An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages

Pieter J. Mosterman*

Institute of Robotics and System Dynamics
DLR Oberpfaffenhofen
P.O. Box 1116
D-82230 Wessling
Germany
FAX: +49 8153 28 2434
Pieter.J.Mosterman@dlr.de
http://www.op.dlr.de/~pjm

Abstract. Continuous system dynamics can be described by, possibly large, systems of differential equations. These can be either ordinary differential equations (ODEs) or contain algebraic constraints as well to form differential and algebraic equations (DAEs). Complex systems, such as aircraft, often operate in different *modes* of continuous operation and when mode changes occur, the continuous dynamics change abruptly. Even small physical components may operate in different modes, e.g., a diode can operate as a short or open circuit, requiring abrupt discrete changes in the system of equations.

Hybrid systems combine the continuous behavior evolution specified by differential equations with discontinuous changes specified by discrete event switching logic. Numerical simulation of continuous behavior and of discrete behavior is well understood. However, to facilitate simulation of mixed continuous/discrete systems a number of specific hybrid simulation issues must be addressed. This paper presents an overview of phenomena reported in previously published literature, that emerge in simulation of hybrid systems. They can be classified as (i) event handling, (ii) run-time equation processing, (iii) discontinuous state changes, (iv) event iteration, (v) chattering, and (vi) comparing Dirac pulses.

Based on these phenomena, numerical simulation requires the implementation of specific hybrid simulation features. An evaluation of existing simulation packages with respect to these features is presented. No assessment of the quality of particular implementations is given. The evaluation shows that some of the hybrid simulation features are incorporated in a number of simulation packages, whereas a number of other features are only implemented by a few or none. The aim of this paper is to categorize issues in hybrid simulation to benchmark the quality of simulation packages, and to refer to possible methods of implementation.

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

Extended Abstract

Simulation of pure continuous and pure discrete systems is well-understood. Ordinary differential equations (ODEs) or differential and algebraic equations (DAE) are a common representation for continuous systems from which numerical simulation algorithms generate behaviors. Variable step size approaches may be applied to ensure the numerical grid is sufficiently dense with respect to some error measure given the dynamic behavior of the system. Discrete simulation is often based on the particular description formalism, such as Petri nets [21] and finite state machines [12]. Typical is the use of random distribution functions to model, e.g., queue processing and to facilitate discrete phenomena such as nondeterminism and parallelism.

Recently, there is a growing interest in *hybrid systems*, i.e., systems with mixed continuous/discrete behavior. This interest is driven by (i) the increasing need for comprehensive analysis of systems where discrete controllers operate on a continuous process, and (ii) efficient handling of otherwise stiff continuous equations. In hybrid simulation, typically, the continuous model part generates discrete events when continuous signal variables cross threshold values. These discrete events may affect continuous behavior evolution by changing active model components and discontinuously changing the continuous state variables.

Though individually the continuous and discrete formalisms can be treated well, their interaction causes a number of unique problems in simulation. This paper discusses and illustrates these issues and refers to possible solutions. It identifies which of the features specific to hybrid simulation are incorporated by a number of simulation packages. It does not evaluate robustness or quality of the implemented solutions. For additional information refer to [13].

Generating the Simulation Model

This section describes how models are processed to achieve a DAE form.

1. Complex dynamic system models are composed of declarative submodels specified by noncausal equations [6]. After the complete system of equations is *compiled* a *sorting* procedure assigns computational causality. In matrix form, this corresponds to a lower triangular form. Circular dependencies between variables cause blocks of dependent equations (which can be solved symbolically or numerically) to be kept together, resulting in a block lower triangular (BLT) matrix structure of the sorted equations.
2. After sorting, the complexity of the simulation problem, indicated by the *index*, needs to be sufficiently low [5]. Higher-index problems contain algebraic constraints on time derivative variables, and the system of equations needs to be *solved* to arrive at a lower index, e.g., by Pantelides' algorithm [22] and the dummy derivative approach [15].
3. Next, consistent initial values of the state variables need to be calculated from user specified values. For example, to start simulation from steady state, all time derivative values can be set to 0. The corresponding variable values are then computed based on these conditions.

4. Once the consistent initial values of the system of equations are computed, a numerical solver evolves the system behavior over time. Numerical solvers vary from an explicit fixed step Euler to complex implicit integration schemes with variable step size and error control such as DASSL.

Hybrid Simulation Phenomena

This section presents a number of benchmark examples.

Newton's Cradle Consider the three colliding bodies in Fig. 1 where m_1 has an initial velocity, $v_1 = v$. Initially, m_1 moves towards m_2 with constant velocity while m_2 and m_3 are at rest:

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \\ \dot{v}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \tag{1}$$

with algebraic equations

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

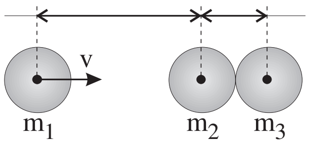

Fig. 1. A sequence of collisions.

Upon collision, $(x_1 \geq x_2) \wedge (v_1 > v_2)$, momentum is instantaneously transferred to m_2. The first condition specifies that in case of point masses there is contact between the two bodies and the second condition that there is a collision. For m_2 and m_3 this condition is not satisfied because $v_2 = v_3$, and, therefore, no collision occurs. Note that when $x_1 \geq x_2$ a state event is generated that needs to be detected and located to improve precision.

The change of the velocities v_i immediately before collision to their values immediately after collision, v_i^+, is governed by Newton's collision rule

$$v_2^+ - v_1^+ = -\epsilon(v_1 - v_2), \tag{3}$$

The values of v_1 and v_2 are known as their final value when $x_1 \geq x_2$ and continuous behavior was halted. However, both v_1^+ and v_2^+ are unknown and cannot be solved with one equation. In case of the colliding bodies, upon collision the forces $F_1 = F_2$, which can be added to the system of equations by replacing Eq. (2) with

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

Using Eq. (1) to solve the equation in the top row yields $m_1\dot{v}_1 = m_2\dot{v}_2$ and this can be integrated to $m_1(v_1^+ - v_1) = m_2(v_2^+ - v_2)$. This embodies the physical conservation of momentum constraint. Combined with Eq. (3) it can be uniquely solved for v_1^+ and v_2^+.

When the new velocities are updated $(v_i \leftarrow v_i^+)$, $v_2 > v_3$ and an immediate further configuration change occurs that models the collision between m_2 and m_3. Again Newton's collision rule and conservation of momentum applies and the new velocities of m_2 and m_3 are computed. In case $m_1 = m_2 = m_3$, after momentum is transferred to m_3, no further configuration changes occur and m_3 starts to move continuously in time.

The Falling Rod Consider the rigid rod sliding on a rough surface in Fig. 2. In case of Coulomb friction, the friction force F_f depends on the normal force F_N by a constant coefficient μ, i.e., $F_f = \mu|F_N|$ [14], active in the direction opposite to $v_{A,x}$, the velocity of the contact point at the surface. F_y is the kinetic force exerted by the center of mass in the vertical direction. Combined with the gravitational force, F_g, this yields the normal force $F_N = F_y - F_g$. Velocity v_x is the horizontal velocity of the center of mass, M, v_y the vertical velocity, and ω the angular velocity. The system has three inertial, energy storing, components, *viz.*, the linear inertias m_x and m_y and the rotational inertia J.

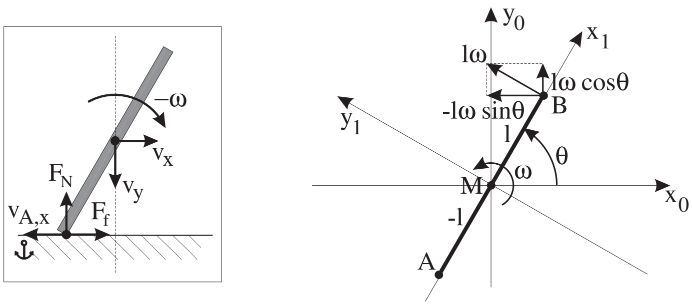

Fig. 2. Coulomb friction.

If initially the rod is falling freely towards the floor,

$$\begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_\omega \end{bmatrix} \tag{5}$$

with algebraic constraints

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_\omega \end{bmatrix} = \begin{bmatrix} 0 \\ F_g \\ 0 \end{bmatrix}. \tag{6}$$

Upon collision, Fig. 2 shows that the linear velocities v_x and v_y are constrained to move with respect to ω according to

$$\begin{cases} v_x = -l\omega sin\theta \\ v_y = l\omega cos\theta \end{cases} \tag{7}$$

Furthermore, the forces in the x and y direction relate to the torque F_ω as

$$-lsin\theta F_x + lcos\theta F_y + F_\omega - lcos\theta F_g = 0. \tag{8}$$

Therefore, upon collision Eq. (6) needs to be replaced by

$$\begin{bmatrix} 1 & 0 & lsin\theta & 0 & 0 & 0 \\ 0 & 1 & -lcos\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & -lsin\theta & lcos\theta & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \\ F_x \\ F_y \\ F_\omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ lcos\theta F_g \end{bmatrix} \tag{9}$$

These constraints require reinitialization of the state vector, i.e., the linear and angular velocities, and because of the time-derivative nature of forces discontinuous velocity changes may cause an impulsive force $F_y = m_y \dot{v}_y - F_g$. Here, $\dot{v}_y$ is a Dirac pulse, δ, with area $v_y^+ - v_y$, $\dot{v}_y = \delta[v_y^+ - v_y]$. F_x is also of an impulsive nature, $\dot{v}_x = \delta[v_x^+ - v_x]$. Since $F_N = F_y - F_g$, the condition for sliding, $|F_x| > \mu F_N$, becomes $|m_x \dot{v}_x| > \mu(m_y \dot{v}_y - F_g)$ which requires evaluating $|m_x \delta[v_x^+ - v_x]| > \mu(\delta[v_y^+ - v_y] - F_g)$. By numerically approximating the infinite magnitude of the Dirac pulses, F_g may inadvertently affect the comparison.

This is solved by comparing Dirac pulse areas. So, if upon collision $|v_y^+ - v_y| > \mu(v_y^+ - v_y)$, the rod starts to slide and another set of algebraic constraints becomes active

$$\begin{bmatrix} 0 & 0 & 0 & 1 & \mu & 0 \\ 0 & 1 & -lcos\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & -lsin\theta & lcos\theta & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \\ F_x \\ F_y \\ F_\omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ lcos\theta F_g \end{bmatrix} \tag{10}$$

Again, consistent initial values have to be computed from the current state variable values before continuous integration can resume. Note that in this case v_x is not constrained, and, therefore, is not required to change discontinuously. To generate physically meaningful behavior, the new state variable values should not be calculated from the values that were obtained by solving the initialization problem in the stuck mode that is active upon collision, because this mode was instantaneously departed. Instead, the values should be derived from the final values in the last mode of continuous behavior, i.e., where the rod was falling freely [16].

An Evaporator Vessel In a fast breeder reactor, an evaporator vessel stores hot sodium and warms water that flows through a helical coil inside the vessel. As a safety mechanism, an overflow that connects to the sodium sump may become active when a specific fluid level is reached, see Fig. 3.

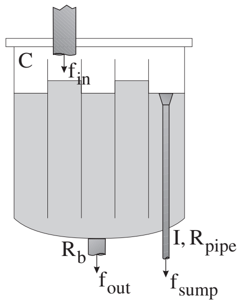

Fig. 3. An evaporator vessel.

A model of this system may consist of the vessel with capacity, C, and outflow resistance, R_b. The overflow may be modeled by its flow resistance, R_{pipe}, and fluid inertia, I. When the fluid level is below the overflow level,

$$\begin{bmatrix} I & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} \dot{f} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} -R_{pipe} & 0 \\ 0 & -\frac{1}{R_b} \end{bmatrix} \begin{bmatrix} f \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ f_{in} \end{bmatrix}, \tag{11}$$

a steady state is achieved such that the inflow, f_{in}, equals the outflow f_{out}. If R_b requires a liquid level higher than the overflow level, this mechanism becomes active,

$$\begin{bmatrix} I & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} \dot{f} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} -R_{pipe} & 1 \\ -1 & -\frac{1}{R_b} \end{bmatrix} \begin{bmatrix} f \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ f_{in} \end{bmatrix}, \tag{12}$$

and another steady state level is achieved such that $f_{in} = f_{out} + f_{sump}$. For specific parameters (e.g., $R_b = 1, R_{pipe} = 0.5, I = 0.5, C = 15, f_{in} = 0.25, \Delta T = 0.025$), this may cause the liquid level to fall below the overflow level and the mechanism becomes inactive but now the liquid level rises and the overflow becomes active again after an infinitesimal short period of time, causing the system to *chatter* between modes.

In general, simulation across discontinuities may cause large errors for a fixed integration step, see Fig. 4(a). Continuous time integration may be executed until within a small tolerance of the discontinuity, see Fig. 4(b). However, in case of chattering, this causes a repeated reduction of the integration step size to its minimal value, Fig. 4(c), and simulation times become excessively long.

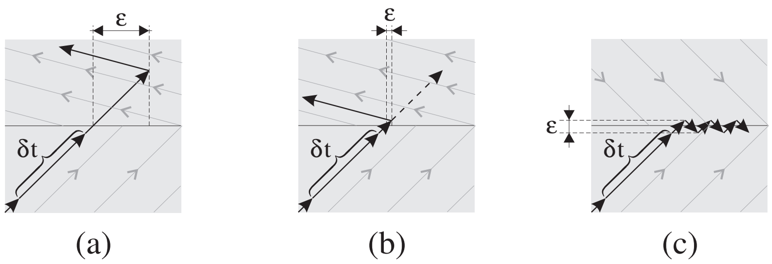

Fig. 4. Chattering.

Summary The following hybrid simulation phenomena can be identified.

- **time events:** Events are generated at predetermined times which can be treated efficiently [7].
- **state events:** If events occur because of system variables crossing threshold values, the time of their occurrence is not known *a priori*. In this situation [2,7,23],
  - the event needs to be **detected**, and
  - its time of occurrence needs to be **located**.
- **simulation model:** The system of equations may change.
  - Blocks of sorted and solved equation may simply appear or disappear (e.g., a vehicle entering and leaving a highway), and, therefore, can be dynamically **added/removed**.
  - In some cases equations can be replaced by others, changing computational causality, and the system of equations may have to be **sorted** again.
  - In other cases, algebraic constraints between state variables may become active and the system of equations needs to be **solved** again (e.g., the rod making contact with the floor).
- **reinitialization:** There may be a discontinuous change in state variable values.
  - This change may be **explicitly** specified by the user by a new initial state equation (e.g., Eq. (3) of the colliding bodies).
  - The system of equations may have to be **integrated** to derive physically consistent initial values for a new mode. This ensures conservation of the thermodynamic extensity holds [17].
- **event iteration:** When an event occurs, new system variable values may immediately trigger a further event. Two types of event iteration exist [18],
  - the state vector is **invariant** across the entire iteration (e.g., the falling rod that immediately starts to slide), and
  - the state vector is **updated** after each iteration step (e.g., the sequence of colliding bodies).

- **chattering:** If the system moves back and forth between modes the system starts to chatter. Root finding to locate the exact time of occurrence of the event causes continuous integration to become excessively slow. An equivalence relation eliminates the fast chattering motion, but preserve the dynamics of the slow motion along the chattering surface [20].
- **Dirac pulses:** Discontinuous changes in continuous variables may cause Dirac pulses to occur. If their magnitudes are numerically approximated, comparison may be affected by non-Dirac type variables (e.g., the sliding condition for the falling rod). To ensure numerically precise treatment, Dirac pulse values should be distinguished from non-Dirac pulse values and evaluation of Dirac pulses can be based on their areas [16].

Packages Evaluation and Conclusions

The following software packages were investigated with respect to their support of the described hybrid features, see Table 1.

- χ is a simulation environment initially developed for modeling and simulation of manufacturing plants at the University of Eindhoven [10].
- *ABACUSS* is a derivative work of the gPROMS software [1]. It is developed at the Massachusetts Institute of Technology.
- *BaSiP* is developed at the University of Dortmund for simulation of recipe-driven production in complex multi-purpose batch plants [26].
- *DOORS* is a prototype distributed real-time simulator for mechatronic design developed at the University of Magdeburg [11].
- *Dymola* provides a powerful object oriented modeling and simulation environment for education and the professional engineer [9].
- *gPROMS* was initially developed at Imperial College, London for process modeling, simulation and optimization [3]. It is now commercially available.
- HyBrSim is an experimental hybrid bond graph modeling and simulation tool based on physical principles developed at the DLR Oberpfaffenhofen [19].
- *Omola* is developed at the Lund Institute of Technology for modeling and simulation of continuous time and discrete event dynamic systems [2].
- Shift is a programming language for describing dynamic networks of hybrid automata, developed at the University of California, Berkeley [8].
- *SIMULINK* is a block diagram based modeling and simulation environment of the MathWorks [24].
- *Smile* is a simulator for energy systems of GMD FIRST, Berlin [25].
- *20-SIM* ("Twente Sim") is a modeling and simulation program developed at the University of Twente [4].

Table 1 shows that both time and state events are typically handled by these packages, though the implementation may vary [23]. Also, they facilitate adding and removing equations that do not change causality. Furthermore, the use of noncausal numerical simulation algorithms supports conditional equations that do not affect the DAE structure, i.e., those that would only require a new sorting stage in case explicit numerical integration routines are used.

Table 1. Tools evaluation.

		χ	ABACUSS	BaSiP	DOORS	Dymola	gPROMS	HyBrSim	Omola	SHIFT	SIMULINK	Smile	20sim
time events		✓	✓		✓	✓	✓		✓	✓	✓	✓	✓
state events	detection	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	location	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
simulation model	add/remove	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	re-sort	✓	✓		✓	✓	✓	✓	✓		✓	✓	
	re-solve							✓					
reinitialization	explicit	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	integration							✓					
event iteration	invariant							✓					
	update	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
chattering													
Dirac pulses													

Solving the system of equations during run-time is more complicated. This may require index-reduction and new initial values of the state variables may have to be calculated. This has been implemented in HyBrSim for index 2 DAEs by built-in physical conservation constraints. However, HyBrSim is an interpreted simulator, and, therefore, not as efficient. ABACUSS provides some support for run-time solving, but this is still under development.

Event iteration is a crucial part of hybrid simulation. Most packages only implement event iteration that updates the state vector values at every step. This allows simulation of hybrid systems that only contain time scale abstraction [18]. Further hybrid simulation issues such as chattering and comparing Dirac pulses is at present not addressed in any general purpose simulator.

Acknowledgement

Thanks to the following people for their discussion and providing their expertise: Jan F. Broenink, Georgina Fábián, Martin Fritz, Martin Otter, Clemens Klein-Robbenhaar, Olaf Stursberg, Hubertus Tummescheit, and Andreas Wolf.

References

1. ABACUSS. http://yoric.mit.edu/abacuss/abacuss.html, 1995. MIT.
2. M. Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems.* PhD diss., Dept. of Automatic Control, Lund Inst. of Technology, Lund, Sweden, 1994.
3. P. I. Barton. *The Modelling and Simulation of Combined Discrete/Continuous Processes.* PhD diss., University of London, 1992.
4. J.F. Broenink. Modelling, simulation and analysis with 20-sim. *Journal A*, 38(3):22–25, Jan. 1998. Special CACSD issue.
5. P. Bujakiewicz. *Maximum weighted matching for high index differential algebraic equations.* PhD diss., TU Delft, Netherlands, 1994.

6. F.E. Cellier, H. Elmqvist, and M. Otter. Modelling from physical principles. In W.S. Levine, editor, *The Control Handbook*, pp. 99–107. CRC Press, Boca Raton, FL, 1991.
7. F.E. Cellier. *Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools.* PhD diss., ETH, Zurich, Switzerland, 1979.
8. A. Deshpande, A. Göllü, and L. Semenzato. *SHIFT Programming Language and Run-Time System for Dynamic Networks of Hybrid Automata.* California PATH, UC Berkeley.
9. H. Elmqvist, D. Brück, and M. Otter. *Dymola — User's Manual.* Dynasim AB, Research Park Ideon, Lund, 1996.
10. G. Fábián, D. A. van Beek, and J. E. Rooda. Integration of the discrete and the continuous behaviour in the hybrid Chi simulator. In *Proc. 1998 European Simulation Multiconference*, pp. 252–257, Manchester, 1998.
11. R. Kasper and W. Koch. Object-oriented behavioural modelling of mechatronic systems. In 3^{rd} *Conf. on Mechatronics and Robotics '95*, Paderborn, Germany, Oct. 1995.
12. Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, New York, 1978.
13. S. Kowalewski, M. Fritz, H. Graf, J. Preußig, S. Simon, O. Stursberg, and H. Treseler. A Case Study in Tool-Aided Analysis of Discretely Controlled Continuous Systems: the Two Tanks Problem. In *Fifth Inl. Conf. on Hybrid Systems*, Notre Dame, Indiana, Sep. 1997.
14. P. Lötstedt. Coulomb friction in two-dimensional rigid body systems. *Z. angew. Math. u. Mech.*, 61:605–615, 1981.
15. S.E. Mattsson and G. Söderlind. A new technique for solving high-index differential-algebraic equations. In *Proc. of the 1992 Symp. on Computer-Aided Control System Design*, pp. 218–224, Napa, California, March 1992.
16. P.J. Mosterman. *Hybrid Dynamic Systems: A hybrid bond graph modeling paradigm and its application in diagnosis.* PhD diss., Vanderbilt University, 1997.
17. P.J. Mosterman. State Space Projection onto Linear DAE Manifolds Using Conservation Principles. Tech. Report #R262-98, Inst. of Robotics and System Dynamics, DLR Oberpfaffenhofen, P.O. Box 1116, D-82230 Wessling, Germany, 1998.
18. P.J. Mosterman and G. Biswas. Principles for Modeling, Verification, and Simulation of Hybrid Dynamic Systems. In *Fifth Intl. Conf. on Hybrid Systems*, pp. 21–27, Notre Dame, Indiana, Sep. 1997.
19. P.J. Mosterman, G. Biswas, and M. Otter. Simulation of Discontinuities in Physical System Models Based on Conservation Principles. In *SCS Summer Simulation Conf.*, pp. 320–325, Reno, CA, July 1998.
20. P.J. Mosterman, F. Zhao, and G. Biswas. Sliding mode model semantics and simulation for hybrid systems. In *Hybrid Systems V.* Springer-Verlag, 1998. Lecture Notes in Computer Science.
21. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
22. C.C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific and Statistical Computing*, 9(2):213–231, March 1988.
23. T. Park and P.I. Barton. State event location in differential-algebraic models. *ACM Transactions on Modeling and Computer Simulation*, 6(2):137–165, April 1996.
24. SIMULINK. *Dynamic System Simulation for Matlab.* The MathWorks, Jan. 1997.
25. Smile. http://gargleblaster.cs.tu-berlin.de/~smile. TU Berlin.
26. K. Wöllhaf, M. Fritz, C. Schulz, and S. Engell. BaSiP - Batch process simulation with dynamically reconfigured process dynamics. *Proc. of ESCAPE-6, Suppl. to Comp. & Chem. Engng*, 20(972):1281–1286, 1996.