Discrete Event Simulation of Hybrid Systems

Ernesto Kofman
Laboratorio de Sistemas Dinmicos - FCEIA - Universidad Nacional de Rosario
Riobamba 245 bis - (2000) Rosario - Argentina
Email: kofman@fceia.unr.edu.ar

Abstract

This paper describes the quantization–based integration methods and extends their use to the simulation of hybrid systems. Using the fact that these methods approximate ordinary differential equations (ODEs) and differential algebraic equations (DAEs) by discrete event systems, it is shown how hybrid systems can be approximated by pure discrete event simulation models (within the DEVS formalism framework). In this way, the treatment and detection of events representing discontinuities –which constitute an important problem for classic ODE solvers– does not introduce any difficulty. It can be also seen that the main advantages of quantization–based methods (error control, reduction of computational costs, possibilities of parallelization, sparsity exploitation, etc.) are still verified in presence of discontinuities. Finally, some examples which illustrate the use and the advantages of the methodology in hybrid systems are discussed.

Keywords: Hybrid systems, ODE simulation, DAE simulation, Discrete Event Systems, DEVS, Quantized Systems.

1 Introduction

Continuous system simulation is a topic which has advanced significantly with the appearance of modern computers. Based on classic methods for numerical resolution of ODE's like Euler, Runge–Kutta, Adams, etc., several variable–step and implicit ODE solver methods were developed. These modern methods –which usually make use of iteration rules and symbolic manipulation– allow the efficient simulation of complex systems, including DAE and variable structure systems.

Although there are several differences between the mentioned ODE solver algorithms, all of them share a property: they are based on time discretization. That is, they give a solution obtained from a difference equation system (i.e. a discrete–time model) which is only defined in some discrete instants.

A completely different approach for ODE numerical simulation has been being developed since the end of the 90's. In this new approach, the time discretization is replaced by the state variables quantization and a discrete event simulation model (within the DEVS formalism framework) is obtained instead of a discrete time one.

DEVS (Zeigler et al., 2000) is a formalism which allows representing and simulating any system having a finite number of changes in a finite interval of time. In that way, systems modelled by Petri

Nets, State Charts, Event Graphs, and even Difference Equations can be seen as particular cases of DEVS models.

The origin of the quantization based integration methods can be found in the definition of Quantized Systems and their representation in terms of DEVS models (Zeigler and Lee, 1998). This idea was reformulated with the addition of hysteresis and it was formalized as a simulation method for ODE's in (Kofman and Junco, 2001) where the Quantized State Systems (QSS) were defined.

This new method was improved with the definition of the Second Order Quantized State Systems (QSS2) (Kofman, 2002a) and then extended to the simulation of DAEs (Kofman, 2002c). In all the cases, the idea is to modify the continuous system with the addition of quantizers and then to represent and to simulate the resulting system by an equivalent DEVS model.

Despite their simplicity, the QSS and QSS2 methods satisfy some stability, convergence and error bound properties which are only shared by complex implicit discrete time algorithms. From the computational cost point of view, the quantization based methods also offer some advantages. They can reduce the number of calculations, their parallel implementation is straightforward and they can exploit structural properties like sparsity in a very efficient fashion.

When it comes to the simulation of hybrid systems and discontinuity handling, they have been always a problem for discrete time classic methods. The problem is that numerical integration algorithms in use are incompatible with the notion of discontinuous functions (Otter and Cellier, 1996) and an integration step which jumps along a discontinuity may produce an unacceptable error.

To avoid this, the methods should perform steps in the instants of time in which the discontinuities take place and then, the simulation of a hybrid system should be provided of tools for detecting the discontinuities occurrence (what includes iterations and extra computational costs), for adapting the step size to hit those instants of time and of course, to represent and simulate the discrete part of the system (which can be quite complicated itself) in interaction with the continuous part.

Although there are several methods and software tools which simulates hybrid systems in a quite efficient way, none of them can escape from these problems.

As it will be shown in this work, all the mentioned difficulties disappear with the use of the QSS and QSS2 methods. On one hand, the DEVS formalism solves the problem of the discrete part representation and interaction with the continuous part. On the other hand, the knowledge of the trajectory forms (which are piecewise linear and parabolic) transforms the discontinuity detection in a straightforward problem where iterations are not necessary.

Moreover, all these features are achieved without any kind of modification to the methodologies. In fact, all what has to be done is to connect a sub-system representing the discrete dynamics to the sub-system corresponding to the QSS or QSS2 approximation and it works. In that way, all the qualities of the methods are conserved (error control, reduction of computational costs, possibilities of parallelization, sparsity exploitation, etc.).

The simulation examples included not only corroborate what is said with respect to the reduction of the number of steps and calculations, the sparsity exploitation, the error control, etc. but they also show the simplicity of the implementation.

The paper is organized as follows: Section 2 introduces the DEVS formalism and the Quantization Based Integration Methods, describing their implementation as well as their main theoretical properties. Then, in Section 3 the details about the use of the methodology in a wide class of Hybrid Systems is presented. Finally, in Section 4, two hybrid examples are introduced and simulated with the QSS2–method. The results are then compared with the results obtained with all the methods

Figure 1: Input/Output behavior of a DEVS model

implemented in Matlab/Simulink in order to illustrate the advantages mentioned above.

2 DEVS and Quantization–Based Methods

QSS and QSS2 methods are based on the hysteretic quantization of the state variables. This quantization transforms the state trajectories into piecewise constant ones (or piecewise linear in QSS2) allowing the system representation and simulation in terms of the DEVS formalism.

2.1 DEVS Formalism

The DEVS formalism was developed by Bernard Zeigler in the mid-seventies (Zeigler, 1976; Zeigler et al., 2000). DEVS allows to represent all the systems whose input/output behavior can be described by sequence of events with the condition that the state has a finite number of changes in any finite interval of time.

A DEVS model processes an input event trajectory and, according to that trajectory and to its own initial conditions provokes an output event trajectory. This Input/Output behavior is represented in Figure 1.

Formally, a DEVS *atomic* model is defined by the following structure:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- X is the set of input event values, i.e., the set of all possible values that and input event can adopt.
- Y is the set of output event values.
- S is the set of state values.
- δ_{int}, δ_{ext}, λ and ta are functions which define the system dynamics.

Each possible state s ($s \in S$) has an associated *Time Advance* calculated by the *Time Advance Function* $ta(s)$ ($ta(s) : S \rightarrow \Re_0^+$). The *Time Advance* is a non-negative real number saying how long the system remains in a given state in absence of input events.

Thus, if the state adopts the value s_1 at time t_1, after $ta(s_1)$ units of time (i.e. at time $ta(s_1) + t_1$) the system performs an *internal transition* going to a new state s_2. The new state is calculated as $s_2 = \delta_{\text{int}}(s_1)$. The function δ_{int} ($\delta_{\text{int}} : S \rightarrow S$) is called *Internal Transition Function*.

When the state goes from s_1 to s_2 an output event is produced with value $y_1 = \lambda(s_1)$. The function λ ($\lambda : S \to Y$) is called *Output Function*. The functions ta, δ_{int} and λ define the autonomous behavior of a DEVS model.

When an input event arrives the state changes instantaneously. The new state value depends not only on the input event value but also on the previous state value and the elapsed time since the last transition. If the system goes to the state s_3 at time t_3 and then an input event arrives at time $t_3 + e$ with value x_1, the new state is calculated as $s_4 = \delta_{\text{ext}}(s_3, e, x_1)$ (note that $ta(s_3) > e$). In this case, we say that the system performs an *external transition*. The function δ_{ext} ($\delta_{\text{ext}} : S \times \Re_0^+ \times X \to S$) is called *External Transition Function*. No output event is produced during an external transition.

DEVS models can be coupled. One of the most used coupling schemes for DEVS models includes the use of ports. Here, the external transition functions of the atomic models distinguish the value and arrival port of the events to calculate the next state. Similarly, the output functions produce output events which carry a value through a given port. Then, the coupling basically consists in connections from output ports to input ports of different atomic models.

It was proven that DEVS is closed under coupling, i.e the coupling of different DEVS models behaves like an equivalent atomic DEVS model. Thus, the coupling can be done in a hierarchical way, using coupled DEVS models as if they were atomics.

The simulation of a coupled DEVS model is very simple and efficient. Despite the various software tools developed to allow the simulation of DEVS models, a DEVS simulation can be performed directly in any object oriented programming language. Moreover, some improvements like parallelization and flattening can be applied in a very direct fashion.

2.2 QSS–Method

Consider a time invariant ODE in its State Equation System (SES) representation:

$$\dot{x}(t) = f[x(t), u(t)] \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector and $u(t) \in \mathbb{R}^m$ is an input vector, which is a known piecewise constant function.

The QSS–method (Kofman and Junco, 2001) simulates an approximate system, which is called Quantized State System:

$$\dot{x}(t) = f[q(t), u(t)] \tag{2}$$

where $q(t)$ is a vector of *quantized variables* which are quantized versions of the state variables $x(t)$. Each component of $q(t)$ is related with the corresponding component of $x(t)$ by a hysteretic quantization function, which is defined as follows:

Definition 1. *Let $Q = \{Q_0, Q_1, ..., Q_r\}$ be a set of real numbers where $Q_{k-1} < Q_k$ with $1 \leq k \leq r$. Let Ω be the set of piecewise continuous real valued trajectories and let $x_i \in \Omega$ be a continuous trajectory. Let $b : \Omega \to \Omega$ be a mapping and let $q_i = b(x_i)$ where the trajectory q_i satisfies:*

$$q_i(t) = \begin{cases} Q_m & \text{if } t = t_0 \\ Q_{k+1} & \text{if } x_i(t) = Q_{k+1} \wedge q_i(t^-) = Q_k \wedge k < r \\ Q_{k-1} & \text{if } x_i(t) = Q_k - \varepsilon \wedge q_i(t^-) = Q_k \wedge k > 0 \\ q_i(t^-) & \text{otherwise} \end{cases} \tag{3}$$

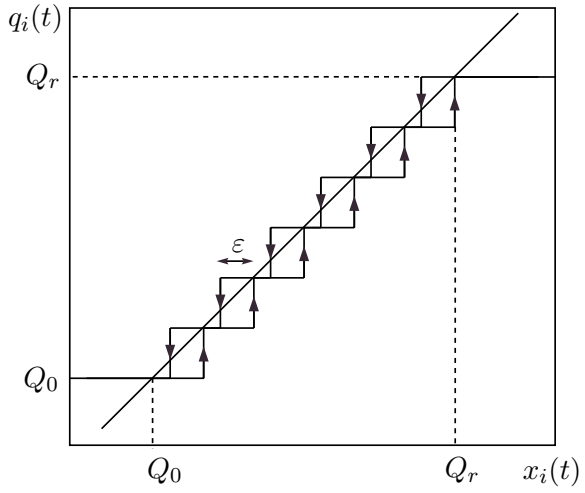

Figure 2: Quantization Function with Hysteresis

and

$$m = \begin{cases} 0 & \text{if } x_i(t_0) < Q_0 \\ r & \text{if } x_i(t_0) \geq Q_r \\ j & \text{if } Q_j \leq x(t_0) < Q_{j+1} \end{cases}$$

Then, the map b is a hysteretic quantization function.

The discrete values Q_k are called *quantization levels* and the distance $Q_{k+1} - Q_k$ is defined as the *quantum*, which is usually constant. The width of the hysteresis window is ε. The values Q_0 and Q_r are the lower and upper saturation bounds. Figure 2 shows a typical quantization function with uniform quantization intervals.

In (Kofman and Junco, 2001) it was proven that the quantized variable trajectories $q_i(t)$ are piecewise constant and the state variables $x_i(t)$ are piecewise linear. As a consequence, the QSS can be simulated by a DEVS model. There, it is also shown that the use of hysteresis in QSS is necessary to ensure those properties. If non hysteretic –or memoryless– quantization were used, infinitely fast oscillations can occur and the resulting DEVS model will produce an infinite number of events in a finite interval of time[1].

A generic QSS (2) can be represented by the Block Diagram of Figure 3. That Block Diagram also shows a possible coupling scheme for the corresponding DEVS model.

Based on Figure 3, a simulation model can be built as the coupling of n atomic DEVS models representing the integrators and quantizers (or *quantized integrators*) and other n atomic DEVS models which simulate the behavior of the static functions f_i.

The DEVS simulation of quantized integrators and static functions is based on the representation of their piecewise constant input and output trajectories by sequences of events where each event represents a change in the corresponding trajectory.

[1]Such a DEVS model is called illegitimate (Zeigler et al., 2000) and it cannot be simulated

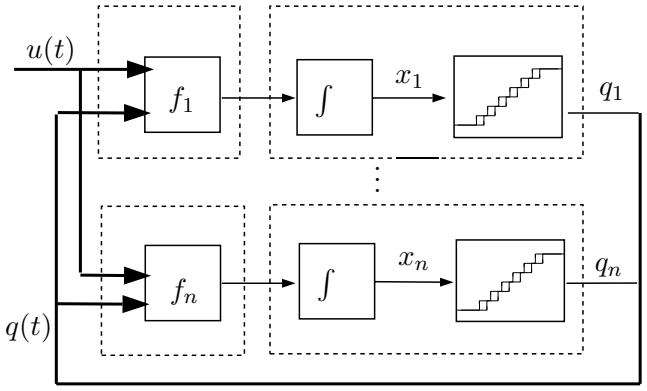

Figure 3: Block Diagram Representation of a QSS

A DEVS model which simulates a quantized integrator, with quantization levels $Q_0, \ldots, Q_r$ and hysteresis width ε can be written as follows:

$$
\begin{aligned}
&M_1 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\
&X = \mathbb{R} \times \{\text{inport}\} \\
&Y = \mathbb{R} \times \{\text{outport}\} \\
&S = \mathbb{R}^2 \times \mathbb{Z} \times \mathbb{R}_0^+ \infty \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(x, d_x, k, \sigma) = (x + \sigma \cdot d_x, d_x, k + \text{sgn}(d_x), \sigma_1) \\
&\delta_{\text{ext}}(s, e, x_u) = \delta_{\text{ext}}(x, d_x, k, \sigma, e, x_v, port) = (x + e \cdot d_x, x_v, k, \sigma_2) \\
&\lambda(s) = \lambda(x, d_x, k, \sigma) = (Q_{k+\text{sgn}(d_x)}, \text{outport}) \\
&ta(s) = ta(x, d_x, k, \sigma) = \sigma
\end{aligned}
$$

where

$$
\sigma_1 = \begin{cases} \dfrac{Q_{k+2} - (x + \sigma \cdot d_x)}{d_x} & \text{if } d_x > 0 \\ \dfrac{(x + \sigma \cdot d_x) - (Q_{k-1} - \varepsilon)}{|d_x|} & \text{if } d_x < 0 \\ \infty & \text{if } d_x = 0 \end{cases}
$$

and

$$
\sigma_2 = \begin{cases} \dfrac{Q_{k+1} - (x + e \cdot x_v)}{x_v} & \text{if } x_v > 0 \\ \dfrac{(x + e \cdot x_v) - (Q_k - \varepsilon)}{|x_v|} & \text{if } x_v < 0 \\ \infty & \text{if } x_v = 0 \end{cases}
$$

A static function $f(z_1, \ldots, z_p)$ can be represented by the DEVS model

$$
\begin{aligned}
&M_2 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\
&X = \mathbb{R} \times \{\text{inport}_1, \ldots, \text{inport}_p\} \\
&Y = \mathbb{R} \times \{\text{outport}\} \\
&S = \mathbb{R}^p \times \mathbb{R}_0^+ \infty \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(z_1, \ldots, z_p, \sigma) = (z_1, \ldots, z_p, \infty) \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(z_1, \ldots, z_p, \sigma, e, x_v, port) = (\tilde{z}_1, \ldots, \tilde{z}_p, \infty) \\
&\lambda(s) = \lambda(z_1, \ldots, z_p, \sigma) = (f(z_1, \ldots, z_p, \sigma), \text{outport}) \\
&ta(s) = ta(z_1, \ldots, z_p, \sigma) = \sigma
\end{aligned}
$$

where

$$
\tilde{z}_j = \begin{cases} x_v & \text{if } port = \text{inport}_j \\ z_j & \text{otherwise} \end{cases}
$$

Then, the QSS–method can be directly applied after choosing the quantization intervals and hysteresis width for each integrator and coupling the resulting DEVS models M_1 and M_2 according to Figure 3.

One of the advantages of this kind of implementation is that each step only involves calculations in the quantized integrator which performs the internal transition and eventually in those quantized integrator connected to it through static functions. In that way, the simulation model can exploit the system sparsity in a very efficient fashion.

2.3 The QSS2-Method

Although the QSS–method satisfies nice stability, convergence and error bound properties –which will be recalled later– it only performs a first order approximation. Thus, the error reduction cannot be accomplished without an important increment of the computational costs.

This problem was solved in (Kofman, 2002a), where a second order approximation was proposed which also shares the main properties and advantages of the QSS–method.

The basic idea of QSS2 is the use of first–order quantization functions instead of the quantization function of Figure 2. Then, the simulation model can be still represented by (2) but now $q(t)$ and $x(t)$ have a different relationship. This new system is called Second Order Quantized State System or QSS2 for short.

The first–order quantization function can be seen as a function which gives a piecewise linear output trajectory, whose value and slope change when the difference between this output and the input becomes bigger than certain threshold. Figure 4 illustrates this idea.

Formally, we say that the trajectories $x_i(t)$ and $q_i(t)$ are related by a first–order quantization function if they satisfy:

$$
q_i(t) = \begin{cases} x_i(t) & \text{if } \; t = t_0 \vee |q_i(t^-) - x_i(t^-)| = \Delta q \\ q_i(t_j) + m_j(t - t_j) & \text{otherwise} \end{cases}
$$

with the sequence $t_0, \ldots, t_j, \ldots$ defined as

$$
t_{j+1} = \min(t | t > t_j \wedge |x_i(t_j) + m_j(t - t_j) - x_i(t)| = \Delta q)
$$

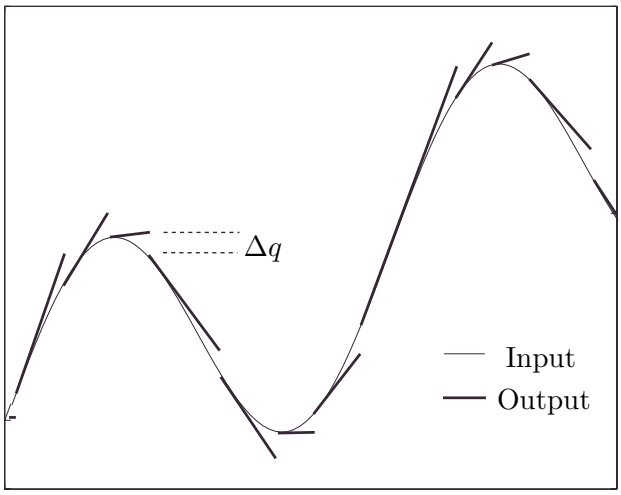

Figure 4: Input and Output trajectories in a *First Order* quantizer

and the slopes are

$$m_0 = 0,\ m_j = \dot{x}_i(t_j^-) \quad j = 1, \ldots, k, \ldots$$

Here Δq plays the role of the quantum and hysteresis width. In fact, in most applications of QSS they are chosen equal to each other, since it is the best election from the point of view of the trade–off between the error and computational costs, as shown in (Kofman et al., 2001).

The DEVS representation is still possible but it is only exact in LTI systems with piecewise linear input trajectories. The reason is that in nonlinear systems we do not know the state derivative trajectory form.

In a QSS2 coming from a LTI system the quantized variable and state derivative trajectories are piecewise linear and the state variable have piecewise parabolic trajectories. Then, they can be represented by DEVS models but now, the events should carry not only the new value of a trajectory but also the new slope.

Anyway, the idea behind the DEVS model construction is similar to the first order method. We represent it as a coupled DEVS model like the one shown in Figure 3, but the atomic models are redefined in order to obtain the new behavior.

The DEVS model corresponding to a *second–order quantized integrator* (i.e. an integrator with a first–order quantizer) can be written as follows.

$M_3 =< X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta >,$ where:
$X = \mathbb{R}^2 \times \{\text{inport}\}$
$S = \mathbb{R}^5 \times \mathbb{R}_0^+ \infty$
$Y = \mathbb{R}^2 \times \{\text{outport}\}$
$\delta_{\text{int}}(u, m_u, x, q, m_q, \sigma) = (u + m_u \cdot \sigma, m_u, x + u \cdot \sigma + \frac{m_u}{2}\sigma^2, x + u \cdot \sigma + \frac{m_u}{2}\sigma^2, u + m_u \cdot \sigma, \sigma_1)$
$\delta_{\text{ext}}(u, m_u, x, q, m_q, \sigma, e, v, m_v, port) = (v, m_v, x + u \cdot e + \frac{m_u}{2}e^2, q + m_q \cdot e, m_q, \sigma_2)$
$\lambda(u, m_u, x, q, m_q, \sigma) = (x + u \cdot \sigma + \frac{m_u}{2}\sigma^2, u + m_u \cdot \sigma, \text{outport})$
$ta(u, m_u, x, q, m_q, \sigma) = \sigma$

where

$$\sigma_1 = \begin{cases} \sqrt{\frac{2\Delta q}{m_u}} & \text{if} \quad m_u \neq 0 \\ \infty & \text{otherwise} \end{cases} \tag{4}$$

and σ_2 can be calculated as the least positive solution of

$$|x + u \cdot e + \frac{m_u}{2}e^2 + v \cdot \sigma_2 + \frac{m_v}{2}\sigma_2^2 - (q + m_q \cdot e + m_q\sigma_2)| = \Delta q \tag{5}$$

The DEVS model associated to a generic static function $f(z_1, \ldots, z_p)$ taking into account values and slopes can be written according to:

$M_4 =< X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta >,$ where:
$X = \mathbb{R}^2 \times \{\text{inport}_1, ..., \text{inport}_p\}$
$S = \mathbb{R}^{3p} \times \mathbb{R}_0^+ \infty$
$Y = \mathbb{R}^2 \times \{\text{outport}_1, ..., \text{outport}_p\}$
$\delta_{\text{int}}((z_1, m_{z_1}, c_1), ..., (z_p, m_{z_p}, c_p), \sigma) = ((z_1, m_{z_1}, c_1), ..., (z_p, m_{z_p}, c_p), \infty)$
$\delta_{\text{ext}}((z_1, m_{z_1}, c_1), ..., (z_p, m_{z_p}, c_p), \sigma, e, v, mv, port) = ((\tilde{z}_1, \tilde{m}_{z_1}, \tilde{c}_1), ..., (\tilde{z}_p, \tilde{m}_{z_p}, \tilde{c}_p), 0)$
$\lambda((z_1, m_{z_1}, c_1), ..., (z_p, m_{z_p}, c_p), \sigma) = (f(z_1, ..., z_p), c_1 m_{z_1} + ... + c_p m_{z_p}, 1)$
$ta((z_1, m_{z_1}, c_1), ..., (z_p, m_{z_p}, c_p), \sigma) = \sigma$

with

$$\tilde{z}_j = \begin{cases} v & \text{if} \quad port = \text{inport}_j \\ z_j + m_{z_j} e & \text{otherwise} \end{cases}$$

$$\tilde{m}_{z_j} = \begin{cases} m_v & \text{if} \quad port = \text{inport}_j \\ m_{z_j} & \text{otherwise} \end{cases}$$

$$\tilde{c}_j = \begin{cases} \frac{f(z + m_z e) - f(\tilde{z})}{z_j + m_{z_j} e - \tilde{z}_j} & \text{if} \quad port = \text{inport}_j \wedge z_j + m_{z_j} e - \tilde{z}_j \neq 0 \\ c_j & \text{otherwise} \end{cases} \tag{6}$$

If the function f is linear, this DEVS model exactly represents its behavior. Equation (6) calculates the coefficients that multiply the input trajectory slopes.

In the nonlinear case, the output trajectory of function f will not be piecewise linear. However, the trajectory given by the DEVS model, which is interpreted as piecewise linear, constitutes a good approximation to the true output. The reason of this is that the coefficients c_j, calculated with (6), are closed to the corresponding partial derivatives of f evaluated at the points given by the input trajectories. Thus, we can affirm that the DEVS model of a static function can be applied to general nonlinear functions and then general nonlinear systems can be simulated under the QSS2 approach.

2.4 Theoretical Properties of QSS and QSS2

The properties of QSS and QSS2 related to the trajectory forms were already mentioned to explain their DEVS representation.

Despite the importance of those properties –which guarantee the possibility of simulating QSS and QSS2 using DEVS– they do not ensure that the QSS and QSS2 solutions are close to the solutions of the SES (1).

However, there are more properties which, based on the representation of the QSS and QSS2 as perturbed SES, show that QSS and QSS2 are good approximations to the continuous systems. These properties –which were proven in (Kofman and Junco, 2001) and (Kofman, 2002a)– not only show theoretical features but also allow deriving rules for the choice of the quantization.

Let us define $\Delta x(t) = q(t) - x(t)$. Then, (2) can be rewritten

$$\dot{x}(t) = f[x(t) + \Delta x(t), u(t)] \tag{7}$$

From the definition of the hysteretic and the first order quantization functions, it can be ensured that each component of Δx is bounded by the corresponding quantum adopted. Thus, the QSS and QSS2 methods simulate an approximate system which only differs from the original SES (1) due to the presence of the bounded state perturbation $\Delta x(t)$. Then, based on this fact, the following properties were proven:

- Under certain conditions, the solutions of a QSS (or QSS2) associated to a continuous system converge to the solutions of the last one when the quantization goes to zero (Convergence).
- It is always possible to find a quantization so that the solutions of the QSS (QSS2) associated to an asymptotically stable continuous system finish inside an arbitrary small region around the equilibrium points of the originally continuous system (Stability[2]).
- In stable and decoupleable LTI systems, the QSS and QSS2 simulation trajectories never differ from the solutions of (1) in more than a bound which can be calculated using a closed formula which depends on the quantum adopted (Error Bound)

The Convergence Property ensures that an arbitrarily small error can be achieved by using a sufficiently small quantization. A sufficient condition which guaranties this property is that the function f is locally Lipschitz.

[2]In fact, we should not talk about stability. What we ensure is ultimate boundedness of the solutions (Khalil, 1996)

The Stability Property relates the quantum adopted with the final error. An algorithm can be derived from the proof of this property which allows the choice of the quantum to be used in the different state variables. However, this is not very practical since it is based on a Lyapunov analysis and the algorithm requires the use of a Lyapunov function. Anyway, this is a very strong theoretical property since it holds for general nonlinear systems.

Finally, the Error Bound is probably the most important property of quantization based methods. Given a LTI system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{8}$$

where A is a Hurwitz and diagonalizable matrix, the error in the QSS or QSS2 simulation is always bounded by

$$|\tilde{\phi}(t) - \phi(t)| \leq |V||\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda||V^{-1}|\Delta q \tag{9}$$

where Λ and V are the matrices of eigenvalues and eigenvectors of A (Λ is diagonal), that is

$$V^{-1}AV = \Lambda$$

and Δq is the vector of quantum adopted at each component.

The symbol $|\cdot|$ denotes the componentwise module of a complex vector and the symbol "$\leq$" in (9) means that each component of the error $|e(t)|$ is always less or equal than the bound calculated at the right hand of the inequality.

Inequality (9) holds for all t, for any input trajectory and for any initial condition. It can be used to choose the quantum Δq in order to satisfy a desired error bound and it also implies very important theoretical properties: the existence of a linear relationship between the error and the quantum and the fact that the error is always bounded and thus we cannot obtain unstable results with the QSS or QSS2 method in the simulation of LTI stable systems.

Finally, it should be mentioned that in many applications the input trajectory is not piecewise constant or piecewise linear. Anyway, it can be approximated by a piecewise constant (or linear) trajectory with the use of an appropriate quantizer. Although this input quantization introduces a new error, the properties we mentioned are still satisfied, but Equation (9) now becomes

$$|\tilde{\phi}(t) - \phi(t)| \leq |V||\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda||V^{-1}|\Delta q + |V||\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}B|\Delta u \tag{10}$$

where Δu is a vector with the quanta adopted in each input component. Inequality (10) can be easily deduced from (Kofman, 2002b).

2.5 QSS and QSS2 Simulation of DAE Systems

There are many continuous systems where an explicit ODE formulation cannot be easily obtained (Cellier, 1996). Indeed, there are cases in which that representation does not exist. These systems, where only implicitly defined state equations can be written, are called Differential Algebraic Equation systems.

The basic idea for an efficient treatment of DAE system consists in applying the ODE solver rules directly on the original DAE (Gear, 1971). This is the idea followed to simulate DAE systems of index 1 with quantization-based methods in (Kofman, 2002c).

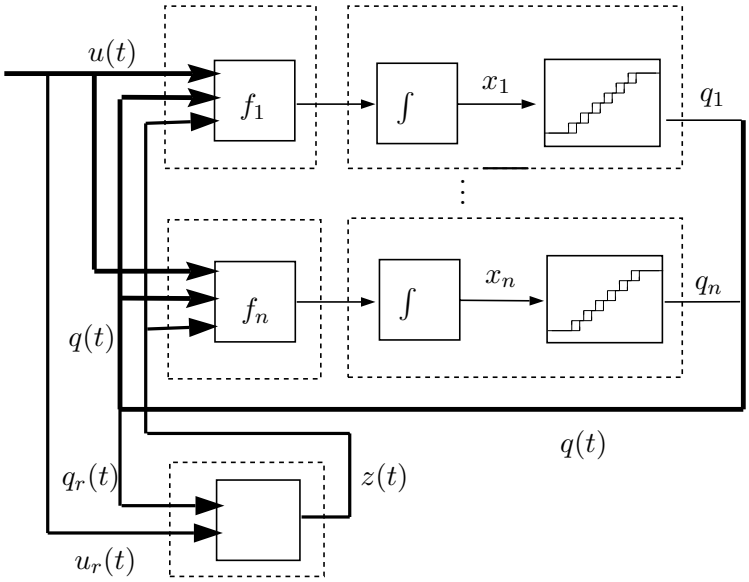

Figure 5: Coupling scheme for the QSS simulation of (11)

A time invariant DAE can be written as

$$\dot{x}(t) = f[x(t), u(t), z(t)] \tag{11a}$$
$$0 = g[x_r(t), u_r, z(t)] \tag{11b}$$

where $z(t)$ is a vector of algebraic variables whose dimension is equal or less than n. The vectors x_r and u_r are reduced versions of x and u respectively.

Equation (11b) expresses the fact that some state and input variables may not act directly on the algebraic loops.

Then, the use of the QSS or QSS2 method transforms (11) into

$$\dot{x}(t) = f[q(t), u(t), z(t)] \tag{12a}$$
$$0 = g[q_r(t), u_r, z(t)] \tag{12b}$$

Here, the iterations should be only performed to solve Eq.(12b) when the components of q_r or u_{q_r} change. When the dimension of x_r is significantly less than the dimension of x, i.e. when there are several state variables which do not influence on the loops, this fact represents an important advantage.

When it comes to the DEVS representation, Subsystem (12a) can be represented by quantized integrators and static functions as it was done before. The only difference now is the presence of the algebraic variables z which act as inputs like u. However, while the components of u are known and they may come from DEVS signal generators, the algebraic variables should be calculated by solving the restriction (12b). Figure 5 shows the new coupling scheme with the addition of a new DEVS model which calculates z.

A DEVS model which solves a general implicit equation like

$$g(v,z) = g(v_1,\ldots,v_m,z_1,\ldots,z_k) = 0 \tag{13}$$

for the QSS case can be written as follows

$$\begin{aligned}
&M_5 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where}\\
&X = \mathbb{R} \times \{\text{inport}_1; \ldots; \text{inport}_m\}\\
&Y = \mathbb{R}^k \times \{\text{outport}\}\\
&S = \mathbb{R}^{m+k} \times \mathbb{R}^+\\
&\delta_{\text{ext}}(s,e,x) = \delta_{\text{ext}}(v,z,\sigma,e,x_v,p) = (\tilde{v}, h(\tilde{v},z), 0)\\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(v,z,\sigma) = (v,z,\infty)\\
&\lambda(s) = \lambda(v,z,\sigma) = (z, \text{outport})\\
&ta(s) = ta(v,z,\sigma) = \sigma
\end{aligned}$$

where

$$\tilde{v} = (\tilde{v}_1,\ldots,\tilde{v}_m)^T;\;\; \tilde{v}_i = \begin{cases} x_v & \text{if} \quad p = \text{inport}_i \\ v_i & \text{otherwise} \end{cases}$$

and the function $h(v,z)$ returns the result of applying Newton iteration or some other iteration rules to find the solution of (13) using an initial value z.

When the size of z (i.e. k) is greater than 1, the output events of model M_5 contains a vector. Thus, they cannot be sent to static functions like M_2. Anyway, we can use a DEVS model which demultiplexes the vectorial input value into scalar output values at different ports in order to solve this difficulty.

The idea for the DAE simulation with the QSS2–method is similar, but now the algebraic variable slopes must be also calculated. The explanation of this and corresponding DEVS model are presented in (Kofman, 2002c).

3 Hybrid Systems and Quantization–based Methods

The complexity of most technical systems yields models which often combine a continuous part (described by ODEs or DAEs) and discrete components. The interaction between these subsystems can produce sudden changes (discontinuities) in the continuous part which must be handled by the integration algorithms.

The mentioned sudden changes are called *events* and two different cases can be distinguished according to the nature of their occurrence. The events which occur at a given time, independently of what happens in the continuous part are called *Time Events*. On the other hand, events which are produced when the continuous subsystem state reaches some condition are called *State Events*.

The integration along discontinuities without event detection techniques can cause severe inefficiency, and even simulation failures or incorrect event sequences to be generated, because the non–smoothness violates the theoretical assumptions on which solvers are founded (Barton, 2000). Thus, time and state events must be detected in order to perform steps at their occurrence.

The incorporation of event detection techniques to numerical methods have been being studied since Cellier's Thesis (Cellier, 1979) and many works can be found in the recent literature (see for instance (Park and Barton, 1996; Taylor and Kebede, 1996; Schlegl et al., 1997; Esposito et al., 2001)).

Although these ideas work quite efficiently, the techniques do not say how to represent discrete parts and how to schedule the time events in general cases. Moreover, the state event detection requires performing some iterations to find the time of the event occurrence.

All these problems disappear with the use of quantization based integration methods. On one hand, the discrete part representation can be easily solved with a DEVS model which sends events to the continuous part. On the other hand, the state trajectories are piecewise linear or parabolic and then the state event detection can be done without any iteration.

To achieve this, the only thing which should be done is to approximate the continuous part by a QSS or a QSS2 and to represent the discrete part by a DEVS model.

3.1 Continuous Part Approximation

There is not a unified representation of hybrid systems in the literature. Anyway, the different approaches coincide in describing them as sets of ODEs or DAEs which are selected according to some variable which evolves in a discrete way (different examples of hybrid systems representation can be found in (Taylor, 1993; Branicky, 1994; Broenink and Weustink, 1996; Barton, 2000)).

Here, it will be assumed that the continuous subsystem can be represented by

$$\dot{x}(t) = f[x(t), u(t), z(t), m(t)] \tag{14a}$$

$$0 = g[x_r(t), u_r(t), z(t), m(t)] \tag{14b}$$

being $m(t)$ a piecewise constant trajectory coming from the discrete part, which defines the different modes of the system. Thus, for each value of $m(t)$ there is a different DAE representing the system dynamics.

It will be considered that the implicit equation (14b) has a solution for each value of $m(t)$ (which implies that the system (14) has always index 1).

Independently of the way in which $m(t)$ is calculated, the simulation sub-model corresponding to the continuous part can be built considering that $m(t)$ acts as an input.

Then, the QSS and QSS2 methods applied to this part will transform (14) into:

$$\dot{x}(t) = f[q(t), u(t), z(t), m(t)] \tag{15a}$$

$$0 = g[q_r(t), u_r(t), z(t), m(t)] \tag{15b}$$

with the same definitions done in (12). Thus, the simulation scheme for the continuous part will be identical to the one shown in Figure 5, but now $m(t)$ must be included with the input.

3.2 Discrete Part Representation

One of the most important features of DEVS is its capability to represent all kind of discrete systems. Taking into account that the continuous part is being approximated by a DEVS model, it is natural representing also the discrete behavior by another DEVS model. Then, both DEVS models can be directly coupled to build a unique DEVS model which approximates the whole system.

In presence of only Time Events, the DEVS model representing the discrete part will be just an event generator, i.e. a DEVS model which does not receive any input and produces different output events at different times. These output events will carry the successive values of $m(t)$

Then, the simulation of the complete hybrid system can be performed by coupling this time event generator with the continuous part.

Taking into account the asynchronous way in which the static functions and quantized integrators work, the events will be processed by the continuous part as soon as they come out from the generator without the need of modifying anything in the QSS or QSS2 methods. This efficient event treatment is just due to intrinsic behavior of the methods.

This fact makes a big difference with respect to discrete time methods which must be modified in order to hit the event times.

When it comes to state events, the discrete part is ruled not only by the time advance but also by some events which are produced when the input and state variables reach some condition.

Here, the QSS and QSS2 methods have a bigger advantage: The state trajectories are perfectly known for all time. Moreover, they are piecewise linear or piecewise parabolic functions which implies that detecting the event occurrence is straightforward.

The only thing which has to be done is to provide those trajectories to the discrete part so it can detect the event occurrence and it can calculate the trajectory $m(t)$. Since the state trajectories are only known inside the quantized integrators, these models could be modified in order to output not only the quantized variables but also the state trajectories.

However, this is not necessary. The discrete part can receive the state derivative trajectories and then integrate them. It is simple and does not require computational effort since the derivative trajectories are piecewise constant or piecewise linear (in QSS2) and their integration only involves the manipulation of the polynomial coefficients.

Using these ideas, the simulation model for a hybrid system like (14) using the QSS or QSS2 method will be a coupled DEVS with the structure shown in Figure 6.

Here the discrete part is a DEVS model which receives the events representing changes in the state derivatives as well as changes in the input trajectories.

Since the discrete model receives and produce only a finite number of events in any finite interval of time (because of its definition as a discrete model), we can ensure that a DEVS model can represent it no matter how complex is its dynamic. Taking into account this, the scheme of Figure 6 can simulate any systems like (14) in interaction with any discrete model.

There are cases in which this scheme can be simplified. As we mentioned before, when only Time Events are considered, the DEVS model of the discrete part will not have inputs.

Usually, the event occurrence condition is related to a zero (or another fixed value) crossing of some state variable. In this case, if the simulation is performed with the QSS–method the event condition can be detected directly by the corresponding quantized integrator. This can be easily done provided that the quantization functions contain quantization levels at the given fixed crossing values.

4 Examples and Results

In order to show the uses and the advantages of quantization–based integration methods in hybrid systems, we introduce here two simple examples. The first one is ruled by time events while the second

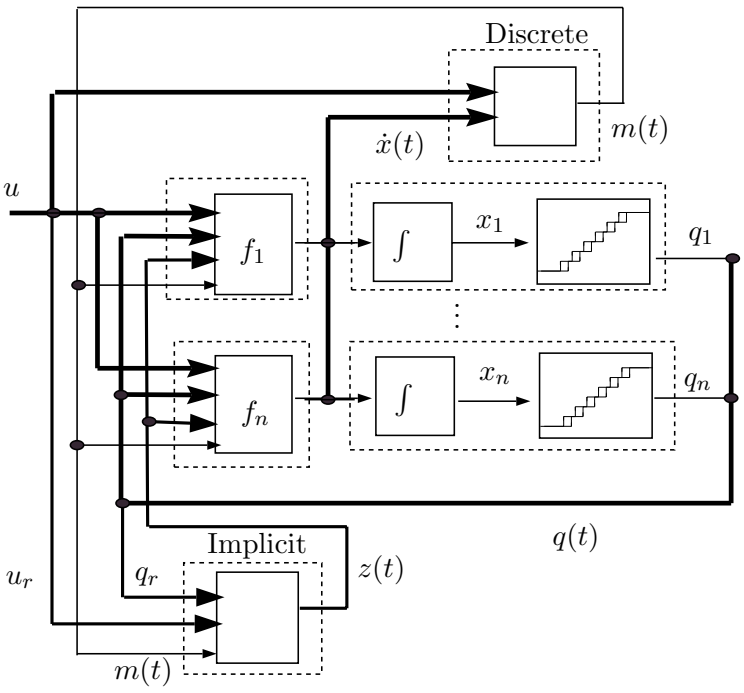

Figure 6: Coupling scheme for the QSS simulation of a hybrid system

one contains state events.

Further examples of hybrid systems simulation with the QSS–method can be found in (Kofman, 2001) and (Kofman, 2002d), where continuous plants with asynchronous controllers were simulated.

4.1 DC-AC inverter circuit

The inverter circuit shown in Figure 7 can be used to feed different electrical machines. The set of

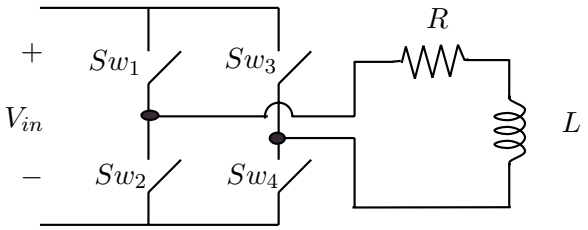

Figure 7: DC-AC Full Bridge Inverter

switches can take two positions. In the first one the switches 1 and 4 are closed and the load receives

a positive voltage. In the second position the switches 2 and 3 are closed and the load receives a negative voltage.

The system can be represented by the following differential equation:

$$\frac{d}{dt}i_L = -\frac{R}{L} \cdot i_L + s_w \cdot V_{in} \tag{16}$$

where s_w is 1 or -1 according to the position of the switches.

A typical way of controlling the switches in order to obtain a harmonic current at he load is using a pulse width modulation (PWM) strategy. The PWM signal is obtained by comparing a triangular wave (carrier) with a modulating sinusoidal reference. The sign of the voltage to be applied ($+V_{in}$ or $-V_{in}$) and the corresponding position is given by the sign of the difference between those signals. Figure 8 shows this idea.

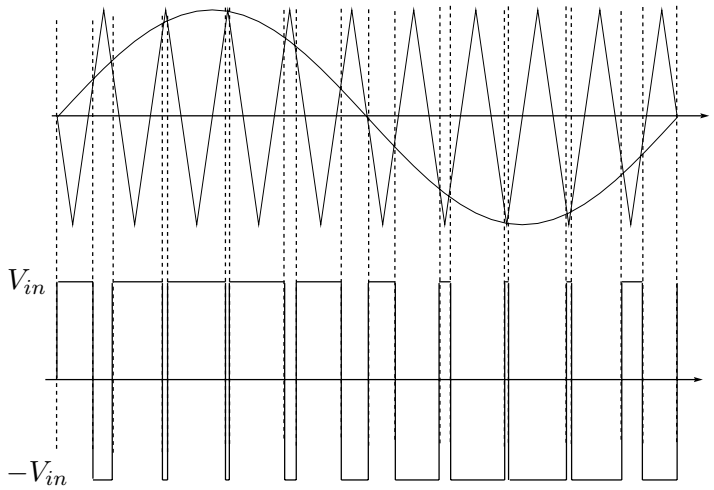

Figure 8: Pulse Width Modulation

In this strategy, the switches change their position independently of what happens in the circuit. Then, the events representing those changes are *time events*.

The system was simulated with the QSS2–method adding to the scheme of Figure 3 a block which produces events at the corresponding times with values $+V_{in}$ and $-V_{in}$.

These times where calculated for a carrier frequency of 1.6kHz and a modulating sinusoidal signal of the same amplitude and a frequency of 50Hz. Thus, the number of events per cycle was 64, which is enough to produce a quite smooth sinusoidal current.

Using parameters $R = 0.6\Omega$, $L = 100$mHy and $V_{in} = 300$V the simulation starting from $i_L = 0$ and taking a quantization $\Delta i_L = 0.01$A gave the result shown in Figures 9–10.

The final time of the simulation was 1 second and then the number of cycles was 50. This gives a total of 3200 changes in the position of the switches.

Despite this number of events, the simulation was completed after only 3100 internal transitions at the second order quantized integrator. Thus, the total number of steps was 6300.

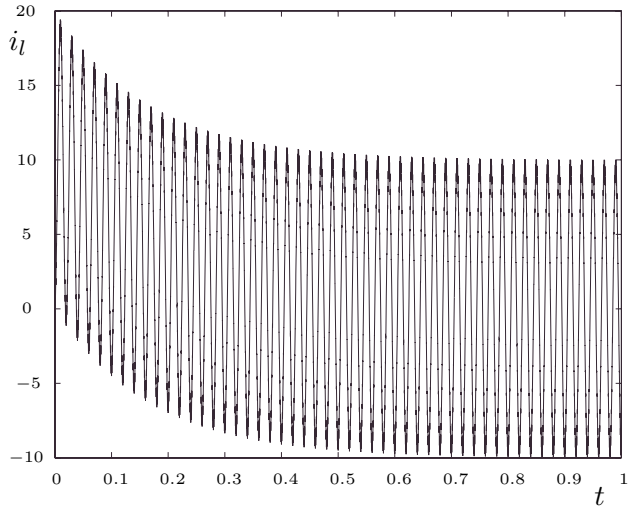

Figure 9: Load current with Pulse Width Modulation

In this case, since the commutations do not produce any structural change (they only affect the input voltage sign), the formula (9) can be applied and it can be ensured that the error in the trajectory of i_L obtained is always less than 10mA (which is about the 0.1% of the oscillation amplitude) .

The same system was simulated with all the discrete time methods implemented in Simulink. The fixed step ode5 algorithm (5th order) needed more than 50000 steps to obtain an acceptable result. Of course, lower order fixed step methods required more steps.

Using variable step methods the result was even worse. Only the ode23s methods gave acceptable results with about 100000 steps.

The simulations were repeated with variable step methods enforcing additional calculations at the event times. In this case they worked sensibly better. Anyway, using the tolerance obtained with QSS2, the ode23 (which now showed the best performance) needed more than 20000 steps to complete the simulation.

However, this trick –enforcing calculations at predetermined time instants– cannot be used in general cases since often the event times are not known before the simulation starts. In the PWM case it is usual to calculate them during the simulation since the frequency and amplitude of the modulating signal often change according to control strategies.

4.2 A ball bouncing downstairs

A typical example of a discontinuous system is the bouncing ball. Here, we shall consider the case in which the ball moves in two dimensions (x and y) bouncing downstairs. Thus, the bouncing condition depends on both variables (x and y).

It will be assumed that the ball has a model when it is in the air –with the presence of friction– and a different model in the floor. Here, a spring–damper model will be consider.

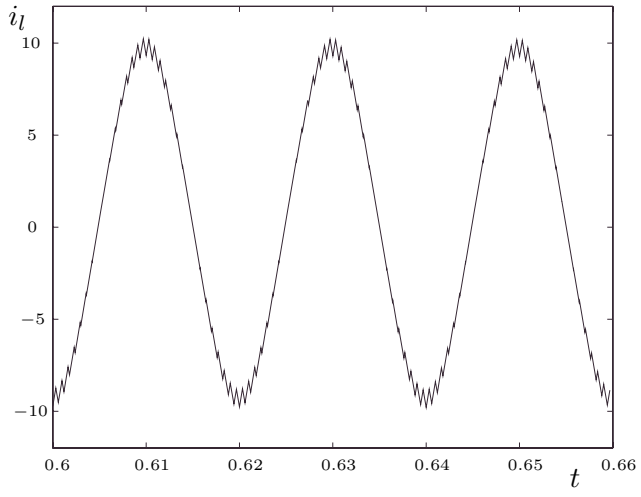

Figure 10: Detail of the permanent regime load current

According to this idea, the model can be written as

$$\begin{aligned}
\dot{x} &= v_x \\
\dot{v}_x &= -\frac{b_a}{m} \cdot v_x \\
\dot{y} &= v_y \\
\dot{v}_y &= -g - \frac{b_a}{m} \cdot v_y - s_w \cdot [\frac{b}{m} \cdot v_y + \frac{k}{m}(y - \text{int}(h + 1 - x))]
\end{aligned}$$

where s_w is equal to 1 in the floor and 0 in the air. The function $\text{int}(h+1-x)$ gives the height of the floor at a given position (h is the height of the first step). Note that we are considering steps of $1m$ by $1m$.

The *state events* are produced when x and y verify the condition:

$$y = \text{int}(h + 1 - x) \tag{17}$$

The simulation model results then similar to the one shown in Figure 3 but without the implicit block. To use the QSS2-method, the quantized integrators and static functions must be just DEVS models as M_3 and M_4. The discrete model should receive the events with the derivatives of x and y and send events when the event condition is achieved (to calculate that, it just has to find the roots of a second degree polynomial).

The system was then simulated using parameters $m = 1$, $k = 100000$, $b = 30$, $ba = 0.1$, initial conditions $x(0) = 0.575$, $v_x(0) = 0.5$, $y(0) = 10.5$, $v_y = 0$ and a quantum of 0.001 in the horizontal position, 0.0001 in the vertical position and 0.01 in the speeds.

The first 10 seconds of simulation were completed after 2984 internal transitions at the integrators (39 at x, 5 at v_x, 2420 at y and 520 at v_y). The trajectories do not differ appreciably from what can be obtained with a fixed step high order method using a very small step size.

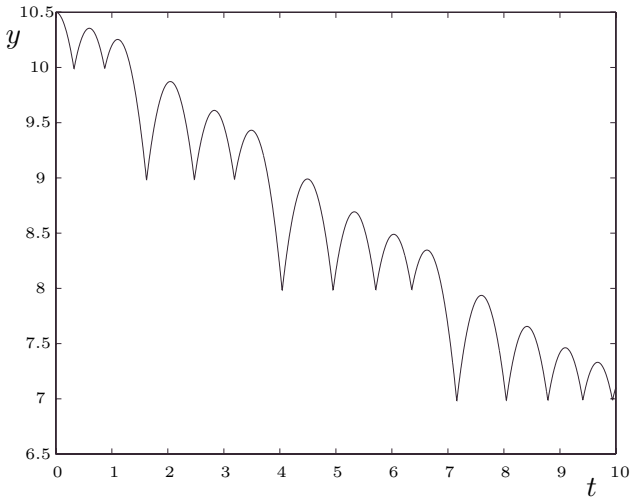

Figure 11: y vs. t in the bouncing ball example

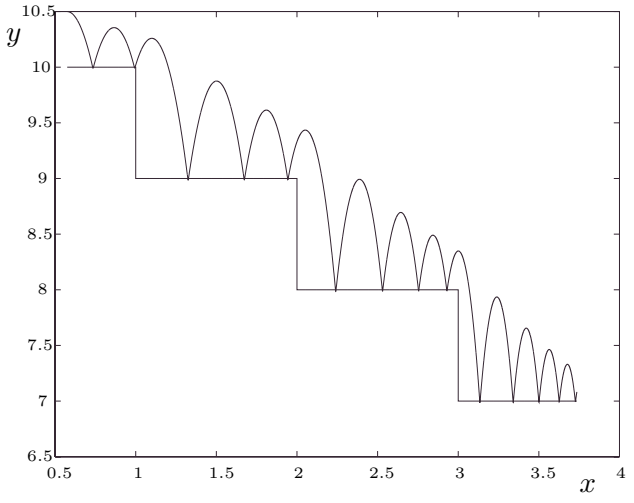

Figure 12: x vs. y in the bouncing ball example

Figures 11 and 12 show the simulation results.

It is important to remark that each step only involves very few calculations and the sparsity is well exploited. In fact, the internal transitions in x does not affect any other subsystem. The steps in v_x give events to itself, to the integrator which calculates x and to the discrete model which predicts the next event occurrence. Similarly, the internal events of y only provokes external events to the integrator corresponding to v_y when the ball is in the floor and finally, the events produced in v_y are propagated to itself, to the integrator which calculates x and to the discrete model.

As a result, the discrete model receives 525 events and it produces only 26 internal transitions (at the event occurrence times, it is, two events for each bounce).

The same model was simulated with Simulink, using different fixed and variable step algorithms. Obtaining a similar result with a fifth order fixed step method requires more than 10000 steps (and each step involves calculations over all the integrators).

When it comes to variable step methods, the best result using Simulink was obtained with the ode23s, which could obtain a similar result with about 5000 steps.

The problem of discrete time methods is that when they increment the step size, they start skipping events as shown in Figure 13. An example of this problem was given in (Esposito et al., 2001) where the authors gave a solution based on decreasing the step size as well as the system approximates the discontinuity condition.

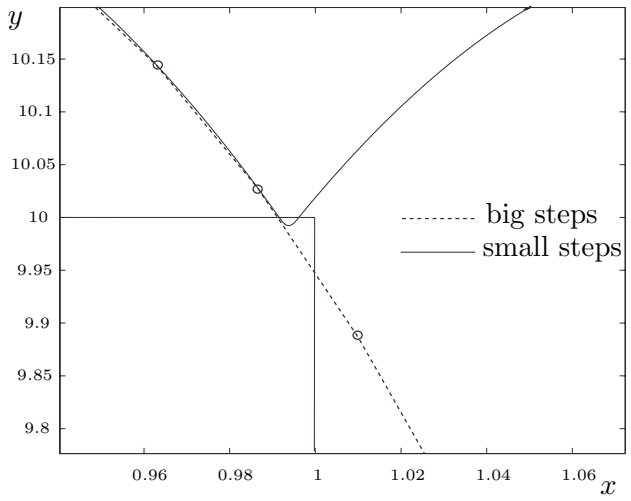

Figure 13: Event skipping in discrete time algorithms

The quantization-based approach does not modify anything. It just makes use of a discrete block which exactly predicts when the next event will occur and then produce an event at that time. The rest of the DEVS models (quantized integrators, static and implicit functions) work without taking into account the presence of discontinuities but they receive the events coming form the discrete part and treat them as if they were coming from an input generator or another quantized integrator. As a consequence, there are not extra calculations and there is no need of modifying anything.

5 Conclusions

The use of the the DEVS formalism and the QSS and QSS2–methods offers an efficient and very simple alternative for the simulation of hybrid systems. The facilities to deal with discontinuities constitutes one of the most important advantages of the methodologies with respect to classic discrete time algorithms.

In the examples analyzed, the methods showed a performance clearly superior to all the complex implicit, high order and variable step methods implemented in Simulink. Taking into account that QSS as well as QSS2 are very simple, low order and explicit algorithms with fixed quantization size; it is quite natural to think that future more complex discrete event methods may offer an unexpected high performance.

Adding to this the advantages observed not only in discontinuous systems, but also in DAE and in simple continuous systems, and taking into account their theoretical properties; it can be claimed that discrete event methods will constitute soon an interesting alternative to the classic methods for general purpose system simulation.

With respect to future work, it comes to be necessary to extend the theoretical stability and error bound analysis to general discontinuous systems in order to establish conditions which ensure the correctness of the simulation. What was done in the DC-AC inverter example might constitute a first step in this direction which could be extended for general systems with time events.

In the approach presented, only hybrid systems whose continuous part does not change its order were considered. It would be interesting to consider also more general cases including variable order.

Finally, in the bouncing ball example, the use of a bigger quantum in the position while the ball was in the air would have resulted in an important reduction of the number of calculation without affecting the error. This is a problem related to the use of fixed quantization. But there is another problem connected to the fact that the quantum has to be chosen. Although there are some practical rules and even theoretical formulas, finding the appropriate quantum is not an easy task.

These observations lead to the convenience of using some kind of adaptive quantization. If such a result can be obtained together with the use of higher order approximations (a third order approximation QSS3 could be easily imagined) the quantization–based approximations may become in a really powerful tool for the simulation of general hybrid systems.

References

Barton, P. (2000). Modeling, Simulation, and Sensitivity Analysis of Hybrid Systems: Mathematical Foundations, Numerical Solutions, and Sofware Implementations. In *Proc.of the IEEE International Symposium on Computer Aided Control System Design*, pages 117–122, Anchorage, Alaska.

Branicky, M. (1994). Stability of Switched and Hybrid Systems. In *Proc. 33rd IEEE Conf. Decision Control*, pages 3498–3503, Lake Buena Vista, FL.

Broenink, J. and Weustink, P. (1996). A Combined–System Simulator for Mechatronic Systems. In *Proceedings of ESM96*, pages 225–229, Budapest, Hungary.

Cellier, F. (1979). *Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools.* PhD thesis, Swiss Federal Institute of Technology.

Cellier, F. (1996). Object-Oriented Modeling: Means for Dealing With System Complexity. In *Proc. 15th Benelux Meeting on Systems and Control*, pages 53–64, Mierlo, The Netherlands.

Esposito, J., Kumar, V., and Pappas, G. (2001). Accurate Event Detection for Simulating Hybrid Systems. In *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 204–217. Springer.

Gear, C. (1971). The Simulataneous Numerical Solution of Differential–Algebraic Equations. *IEEE Trans. Circuit Theory*, TC–18(1):89–95.

Khalil, H. (1996). *Nonlinear Systems*. Prentice-Hall, New Jersey, 2nd edition.

Kofman, E. (2001). Quantized-State Control. A Method for Discrete Event Control of Continuous Systems. Technical Report LSD0105, LSD-UNR. To appear in *Latin American Applied Research* Journal. Available at www.eie.fceia.unr.edu.ar/∼ekofman/.

Kofman, E. (2002a). A Second Order Approximation for DEVS Simulation of Continuous Systems. *Simulation*, 78(2):76–89.

Kofman, E. (2002b). Non Conservative Ultimate Bound Estimation in LTI Perturbed Systems. In *Proceedings of AADECA 2002*, Buenos Aires, Argentina.

Kofman, E. (2002c). Quantization–Based Simulation of Differential Algebraic Equation Systems. Technical Report LSD0204, LSD, UNR. Submitted to Simulation.

Kofman, E. (2002d). Quantized-State Control of Linear Systems. In *Proceedings of AADECA 2002*, Buenos Aires, Argentina.

Kofman, E. and Junco, S. (2001). Quantized State Systems. A DEVS Approach for Continuous System Simulation. *Transactions of SCS*, 18(3):123–132.

Kofman, E., Lee, J., and Zeigler, B. (2001). DEVS Representation of Differential Equation Systems. Review of Recent Advances. In *Proceedings of ESS'01*.

Otter, M. and Cellier, F. (1996). *The Control Handbook*, chapter Software for Modeling and Simulating Control Systems, pages 415–428. CRC Press, Boca Raton, FL.

Park, T. and Barton, P. (1996). State Event Location in Differential-Algebraic Models. *ACM Trans. Mod. Comput. Sim.,*, 6(2):137–165.

Schlegl, T., Buss, M., and Schmidt, G. (1997). Development of Numerical Integration Methods for Hybrid (Discrete-Continuous) Dynamical Systems. In *Proceedings of Advanced Intelligent Mechatronics*, Tokio, Japan.

Taylor, J. (1993). Toward a Modeling Language Standard for Hybrid Dynamical Systems. In *Proc. 32nd IEEE Conference on Decision and Control*, pages 2317–2322, San Antonio, TX.

Taylor, J. and Kebede, D. (1996). Modeling and Simulation of Hybrid Systems in Matlab. In *Proc. IFAC World Congress*, San Francisco, CA.

Zeigler, B. (1976). *Theory of Modeling and Simulation*. John Wiley & Sons, New York.

Zeigler, B., Kim, T., and Praehofer, H. (2000). *Theory of Modeling and Simulation. Second edition.* Academic Press, New York.

Zeigler, B. and Lee, J. (1998). Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment. In *SPIE Proceedings*, pages 49–58.