM

Bond graphs model the exchange of energy, *power*, between idealized physical processes, which allows for multi-domain modeling (e.g., electrical, mechanical, 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 of 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 behavior 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 modeled to have a dissipative effect, R_{in} . If the generated entropy does not affect dynamic behavior, this port (at present not supported by HYBRSIM) is not shown.

Reversible Processes Ideal reversible processes store energy without dissipation. These are marked C and I to indicate stored flow and stored effort, respectively, with initial values generalized displacement, q_0 ($= C e_0$) and generalized momentum p_0 ($= I f_0$). 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 only supports one port

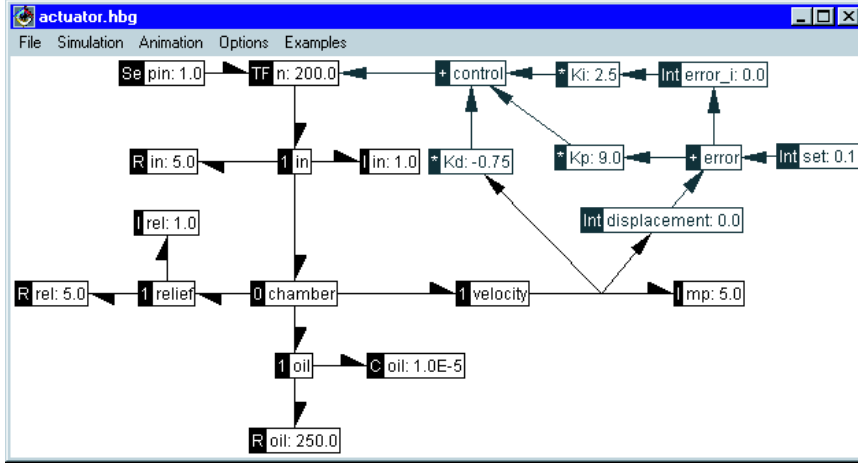


Figure 2: Bond graph of cylinder with PID controller without the valves being modeled.

storage.

The Context The dynamics of the system environment are not modeled. Instead, interaction is modeled 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., Kirchoff's current law), 0-junctions, or common flow type (e.g., Kirchoff's 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 modeled by a 0-junction.

The weighted distribution elements are of TF , the transformer, and GY , the gyrator type. The intake valve in Fig. 1 can be modeled 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.

Table 2: Block 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	$s_o = p * \prod_i s_i$
inverter	inv	p	$s_o = \frac{1}{p * \sum_i s_i}$
square root	$sqrt$	p	$s_o = \sqrt{ p * \sum_i s_i }$
sin	sin	p	$s_o = p * sin(\sum_i s_i)$
cos	cos	p	$s_o = p * cos(\sum_i s_i)$

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 (gray 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, along with a desired setpoint. 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 proportional, integral, and derivative gains (K_p , K_i , K_d) 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 output, s_o , signals. Their functionality is given in Table 2.

2.1.3 User Interface

The HYBRSIM user interface consists of a workspace 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 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 (Table 1 and Table 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].

2.2.1 The Method

First, causality is assigned to the power elements based on a sequential causality assignment procedure (SCAP) like algorithm [12]. Next, an execution order is determined such that, at a given evaluation, k , all input values of an element are known when it is called to compute its output. Modulated elements are handled by introducing pseudo state behavior, 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 first propagating the values of all clock elements so 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 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 are no propagation roots and compute a numerical difference approximation of the time derivative variables, $\frac{1}{p} \frac{x_k - x_{k-1}}{\Delta T}$, where p is the parameter and x_k the state with x_0 as its initial value. Note that this implies 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. Along 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 case of derivative causality, there is dependency between storage elements and sources and straightforward propagation of their values does not apply. Instead, algebraic constraints on state variables determine their values. In case of dependency on sources only, the stored energy can be computed directly. If dependency on other storage elements

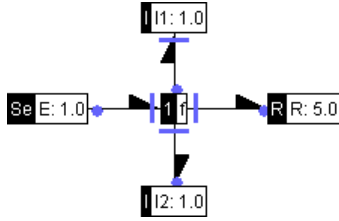


Figure 3: Bond graph with derivative causality.

exists, state values have to be found that are consistent with the algebraic constraints. Two critical issues must be solved: (i) a given set of values has to be made consistent with the algebraic constraints, and (ii) it must be ensured that one integration step of dynamic behavior remains consistent with the algebraic constraints.

To achieve (i), HYBRSIM implements the conservation of state principle [13] as an iteration process between algebraically related storage elements. In Fig. 3, I_2 is in derivative causality¹ because of the algebraic dependency between I_1 and I_2 through their common flow. So the stored momentum p_1 and p_2 has to be consistent with

$$\frac{p_1}{I_1} = \frac{p_2}{I_2}. \quad (1)$$

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 case of a hybrid model. For $I_1 = I_2 = 1$, this is inconsistent with Eq. (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 $p_2 = 1.4$, see Table 3. For these momentum values, Eq. (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 till $f_1 = f_2$, 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.

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

¹Note 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.

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	0.00064

for the dependency. From Eq. (1) and

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

it follows, after differentiation of Eq. (1), that $\dot{p}_1 = \frac{1}{I_1+I_2}(I_1E - Rp_1) = -3.25$ and one $\Delta T = 0.1$ time step gives $p_1(t_{k+1}) = p_1(t_k) + \dot{p}_1(t_k)\Delta T = 1.175$.

HYBRSIM takes a two step approach. First, a gradient is allowed that results in state values that are inconsistent with the algebraic constraints. Second, 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 Eq (1) and iteration computes $p_1 = p_2 = 1.175$ as well.

2.2.3 User Interface

Effort, flow, and signal values can be graphed after simulation. These variables are coded by coloring the effort stroke, the flow circle, and the fill color of the signal arrowhead, corresponding to the colors of the traces in a graph. This graph has a small set of rudimentary display features such as data points on/off, autoscale, user selected maxima and minima for x and y -axis separately, and a mouse driven zooming feature. To allow the use of sophisticated plotting features such as those provided by MATLAB [14], 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.²

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. After some initial transient,

²All plots in this paper are generated by MATLAB.

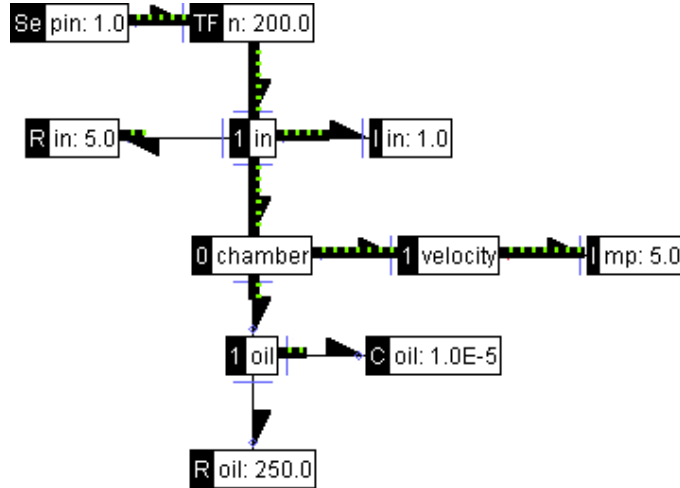


Figure 4: Logarithmic power distribution at $t = 0.2$ s.

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 behavior these elements can be removed, e.g., by a singular perturbation related approach for bond graphs [15]. In general, such power analyses can be used to aid in model reduction by removing elements with low power consumption [16]. Animation can be paused, restarted, and continued from final values when the simulation end time was reached.

3 Hybrid Bond Graphs

Piecewise linearization of nonlinearities and removal of steep gradients may lead to hybrid models with continuous behavior that is interspersed with discrete mode changes [17].

3.1 Hybrid Bond Graph Modeling

Hybrid bond graphs endow the bond graph modeling formalism with a finite state machine model part that communicates by means of a controlled junction [18].

3.1.1 From Discrete to Continuous

The controlled junction operates in one of two states to systematically model switching behavior. When it is *on*, it acts as a normal junction, and when it is *off* a 0-junction acts as a 0 value effort source and a 1-junction as a 0 value flow source (see Fig. 5). This implements ideal switching behavior (e.g., [19, 18, 20]) and changes causality on one port when the junction changes its state. Note that dissipation may still occur when junctions change their state, e.g., in case of a perfectly nonelastic collision.



Figure 5: The controlled junction types.

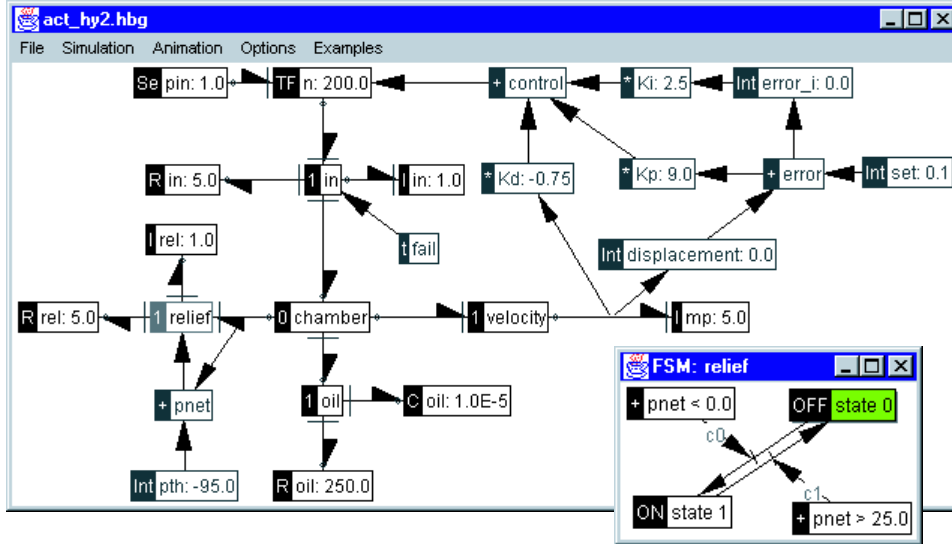


Figure 6: Hybrid bond graph model of actuator with the valves modeled by controlled junctions.

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

3.1.2 The Discrete Event Model

The discrete event model part is implemented by local finite state machines (FSM), one for each controlled junction, that map each of their states onto the *on* and *off* states. The graphical representation is a state transition diagram that is associated to 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 modeled 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 behavior in Fig. 7: During a control maneuver, 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. 7(b), till 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 0. This ‘stuttering’ is typical for relief valve behavior.

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 behavior is required. Though 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 variable in the bond graph, x , with a threshold, δ . Two inquiries are allowed: (i) the new truth value of the crossing function can be requested and (ii) it can be queried whether a crossing, or change of 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 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 idiosyncracies [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

Table 4: Crossing variable, x , and its threshold, δ , comparisons.

request	return value
$x < \delta$	$x < \delta - \epsilon \wedge (\neg cross \vee z > -1)$
$x \leq \delta$	$x \leq \delta + \epsilon \wedge (\neg cross \vee z > 0)$
$x = \delta$	$(cross \wedge z > 0 \wedge x \leq \delta + \epsilon) \vee$ $(cross \wedge z < 0 \wedge x \geq \delta - \epsilon) \vee$ $(\neg cross \wedge x - \delta \leq \epsilon)$
$x \geq \delta$	$x \geq \delta - \epsilon \wedge (\neg cross \vee z < 0)$
$x > \delta$	$x > \delta + \epsilon \wedge (\neg cross \vee z < 1)$

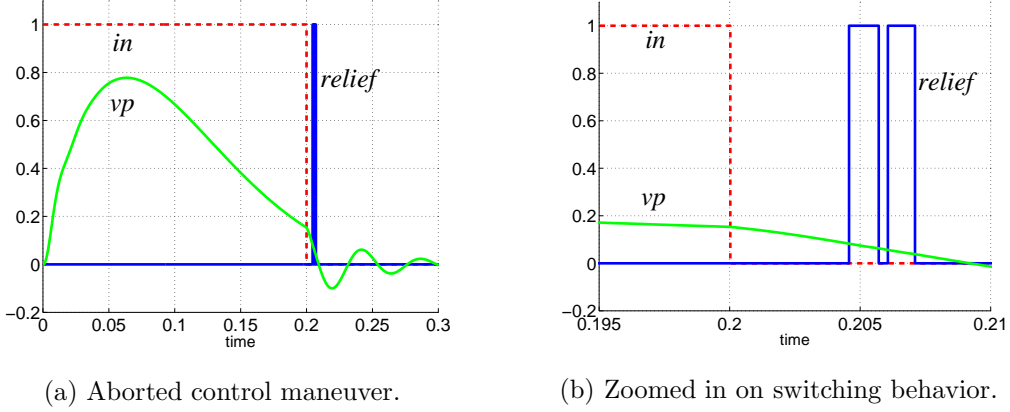


Figure 7: Simulation of the model in Fig. 6.

multiple events) in the ΔT interval occurs. If an event is detected, the stepsize is reduced from ΔT to δt_m , where δt_m is computed based on whether an event occurs, $\sigma = 1$, or not, $\sigma = 0$, in the interval δt_i as follows

$$\begin{aligned} \delta t_{i+1} &= \delta t_i + \Delta t_i(1 - \sigma) \\ \Delta t_{i+1} &= \frac{1}{2}\Delta t_i \end{aligned} \quad (3)$$

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 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 dependencies 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 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 first find 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 2.2.1³ are adopted by the signal port. In case of *a priori* switching conditions, the values are only adopted after the system state is updated and so discontinuous changes 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 0 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 0, which differs from behavior 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*

³If 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 an $\xi \ll \Delta T$.

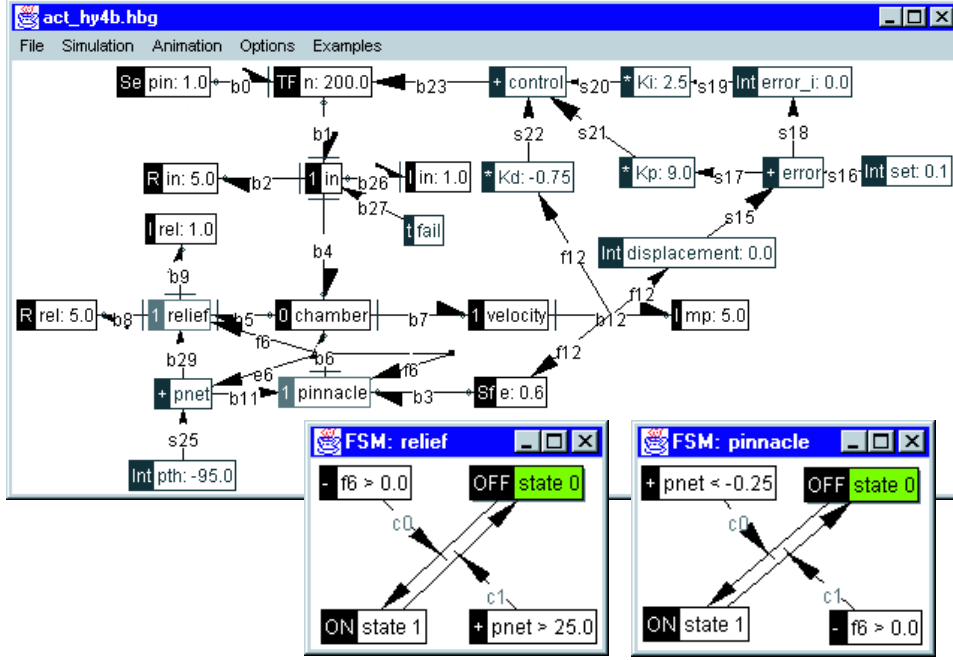


Figure 8: Actuator with explicit state jump.

mode [5, 21]. 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 a significant change in elevator velocity before the relief valve opens, see Fig. 7(b). 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, $v_p = \epsilon v_p^-$, where v_p^- is the value immediately before switching started. In Fig. 8 this is implemented by the Sf element with $\epsilon = 0.6$.⁴

The algebraic equation is activated using *a posteriori* values and deactivated using *a priori* values. The relief valve is opened based on *a priori* values to ensure the computed velocity change is effected. The stuttering behavior shown in Fig. 7 now occurs instantaneously at the same point in time, because the oil parameters are not present anymore. A sequence of activations of the algebraic restitution constraint and opening of the relief valve reduces piston velocity before it can be safely set to 0, shown in Fig. 9 by the data points with decreasing velocity at the switching time. Note that the mode where both valves are closed takes a mythical incarnation, otherwise, the velocity transferred to the

⁴Note the unique labels of connections and that causal conflicts between sources and junctions that are *off* are not terminal.

mode where *pinnacle* is *on* is 0, and no stuttering behavior takes place.

Also note that in Fig. 9(b) 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.

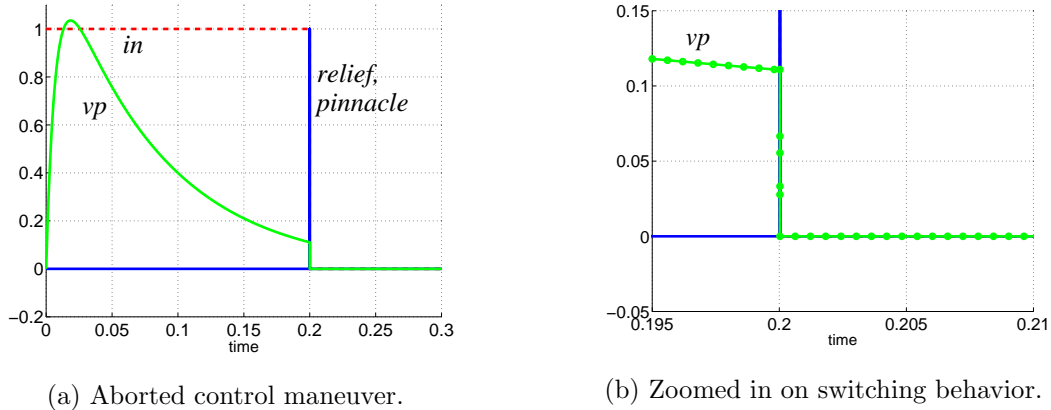


Figure 9: Simulation of the model in Fig. 8.

3.2.4 Impulse Comparison

In the model of the hydraulic cylinder in Fig. 8, when both valves are closed and the piston has nonzero velocity, a pressure spike occurs that takes the form of a Dirac pulse and has infinite magnitude. Therefore, the threshold pressure is always exceeded, regardless the piston velocity. In the more detailed model whether the pressure threshold is exceeded depends on the piston velocity [17]. 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 workspace, 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 behavior threshold and the threshold value in the signal port for the impulse area comparison. So, if the instantaneous change in velocity is less than -0.25 , the relief valve opens, and when during continuous behavior the pressure falls below the combined threshold value of -95.25 , the relief valve opens as well.

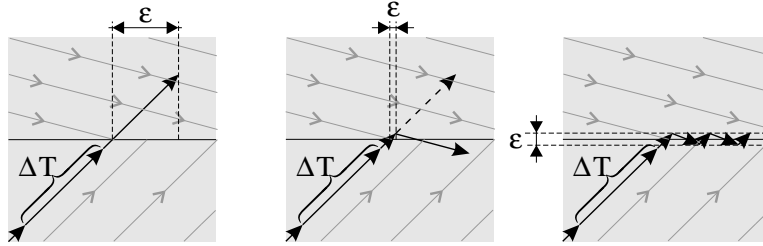


Figure 10: Chattering behavior.

3.2.5 Further Issues

Other phenomena in hybrid dynamic systems behavior that at present cannot be handled by HYBRSIM are (i) chattering and (ii) aborted projections.

Chattering In hybrid dynamic system behavior, 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.⁵ 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 behavior point towards the surface in both modes, see 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 [22].

Aborted Projections Dependency 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, e.g., a bisectional algorithm.

4 Export Filters

A hybrid bond graph model can be exported as an explicit ODE or implicit DAE in Java or C/C++ to generate model behaviors 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

⁵Note that this differs from an immediate switch back, in which case no stable discrete state exists.

execution order of the bond graph model is established, provided 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 the model in Fig. 3, this method is

```
void Zi::iterate() {
    double I2n = I1.f;
    I1.f = (- I2.I * (I2n - I2.f)) / I1.I;
    I2.f = I2n - I2.f;
}
```

and iteration is conform Table 3, with the convergence factor being set by the calling method.

However, each explicit solution is mode dependent because the change of 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 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. Though this means that the system of equations becomes more difficult to handle, i.e., it is of a higher *index* [23], 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:

$$0 = L \sum_i p_i f_i + (1 - L)e \quad (4)$$

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

$$0 = e_i - e \quad (5)$$

are added for each power port, i . In case $L = 1$, the standard equations for a 0-junction in implicit form are active. When $L = 0$, Eq. (4) results in $e = 0$, and combined with

Eq. (5) the efforts on all ports are 0. 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

$$\begin{aligned}
E.e &= E.in; \\
f.L &= (f.state == True) ? 1.0 : 0.0; \\
I1.res &= -I1.f + f.f; \\
I1.e &= I1.derf * I1.I; \\
R.f &= f.f; \\
R.e &= R.f * R.R; \\
I2.res &= -I2.f + f.f; \\
I2.e &= I2.derf * I2.I; \\
f.res &= (- E.e + I1.e + R.e + I2.e) * f.L + (f.L - 1) * f.f;
\end{aligned}$$

with

$$u = \begin{bmatrix} E.in \end{bmatrix}, x = \begin{bmatrix} I1.f \\ I2.f \\ f.f \end{bmatrix}, f = \begin{bmatrix} I1.res \\ I2.res \\ f.res \end{bmatrix}. \quad (6)$$

and *derf* the time derivative of the variable f . The required initial values can be computed by (i) using a routine explicitly generated by HYBRSIM, but that suffers from the combinatorial restriction (the computations differ per mode), and (ii) using a more general (and computationally more expensive) decomposition method that relies on the Weierstrass normal form [24].

5 Conclusions

Bond graphs are a powerful formalism to model the continuous behavior of physical systems in different domains. In many cases the dynamics contain nonlinearities 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 0 value source.

HYBRSIM is a hybrid bond graph modeling 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 runtime causality changes, including derivative causality, performs physically consistent (re-)initialization, and supports two types of event iteration because of dynamic coupling. Continuous behavior 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).

6 Acknowledgement

The graphical appearance of the HYBRSIM elements is based on a bond graph simulator by Róbert Bajzát.⁶

References

- [1] M. Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems*. PhD dissertation, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1994.
- [2] F.E. Cellier, H. Elmqvist, and M. Otter. Modelling from physical principles. In W.S. Levine, editor, *The Control Handbook*, pages 99–107. CRC Press, Boca Raton, FL, 1996.
- [3] Pawel Bujakiewicz. *Maximum weighted matching for high index differential algebraic equations*. PhD dissertation, TU Delft, Delft, Netherlands, 1994. ISBN 90-9007240-3.
- [4] Linda R. Petzold. A description of DASSL: A differential/algebraic system solver. Technical Report SAND82-8637, Sandia National Laboratories, Livermore, California, 1982.
- [5] Pieter J. Mosterman. *Hybrid Dynamic Systems: A hybrid bond graph modeling paradigm and its application in diagnosis*. PhD dissertation, Vanderbilt University, 1997.
- [6] Pieter J. Mosterman. An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569, pages 164–177. Lecture Notes in Computer Science; Springer-Verlag, March 1999.
- [7] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *Systems Dynamics: A Unified Approach*. John Wiley and Sons, New York, 2 edition, 1990.
- [8] Henry M. Paynter. *Analysis and Design of Engineering Systems*. The M.I.T. Press, Cambridge, Massachusetts, 1961.
- [9] Gottfried Falk and Wolfgang Ruppel. *Energie und Entropie: Eine Einführung in die Thermodynamik*. Springer-Verlag, Berlin, Heidelberg, New York, 1976. ISBN 3-540-07814-2.
- [10] Róbert Bajzát. Building geometric information into and getting out of mechanical systems described by bond graphs. In *MicroCAD'96*, University of Miskolc, 1996.
- [11] Pieter J. Mosterman and Gautam Biswas. A Java Implementation of an Environment for Hybrid Modeling and Simulation of Physical Systems. In *ICBGM99*, pages 157–162, San Francisco, January 1999.
- [12] Johannes van Dijk. *On the role of bond graph causality in modelling mechatronic systems*. PhD dissertation, University of Twente, CIP-Gegevens Koninklijke Bibliotheek, Den Haag, The Netherlands, 1994. ISBN 90-9006903-8.
- [13] Pieter J. Mosterman and Gautam Biswas. Formal Specifications from Hybrid Bond Graph Models. In *Qualitative Reasoning Workshop*, pages 131–142, Cortona, Italy, June 1997.
- [14] MATLAB. *The Language of Technical Computing*. The MathWorks, May 1997.
- [15] G. Dauphin-Tanguy, P. Borne, and M. Lebrun. Order reduction of multi-time scale systems using bond graphs, the reciprocal system and the singular perturbation method. *Journal of the Franklin Institute*, 319(1/2):157–171, 1985.
- [16] Willem Minten, Sven Vranckx, Bart De Moor, and Joost Vandewalle. Bondlab, a matlab based gui for bond graph modeling. *Journal A*, 38(3):11–15, 1997. Special CACSD issue.

⁶<http://gold.uni-miskolc.hu/~iitbajzi/bond/>

- [17] Pieter J. Mosterman and Gautam Biswas. Towards Procedures for Systematically Deriving Hybrid Models of Complex Systems. In Nancy Lynch and Bruce Krogh, editors, *Hybrid Systems: Computation and Control*, pages 324–337, 2000. Lecture Notes in Computer Science.
- [18] Pieter J. Mosterman and Gautam Biswas. A theory of discontinuities in dynamic physical systems. *Journal of the Franklin Institute*, 335B(3):401–439, January 1998.
- [19] John G. Kassakian, Martin F. Schlecht, and George C. Verghese. *Principles of Power Electronics*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1991. ISBN 0-201-09689-7.
- [20] Jan-Erik Strömberg, Jan Top, and Ulf Söderman. Variable causality in bond graphs caused by discrete effects. In *Proceedings of the International Conference on Bond Graph Modeling*, pages 115–119, San Diego, California, 1993.
- [21] T. Nishida and S. Doshita. Reasoning about discontinuous change. In *Proceedings AAAI-87*, pages 643–648, Seattle, Washington, 1987.
- [22] Pieter J. Mosterman, Feng Zhao, and Gautam Biswas. Sliding mode model semantics and simulation for hybrid systems. In Panos Antsaklis, Wolf Kohn, Michael Lemmon, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems V*, pages 218–237. Springer-Verlag, 1999. Lecture Notes in Computer Science.
- [23] C. W. Gear. Differential-algebraic equation index transformations. *SIAM Journal on Scientific and Statistical Computing*, 9(1):39–47, 1988.
- [24] Pieter J. Mosterman. Implicit Modeling and Simulation of Discontinuities in Physical System Models. In S. Engell, S. Kowalewski, and J. Zaytoon, editors, *The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, pages 35–40, 2000. invited paper.