Modeling Systems With Variable Algebraic Constraints for Explicit Integration Methods

Pieter J. Mosterman*, Peter Neumann, Carsten Preusche
Institute of Robotics and Mechatronics
DLR Oberpfaffenhofen, Germany
[Pieter.J.Mosterman|Peter.Neumann|Carsten.Preusche]@dlr.de

Abstract

Efficient models are not necessarily the most detailed ones. The purpose of a model is to solve a problem and it needs to be just detailed enough to achieve this. In many cases this may require simplifying models by removing nonlinear continuous behaviors by means of a piecewise linearization. As a consequence, the model operates in a number of different continuous modes where different equations describe system behavior. When models are simplified even further and fast continuous mode transition behaviors are removed, the dynamic coupling between state variables of interest may reduce to algebraic constraints, causing a reduction of degrees of freedom of the system when mode changes occur. To generate behaviors for such variable structure systems requires algebraic manipulations to derive the reduced order system. These algebraic manipulations may include differentiation of equations that is inefficient when performed during behavior generation. The alternative of pre-compilation is restricted to systems with few modes to avoid enumeration problems because of the combinatorial explosion. This paper presents a method to handle variable structure systems with varying algebraic constraints (i.e., run-time index changes) by means of explicit integration methods complemented by a projection in the impulse space that is consistent with the instantaneous dynamics of the vector field. The method is demonstrated by modeling and simulation of an AC induction motor.

Keywords: DAE initialization, DAE modeling, simulation, variable structure systems, run-time index changes, hybrid systems

1 INTRODUCTION

Continuous behavior of physical systems can be well described in terms of nonlinear ordinary differential equations (ODE). To achieve efficient models for analysis, the nonlinearities may be modeled as piecewise linear behaviors. For example, when in the electrical circuit in Fig. 1 the switch, Sw_1, is closed, the voltage v_{12} is connected to ground, and, therefore, at 0 V. The source voltage V_{cc} causes a voltage drop across each of the inductor/resistance series connections. This voltage drop builds up a flux, p, in the inductors until the corresponding current causes a voltage drop across each of the resistors equal to V_{cc} (note that this does not have to happen at the same time for each series connection branch).

When Sw_1 is opened, I_1 and I_2 become coupled and the fluxes of I_1 and I_2 become related to the same current. In a continuous model, the switch can be modeled by a nonlinear resistor that quickly changes its value from very small to very large when the switch opens. In a first approximation, this characteristic can be made piecewise linear by switching between modes with no resistance (the switch is closed), and with a large resistance (the switch is open). Immediately after the switch is opened, the mode with large resistance becomes active which induces a large voltage drop to quickly ensure $i_1 = i_2$, see Fig. 2(a). After this constraint is met, the fast switching dynamics have settled and the current flow through the large resistance becomes negligible. The overall model is of a mixed continuous and discrete nature, called a *hybrid model*, but state trajectories are still continuous. State derivatives may change discontinuously, though, and, therefore, these models are referred to as C^0 *hybrid models*, the 0^{th} derivative is continuous.

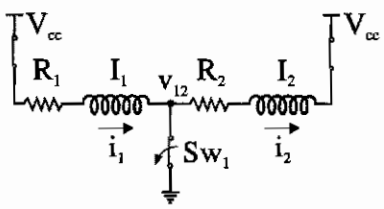

Figure 1: Physical system with two possible modes.

In many cases, the piecewise linearized behaviors are still too complex to handle efficiently, e.g., because of steep gradients in system behavior that require a small integration time step in numerical simulation. These gradients can be abstracted away, but their effect on gross system behavior needs to be maintained, *viz.*, the final flux values of the fast transient in Fig. 2(a) when the $i_1 = i_2$ constraint is first met. This is achieved by

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

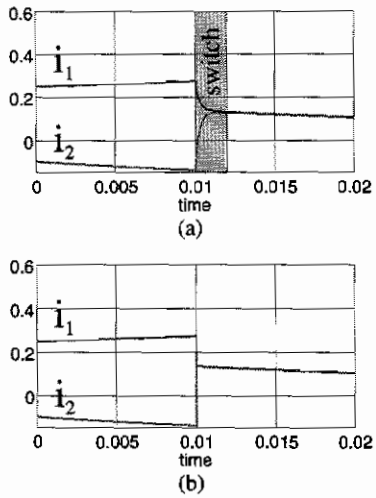

Figure 2: A model with (a) C^0 and (b) discontinuous state behavior.

modeling discontinuous changes, 'jumps', in the state variables, illustrated in Fig. 2(b). A consequence of the abstraction is that the corresponding models do not satisfy the C^0 criterion anymore but are of a general hybrid systems form.

Singular perturbation methods [KKO86] reduce the order of a system by abstracting fast dynamic behavior away. In case of the inductor circuit, when the switch is opened the system reduces from a second order ODE to a first order ODE with a specification of the discontinuous change in flux. ODE representations combined with some discrete modeling approach are utilized by many formalisms to model hybrid behavior [ACHH93, GJ95]. In these paradigms, each global discrete state, or system *mode*, has an associated ODE with varying degree of complexity associated with it.

Designing such models works well if ODE formulations for each mode can be analyzed exhaustively. However, in case of complex systems with many modes, the method breaks down quickly. For example, in Fig. 3 already four modes are present, just by cascading one inductor/switch combination. Cascading further inductor/switch combinations causes a combinatorial explosion as there are 2^n modes for n switches. It is clear that the enumeration approach fails in case of an AC induction motor with six or more such switches, the subject of this paper.

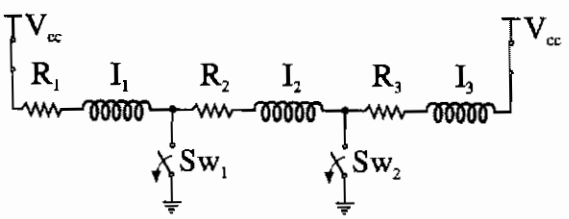

Figure 3: Physical system with four possible modes.

In an attempt to mitigate these problems, compositional modeling approaches try to exhaustively analyze model components that can be pieced together without further analysis [Mos97]. For C^0 hybrid systems, this approach works well. However, since discontinuous changes often are the aggregate of different interacting model components, compositionality becomes difficult to uphold when the C^0 constraint is relaxed [MB99b]. For the interacting inductors in Fig. 3, a decomposition in k sets of m interacting switches is not possible because all inductors may interact and cause discontinuous state changes that are the result of the aggregate behavior. In other words, all inductors belong to the same *causal area* [Mos97] and the C^0 constraint is violated.

This paper presents an approach to modeling variable structure systems where a set of state variables may be collapsed into one new state. The presented approach dynamically infers the new ODE behavior and possible discontinuous state changes, which allows the use of general purpose simulation tools such as MATLAB-SIMULINK [SIM97] to be used for simulation.

2 ANALYSIS

To introduce the critical problem in modeling the AC induction motor that is presented in Section 5, consider the electrical circuit in Fig. 1. The required fast dynamics to achieve $i_1 = i_2$ when Sw_1 is opened are caused by some switch resistance, R. If the switch resistance, R, is abstracted away from the model, the fluxes p_1 and p_2 that are related to the current by $p = iL$, with L the inductance, change instantaneously to achieve the final values of the fast continuous behavior. This requires to systematically find (i) the magnitude of the instantaneous state change such that in gross terms it is consistent with detailed continuous behavior, and (ii) the dynamic behavior of the reduced order system. To avoid the need for exhaustive analysis, a critical requirement is that this derivation can be automated and performed during run-time. This section discusses how to derive an explicit formulation of the discontinuous state change from the system of equations that govern continuous behavior in a mode.

2.1 The Model

First, the electrical circuit in Fig. 1 with an ideal switch is modeled as a system of equations with a differential equation and algebraic equation part. The model specification is graphically depicted in Fig. 4. The differential equation part represents the first order behavior of each series branch

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{v}_{12} \end{bmatrix} = \begin{bmatrix} -\frac{R_1}{L_1} & 0 & -1 \\ 0 & -\frac{R_2}{L_2} & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ v_{12} \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} V_{cc} \end{bmatrix}. \quad (1)$$

When Sw_1 is closed, this is complemented by an algebraic constraint, $v_{12} = 0$,

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{v}_{12} \\ p_1 \\ p_2 \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ V_{cc} \end{bmatrix}. \tag{2}$$

When Sw_1 is open, the algebraic constraint $i_1 = i_2$ replaces $v_{12} = 0$. In terms of the state variables p_1 and p_2 this becomes

$$\begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{L_1} & -\frac{1}{L_2} & 0 \end{bmatrix} \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{v}_{12} \\ p_1 \\ p_2 \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ V_{cc} \end{bmatrix}, \tag{3}$$

which establishes a manifold in phase space to which behavior is confined.

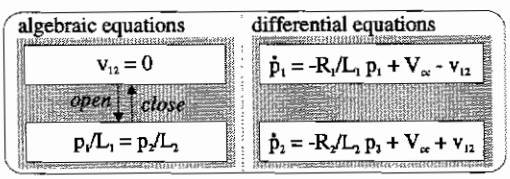

Figure 4: Differential and variable algebraic part.

2.1.1 Consistent Initial Conditions

A projection onto the manifold embodied by the algebraic constraint in Eq. (3) is derived. Adding the two rows in Eq. (1) to eliminate v_{12} gives $\dot{p}_1 + \dot{p}_2 = -\frac{R_1}{L_1} - \frac{R_2}{L_2}$. This can be integrated over an infinitesimal interval $[t_0^-, t_0]$ where only the time derivative terms contribute beeause of their impulsive behavior upon opening the switch [Mos98]. This yields $\int_{t_0^-}^{t_0} \dot{p}_1 + \dot{p}_2 dt = (p_1 - p_1^-) + (p_2 - p_2^-) = 0$ with $p_i = p_i(t_0)$ and $p_i^- = p_i(t_0^-)$. From Eq. (3) it follows that $\frac{p_1}{L_1} = \frac{p_2}{L_2}$ and substituting $p_2 = \frac{L_2}{L_1} p_1$ and $p_1 = \frac{L_1}{L_2} p_2$ results in the *projection equation*

$$\begin{aligned} p_1 &= \tfrac{L_1}{L_1+L_2}(p_1^- + p_2^-) \\ p_2 &= \tfrac{L_2}{L_1+L_2}(p_1^- + p_2^-) \end{aligned} \tag{4}$$

where the flux values immediately before the switch opened, the *a priori* values, are marked by a $-$ superscript. A closed form solution to derive the state projection for a class of equation systems is given in [Mos98].

2.1.2 The Redueed Order Dynamics

Dynamie behavior when Sw_1 is open is that of a first order system, and is derived from Eq. (1) by differentiating the algebraic constraint in Eq. (3) to express $\dot{p}_2$ in terms of $\dot{p}_1$. Next, the rows in Eq. (1) are added to eliminate v_{12}, which gives $\dot{p}_1 + \dot{p}_2 = -\frac{R_1}{L_1} p_1 - \frac{R_2}{L_2} p_2$ and after substituting $p_2 = \frac{L_2}{L_1} p_1$ and $\dot{p}_2 = \frac{L_2}{L_1} \dot{p}_1$, the first order ODE ean be derived

$$\dot{p}_1 = -\frac{R_1 + R_2}{L_1 + L_2} p_1. \tag{5}$$

Algorithms to derive the dynamic behavior on a manifold (the manifold dynamics) are well known [Pan88]. The mode switch between Sw_1 closed and Sw_1 open can now be modeled by the hybrid automata in Fig. 5.

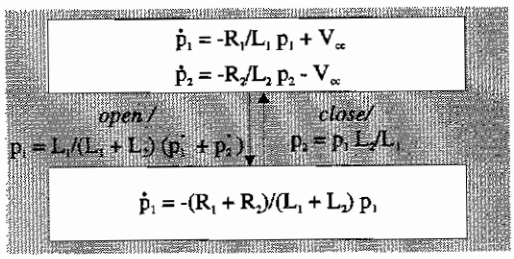

Figure 5: Hybrid automata for the diode circuit.

2.2 Automated Handling

In case the pre-enumeration approach is not feasible, the reduced order system model and its explicit state change have to be computed during run-time, *viz.*, when behavior generation infers a new mode. This would allow the electrical circuit in Fig. 1 to be modeled direetly by the hybrid automata in Fig. 4. When the $i_1 = i_2$ constraint becomes active, the simulation engine (i) derives the appropriate reduced order model, and (ii) derives the explicit state variable value changes. Currently, such sophisticated run-time model processing techniques are not generally available in simulation tools [Mos99]. The feasibility is demonstrated, e.g., by HYBRSIM [MB99a], where systems such as in Fig. 1 can be conveniently modeled by *hybrid bond graphs* [Mos97], representing the switch by a local finite state machine with variable algebraic constraints.

3 COMPOSITIONAL MODEL FRAGMENTS

To climinate the need for run-time equation processing, the state projeetion in Eq. (4) is stated locally in each model component by the general formula

$$p_i = L_i \frac{\sum_j p_j^-}{\sum_j L_j}, \ (\forall j)(p_j \in C) \tag{6}$$

Here C is the set of all states p_j that are coupled, i.e., algebraically related, with p_i. Based on this computation, the projection for each component only requires numerical knowledge of the total value of the coupled states, p_j, that are collapsed into one, p_i, and the parameters that determine the weighting, L_j. No algebraie knowledge of a model component's internal structure and algebraie manipulations are required to execute the state projection.

Figure 6 shows the resulting set of equations for the model components of the switch and inductor in Fig. 1. The projection equation models the explicit state variable jump when two or more inductors are coupled. To facilitate this, the inductor model component requires (i) the inductance of the coupled inductors, needed for the weighting calculations, and (ii) the flux of the coupled inductors, needed to compute the total value of the collapsed states. Therefore, these values are made available as model component output. It will be shown in Section 4 that this projection also serves to generate the reduced order (manifold) dynamics.

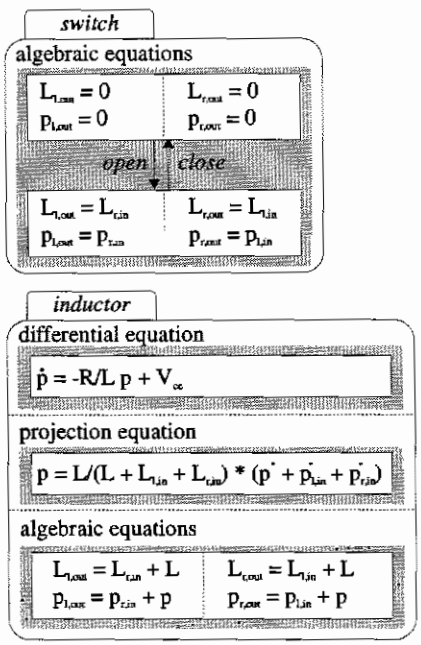

Figure 6: General equations for the model components.

The switch can now be modeled as a local finite state machine that either disconnects its left and right inductors, i.e., the switch is closed, or it enforces algebraic constraints, i.e., the switch is open. When there is no coupling between the inductors, the coupled flux and inductance are passed as 0 values. Substitution of $L_{l,in} = L_{r,in} = 0$ and $p_{l,in} = p_{r,in} = 0$ in the projection equation shows that $p = p^-$, and no jump occurs. When there is a coupling, the sum of all dependent inductances to the left is passed to the right and the sum of all dependent inductances to the right is passed to the left. Similarly, the total flux is passed to connected inductors to compute the projection.

The use of this extended inductor model is illustrated in Fig. 7. This model is now amenable for direct implementation in widely proliferated simulation packages such as MATLAB-SIMULINK. Moreover, the behavior generation technique can be applied without explicit knowledge of the internal equation structure of components (no algebraic manipulations are necessary), which supports component based simulation. The model clearly shows that it can be easily extended to facilitate any number of inductors without additional modeling effort. The variable structure that causes run-time mode changes is not limited to exhaustive analysis anymore, but the behavior in the actual mode of operation is dynamically generated during run-time.

4 THE SIMULATION ALGORITHM

Numerical simulation of continuous behavior occurs with discrete time steps, the integration time step. At a point in time, t_k, the gradient of continuous behavior is calculated, and used to compute new values at the next integration time t_{k+1}. Abbreviating $x(t_k)$ as x_k, this results in the general numerical integration approach $x_{k+1} = \Delta T \dot{x}_k + x_k$. The time step, $\Delta T = t_{k+1} - t_k$, between two integration points, k and $k+1$, is not necessarily fixed and may be adjusted based on the complexity of the continuous dynamics. To further improve accuracy, this basic scheme can be modified, e.g., by applying sophisticated interpolation methods and previous integration points, x_{k-i}.

After a mode change, the projection equation of the model derived in Section 3 dynamically computes the new initial values for the state variables on the manifold of behavior. However, because the continuous dynamics are of higher order, future behavior is not ensured to be confined to this manifold. This is solved by using the integration as an intermediate step to calculate the preliminary point, x^-_{k+1}. After each time step the computed values x^-_{k+1} are projected onto the manifold, see Fig. 8, to compute the actual state change, x_{k+1}.

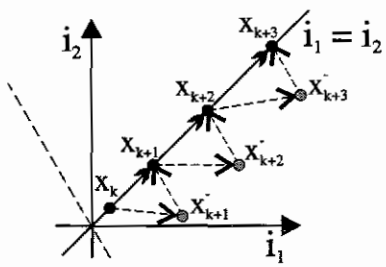

Figure 8: Numerical integration with additional projection step.

For n, $n > 1$, inductors cascaded as illustrated in Fig. 3, the projection equation becomes

$$p_{1,k+1} = \frac{\sum_{i=1}^{n} p^-_{i,k+1}}{\sum_{i=1}^{n+1} L_i} L_1 \qquad (7)$$

and when treated as independent states, one integration time step for each individual state results in

$$p^-_{i,k+1} = p_{i,k} + \Delta T(-\frac{R_i}{L_i} p_{i,k}). \qquad (8)$$

Summing these values for n coupled states yields

$$\sum_{i=1}^{n} p^-_{i,k+1} = \sum_{i=1}^{n} p_{i,k} + \Delta T(-\sum_{i=1}^{n} \frac{R_i}{L_i} p_{i,k} + nV_{cc}) \qquad (9)$$

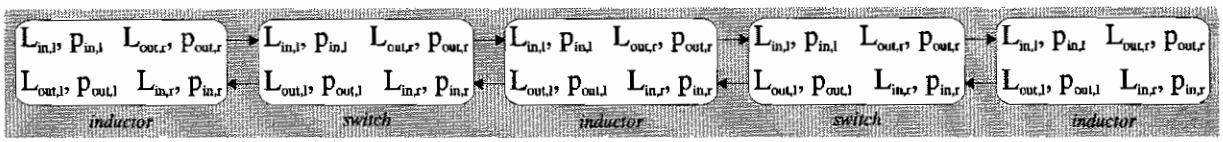

Figure 7: Compositional model of the inductor circuit.

This can be substituted in Eq. (7) to yield

$$p_{1,k+1} = \frac{\sum_{i=1}^{n} p_{i,k} + \Delta T(-\sum_{i=1}^{n} \frac{R_i}{L_i} p_{i,k})}{\sum_{i=1}^{n} L_i} L_1 \qquad (10)$$

Using the algebraic constraints $p_{i,k} = \frac{L_i}{L_1} p_{1,k}$ this can be expressed in terms of p_1 as

$$p_{1,k+1} = p_{1,k} - \Delta T \frac{\sum_{i=1}^{n} R_i}{\sum_{i=1}^{n} L_i} p_{1,k}, \qquad (11)$$

which verifies that behavior equals one integration step of dynamic behavior of a series connection of n inductors ($L_{tot} = \sum_{i=1}^{n} L_i$) and n resistors ($R_{tot} = \sum_{i=1}^{n} R_i$). Note that the positive contribution of V_{cc} at the beginning of the cascaded inductor/switch circuit is canceled by the negative contribution at the end (see Fig. 3) and the total voltage drop across the series connection of coupled inductors is zero. Other work has shown that projection methods in general can be integrated with sophisticated numerical simulation methods that employ interpolation to achieve high accuracy [Eic91].

5 Modeling the AC Induction Motor

The electrical circuitry of an induction motor contains a number of cascaded inductances, each with parasitic resistance, as shown in Fig. 9. To control the flux in each of the inductors, they are conneeted in series separated by a *bridge* consisting of a switch to ground and a switch to the source voltage. Each switch is equipped with a freewheeling diode to protect the electrical circuit from voltage spikes.

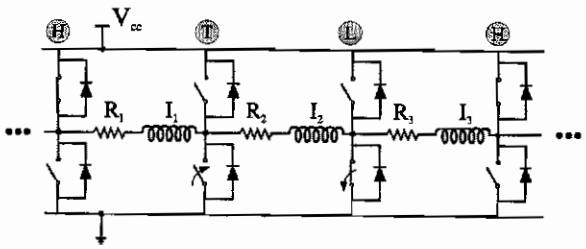

Figure 9: Electrical circuit for driving the induction motor.

The motor is driven by changing the flux in the induetors, depending on the angle of the rotor. To achieve the desired flux values, each inductor can be connected to the source voltage and ground in two different directions, causing a voltage drop V_{cc} or $-V_{cc}$. A complex scheme switches each of the switches between closed and open. When a switch opens, the corresponding freewheeling diode may become active until the two connected inductors draw the same current and they can be coupled without inducing a spike. If a bridge closes a current path to V_{cc} it operates in its *high* (H) state, if it closes a current path to ground, it operates in its *low* (L) state, and if neither current path is closed, it is in its *tri* (T) state, see Fig. 9.

The system is modeled in MATLAB-SIMULINK by the structure in Fig. 7 where the first and last components are connected. The constituent equations of the model components are shown in Fig. 6. Control logic switching is modeled by a state transition table and comparators model the internal event diode switching. A simulation run of six diodes connected as a ring, with three bridge state changes is shown in Fig. 10. The solid curve shows how the current from one inductor changes over time to achieve desired flux values. The dashed curves represent neighboring currents that may be coupled with the current of the solid or dashed curves. In Fig. 10(a), the gray intervals show periods of time when the freewheeling diodes become active, the *commuting* phase, resulting in C^0 hybrid behavior. In Fig. 10(b) the continuous transients because of the freewheeling diodes are abstraeted away, i.e., the diodes are removed from the model, to obtain faster simulation. As a result, the system includes discontinuities in state variables that are handled based on conservation of flux. In other work [Neu99], a detailed model of an induction motor using the projection prineiple is shown to have good conformance with actual measured data.

6 Conclusions

In physical system models, when fast continuous behavior is abstracted away algebraic constraints between state variables may arise that change dynamically. This changes the order of the system, and typically algebraic manipulations are required to derive this reduced order system and generate behavior. This paper shows that this can be circumvented by augmenting the model with projection equations. The equations describe a consistent projection onto the manifold in phase space constituted by the activated algebraic constraints. When the algebraic constraints are activated, the projection equations are first applied to find consistent new initial conditions in the new mode. Next, dynamic behavior is generated by making a numerical time step while disregarding the constraints and after new values are computed these are immediately correeted by

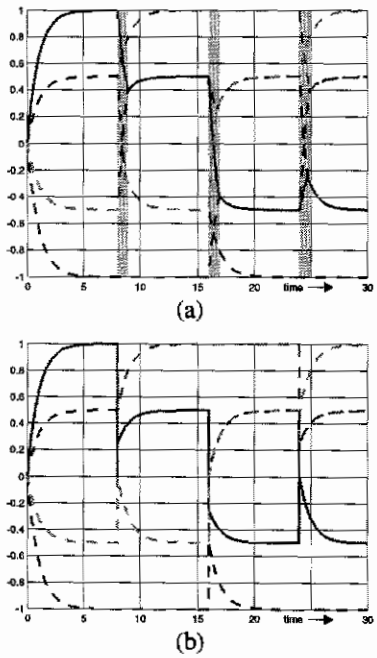

Figure 10: Induction motor simulation with (a) and without (b) commuting diodes.

applying the projection. This allows simulation of differential and algebraic equations (DAE) systems with variable index by MATLAB-SIMULINK. A modular approach is supported by designing model components for the two basic elements, inductor/resistance blocks and ideal bridges.

Overall, this presents an important result, i.e., that variable structure systems with complex behaviors can be simulated by ODE based simulation packages, in this case, MATLAB-SIMULINK. Furthermore, the modularized nature allows encapsulation of the model, disallowing access to the internal equation structure while still generating correct behavior when dependencies between state variables of several such modules arise. This is an important advance towards achieving component based modeling and simulation.

REFERENCES

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Lecture Notes in Computer Science*, volume 736, pages 209–229. Springer-Verlag, 1993.

[Eic91] Edda Eich. *Projizierende Mehrschrittverfahren zur numerischen Lösung von Bewegungsgleichungen technischer Mehrkörpersysteme mit Zwangsbedingungen und Unstetigkeiten*. PhD dissertation, Universität Augsburg, 1991.

[GJ95] John Guckenheimer and Stewart Johnson. Planar hybrid systems. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems II, Lecture Notes in Computer Science*, volume 999, pages 202–225. Springer-Verlag, 1995.

[KKO86] Petar V. Kokotović, Hassan K. Khalil, and John O'Reilly. *Singular Perturbation Methods in Control: Analysis and Design*. Academic Press, London, 1986. ISBN 0-12-417635-6.

[MB99a] 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.

[MB99b] Pieter J. Mosterman and Gautam Biswas. Deriving Discontinuous State Changes for Reduced Order Systems and the Effect on Compositionality. In 13^{th} *International Workshop on Qualitative Reasoning*, pages 160–168, Lock Awe Hotel, Scotland, June 1999.

[Mos97] Pieter J. Mosterman. *Hybrid Dynamic Systems: A hybrid bond graph modeling paradigm and its application in diagnosis*. PhD dissertation, Vanderbilt University, 1997.

[Mos98] Pieter J. Mosterman. State Space Projection onto Linear DAE Manifolds Using Conservation Principles. Technical Report #R262-98, Institute of Robotics and System Dynamics, DLR Oberpfaffenhofen, P.O. Box 1116, D-82230 Wessling, Germany, 1998.

[Mos99] 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, Lecture Notes in Computer Science*, volume 1569, pages 164–177. Springer-Verlag, March 1999.

[Neu99] Peter Neumann. *Modellbildung und Simulation eines Aussenläufenmotors*. Diplomarbeit, Institut für Elektrische Informationstechnik, Technische Universität Clausthal, 1999.

[Pan88] Constantinos C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific and Statistical Computing*, 9(2):213–231, March 1988.

[SIM97] SIMULINK. *Dynamic System Simulation for Matlab*. The MathWorks, January 1997.