DISCRETE EVENT SIMULATION OF HYBRID SYSTEMS

ERNESTO KOFMAN*

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– is notably simplified. 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 the presence of discontinuities. Finally, some examples which illustrate the use and the advantages of the methodology in hybrid systems are discussed.

Key words. hybrid systems, ordinary differential equation simulation, differential algebraic equations, discrete event systems, quantized systems.

AMS subject classifications. 65L05, 65L80, 34A36, 93C55

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 ordinary differential equations (ODEs) like Euler, Runge–Kutta, Adams, etc., several variable–step and implicit ODE solvers were developed. These modern methods –which usually make use of iteration rules and symbolic manipulation– allow the efficient simulation of complex models, including differential algebraic equation (DAE) and variable structure systems.

Although there are several differences among the mentioned ODE solvers, all of them share a property: they are based on time discretization and give a solution obtained from a difference equation system (i.e., a discrete time simulation model).

A completely different approach for ODE numerical simulation has been developed since the end of the 1990s. Here, time discretization is replaced by state variables quantization, arriving at discrete event simulation models within the DEVS formalism framework.

DEVS [26] is a formalism which allows one to represent and simulate any system with a finite number of changes in a finite interval of time. This way, systems modeled 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 [27]. This idea was reformulated with the addition of hysteresis, and it was formalized as a simulation method for ODEs in [15], where the quantized state systems (QSS) were defined.

This new method was improved with the definition of the second order quantized state systems (QSS2) [9] and then extended to the simulation of DAEs [12]. In all the cases, the idea is to modify the continuous system with the addition of quantizers and then to represent and simulate the resulting system by an equivalent DEVS model.

Despite their simplicity, the QSS and QSS2 methods satisfy strong stability, convergence, and error bound properties. From the computational cost point of view, the

*Laboratorio de Sistemas Dinámicos - FCEIA - Universidad Nacional de Rosario. Riobamba 245 bis, (2000) Rosario, Argentina. (kofman@fceia.unr.edu.ar).

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.

The simulation of hybrid systems and discontinuity handling have been always a problem for classical discrete time methods. The problem is that numerical integration algorithms in use are incompatible with the notion of discontinuous functions [17], and an integration step which jumps along a discontinuity may produce unacceptable errors.

To avoid this, the methods should perform steps in the instants of time in which the discontinuities occur. Thus, the solvers should be provided with tools for detecting discontinuities (what include iterations and extra computational costs), for adapting the step size to hit those instants of time, and for simulating 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 simulate hybrid systems in a quite efficient way, none of them can escape from these problems.

This work attempts to show that all the mentioned difficulties are considerably simplified with the use of the QSS and QSS2 methods. On one hand, the DEVS formalism and its closure under coupling 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 modifying the methodologies. In fact, all what has to be done is to connect a subsystem representing the discrete dynamics to the subsystem corresponding to the QSS or QSS2 approximation. In that way, all the qualities of the methods are conserved (error control, reduction of computational costs, 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, three hybrid examples are introduced and simulated with the QSS2 method. The results are compared with the results obtained with all the methods implemented in Matlab/Simulink in order to illustrate the advantages mentioned above.

2. DEVS and quantization–based methods.

2.1. Introductory example. Consider the ODE

$$\begin{aligned} \dot{x}_1(t) &= -x_1(t), \\ \dot{x}_2(t) &= x_1(t) + 2 \cdot x_2(t) \end{aligned} \tag{1}$$

and its *approximation*[1]

$$\begin{aligned} \dot{x}_1(t) &= -\text{int}[x_1(t)], \\ \dot{x}_2(t) &= \text{int}[x_1(t)) - 2 \cdot \text{int}(x_2(t)]. \end{aligned} \tag{2}$$

[1]Variables x_1 and x_2 are *local* to each system. In (2) $\text{int}(x_1)$ stands for the integer part of the state variable x_1 of that system, which is not connected to the solution of (1).

Equation (2) is nonlinear and discontinuous but can be easily and exactly solved. Consider, for instance, the initial condition $x_1(0) = 10.5$, $x_2(0) = 0$.

Initially we have that $\dot{x}_1 = -10$ and $\dot{x}_2 = 10$. These slopes will be constant until $t = 0.05$, when $\text{int}(x_1)$ becomes equal to 9. The new situation is $x_1 = 9.999\cdots$, $x_2 = 0.5$, $\dot{x}_1 = -9$, and $\dot{x}_2 = 9$.

The slopes will change again when x_2 reaches 1, i.e., when $t = 0.05 + 0.5/9 = 0.1056$. Now we have $\dot{x}_2 = 7$, but the slope in x_1 does not change.

This situation will remain the same until $x_1 = 8.999\cdots$. This situation is reached when $t = 0.05 + 1/9 = 0.1611$ and the new slopes are $\dot{x}_1 = -8$, $\dot{x}_2 = 6$, while $x_2 = 1 + 7 \cdot (0.1611 - 0.1056) = 1.3889$. The rest of the analysis continues in the same way.

Figure 1 draws the first steps of this solution with $q_1 \triangleq \text{int}(x_1)$ and $q_2 \triangleq \text{int}(x_2)$.

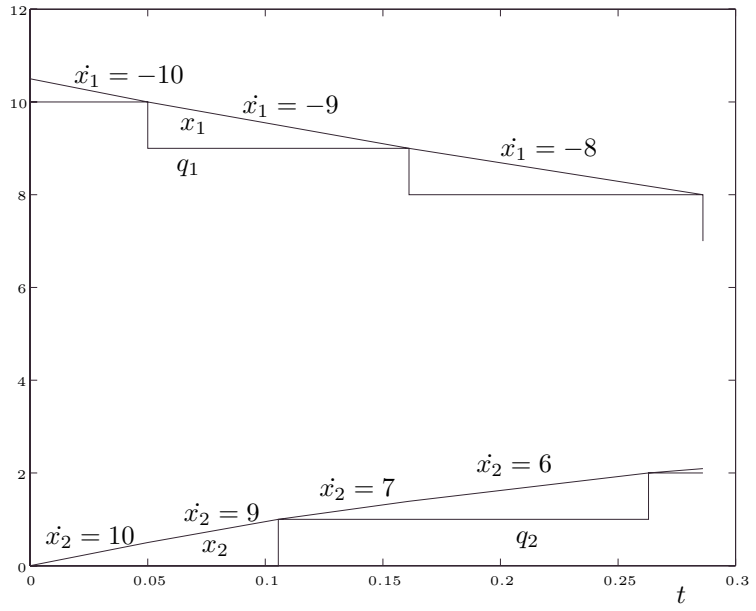

FIG. 1. *Solution of (2)*

What this example shows is that replacing x by $\text{int}(x)$ on the right–hand side of an ODE (in its state space representation), a kind of *numerical method* is obtained.

Up to this point, there were 3 steps, in $t = 0.05$, $t = 0.1056$, and $t = 0.1611$. The first and the third steps involved changes in the integer part of x_1 (which provoked changes in the slopes of x_1 and x_2), while in the second step there was a change in $\text{int}(x_2)$ which did not affect the slope of x_1.

Here, different steps involve different kind of calculations at each state variable. In that way, the behavior of this *numerical method* cannot be simply expressed by a difference equation such as standard integration algorithms.

In this *method* each state variable evolves following its own time, scheduling its next integer value change by a simple calculation using the fact that the slope is constant. When the slope changes (because some variable changes its integer value), the time to the next step (event) should be recalculated (rescheduled). In that way, it becomes clear that (2) behaves like a discrete event system.

Each step involves one discontinuity (introduced by the method itself) corresponding to a change of the integer part of one state variable. Thus, the simulation is in fact driven by discontinuities, and it is natural to expect this method to have some advantages to deal with discontinuous models.

The idea presented in this example (replacing x by $\text{int}(x)$) is a particular case of a quantized system [27], which does not work in general cases due to the presence of infinitely fast oscillations, as proven in [15]. However, a simple modification with the addition of hysteresis solves that problem.

It was already mentioned that (2) behaves like a discrete event system. The only problem is that it cannot be expressed in terms of Petri nets, state charts, grafcets, or other popular discrete event languages. That behavior can only be represented by a more general formalism like DEVS.

Although in this example we could state an algorithm with few steps to describe the method, in more complex cases (higher order systems, DAEs, and hybrid models) that algorithm would become very complicated. However, its representation in terms of DEVS is rather simple.

2.2. DEVS formalism. The DEVS formalism was developed by Zeigler in the mid–seventies [25, 26]. DEVS allows one 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.

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 the values that an input event can take;
- 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* $ta(s)$ ($ta(s) : S \to \Re_0^+$). The *time advance* is a nonnegative 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 $s_2 = \delta_{\text{int}}(s_1)$, where δ_{int} ($\delta_{\text{int}} : S \to S$) is called *internal transition function*.

When the state goes from s_1 to s_2 an output event is produced with value $y_1 = \lambda(s_1)$, where λ ($\lambda : S \to Y$) is called *output function*. 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*. Function δ_{ext} ($\delta_{\text{ext}} : S \times \Re_0^+ \times X \to S$) is called the *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 atomic ones.

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 quite direct way.

2.3. QSS method. Consider a time invariant ODE in its state equation system (SES) representation:

$$\dot{x}(t) = f[x(t), u(t)], \tag{3}$$

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 [15] simulates an approximate system, which is called quantized state system,

$$\dot{x}(t) = f[q(t), u(t)] \tag{4}$$

where $q(t)$ is a vector of *quantized variables* which are quantized versions of the state variables[2] $x(t)$. Each component of $q(t)$ is related to 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

$$q_i(t) = \begin{cases} Q_m & \text{if } t = t_0, \\ Q_{k+1} & \text{if } x_i(t) = Q_{k+1} \wedge q_i(t^-) = Q_k \wedge k < r, \\ Q_{k-1} & \text{if } x_i(t) = Q_k - \varepsilon \wedge q_i(t^-) = Q_k \wedge k > 0, \\ q_i(t^-) & \text{otherwise} \end{cases} \tag{5}$$

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.

[2]As before, the state variables are local to each system. There is no relation between $q(t)$ and the solution of (3).

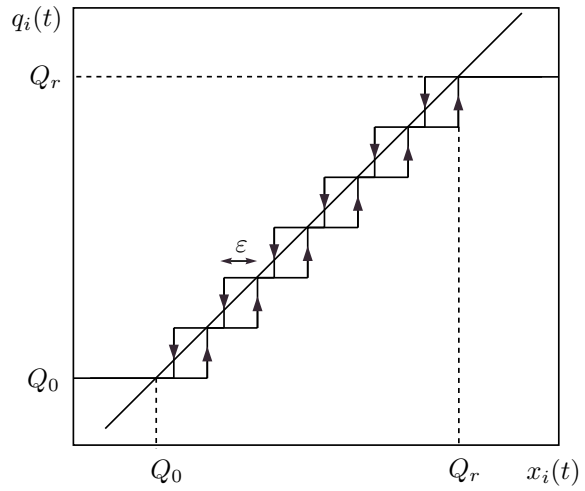

FIG. 2. *Quantization function with hysteresis*

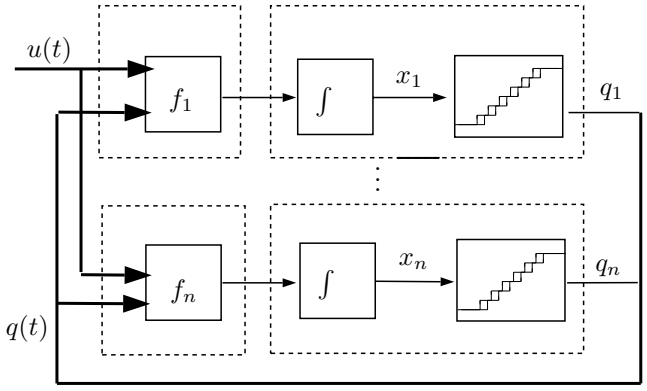

FIG. 3. *Block diagram representation of a QSS*

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 [15] it was proven that the quantized variable trajectories $q_i(t)$ are piecewise constant and the state variables $x_i(t)$ are piecewise linear. Based on that property, it was shown that QSS can be exactly simulated by DEVS models.

A generic QSS (4) 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.

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

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

where

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

and

$$
\sigma_2 = \begin{cases} \dfrac{Q_{k+1} - (x + e \cdot x_v)}{x_v} & \text{if } x_v > 0,\\[2ex] \dfrac{(x + e \cdot x_v) - (Q_k - \varepsilon)}{|x_v|} & \text{if } x_v < 0,\\[2ex] \infty & \text{if } x_v = 0. \end{cases}
$$

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

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

where

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

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.

The simulation of this coupled DEVS model will behave in the following way.

ALGORITHM 1.

1. The quantized integrator which had the minimum time–to–the–next–internal–transition performs it and provokes an output event giving its new quantized variable value q. During the transition this quantized integrator recalculates its new time–to–the–next–internal–transition (σ). This is always calculated as the time to the next change of the quantized variable.
2. The event carrying the output value reaches the static functions connected to the integrator mentioned in the previous step. These models provoke (instantaneously) output events with the new derivatives.
3. The events coming from the static functions arrive at the corresponding quantized integrators, which perform external transitions recalculating their σ.
4. The simulation time advances to the time–to–the–next–internal–transition of the quantized integrator in which it is minimum. Then the algorithm goes back to the first step.

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

2.4. QSS2 method. Although the QSS method satisfies nice stability, convergence and error bound properties –which will be recalled later– it performs only a first order approximation. Thus, the error reduction cannot be accomplished without an important increment in the number of steps.

This problem motivated the development of a second order approximation [9] which 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 (4), 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 greater than certain threshold. Figure 4 illustrates this idea.

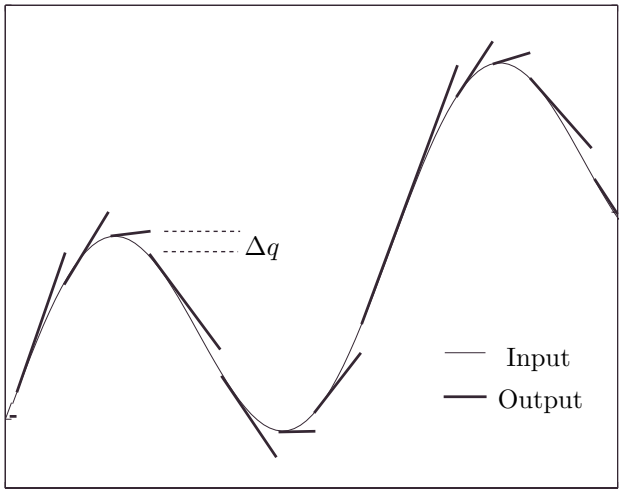

FIG. 4. *Input and output trajectories in a* first order *quantizer*

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

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

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

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

and the slopes

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

Here Δq plays the role of the quantum and hysteresis widths. In fact, in most applications of QSS they are equal to each other since this is the best choice taking into account the trade–off between error and computational costs [16].

The DEVS representation of QSS2 is also possible, but it is only exact in linear time invariant (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 an LTI system the quantized variable and state derivative trajectories are piecewise linear. Thus, the state variables have piecewise parabolic trajectories. In that way they can be represented by DEVS models, but the events should carry not only the new value of a trajectory but also the new slope.

The idea behind the DEVS model construction in QSS2 is similar to that of the QSS case. A coupled DEVS model like the one shown in Figure 3 is built, but the atomic models are redefined in order to obtain the new behavior.

The atomic 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

$$\sigma_1 = \begin{cases} \sqrt{\frac{2\Delta q}{m_u}} & \text{if} \quad m_u \neq 0, \\ \infty & \text{otherwise,} \end{cases} \tag{6}$$

and σ_2 can be calculated as the least positive solution of

$$|x + u \cdot e + \frac{m_u}{2}e^2 + v \cdot \sigma_2 + \frac{m_v}{2}\sigma_2^2 - (q + m_q \cdot e + m_q\sigma_2)| = \Delta q. \tag{7}$$

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

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

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

$$\tilde{c}_j = \begin{cases} \dfrac{f(z + m_z e) - f(\tilde{z})}{z_j + m_{z_j} e - \tilde{z}_j} & \text{if} \quad port = \text{inport}_j \wedge z_j + m_{z_j} e - \tilde{z}_j \neq 0, \\ c_j & \text{otherwise.} \end{cases} \tag{8}$$

If the function f is linear, this DEVS model exactly represents its behavior. Equation (8) –which calculates the coefficients c_j that multiply the input trajectory slopes– also allows approximating the nonlinear cases.

Here, the trajectory given by the DEVS model –which is interpreted as piecewise linear– constitutes a good approximation to the true output. The reason is that the coefficients c_j are closed to the corresponding partial derivatives of f evaluated at the points given by the input trajectories. Thus, the DEVS model of a static function can be applied to general nonlinear functions, and general nonlinear systems can be simulated under the QSS2 approach.

The simulation of a QSS2 has the same behavior expressed in Algorithm 1 in section 2.3. The only difference is that the events carry more information (value and slope) and that information is used by the quantized integrator to calculate their *sigma* in a different way.

At first glance, it may seem that QSS and QSS2 perform a kind of Runge–Kutta approximation. In fact, the way in which x is calculated in the transition functions of the quantized integrators suggests that. However, this is only true in a first order system where $u(t)$ is constant.

In higher order systems ($n \geq 2$) each state variable follows its own time steps and, between two steps, its derivative might be affected by the steps of the other variables.

In that way, the solution of QSS and QSS2 methods cannot be written as usual in the form $x(t_j + h) = x(t_j) + \cdots$ because there is one succession t_j for each state variable. Thus, even considering that $u(t)$ is constant and that f is continuous, the solution given by a QSS (QSS2) will be different from the solution given by any

Runge–Kutta or other discrete time approach, because in those methods the time steps are common to all the variables.

This *asynchronous* behavior is a distinctive property of QSS and QSS2, and this feature –more than the formulas involved in the calculations– provides the most important advantages in dealing with discontinuities.

To illustrate the differences between QSS2 and discrete time methods we shall solve the following system:

$$\begin{aligned} \dot{x}_1(t) &= -x_1(t), \\ \dot{x}_2(t) &= -x_2(t). \end{aligned} \tag{9}$$

The QSS2 method integrates the approximation

$$\begin{aligned} \dot{x}_1(t) &= -q_1(t), \\ \dot{x}_2(t) &= -q_2(t). \end{aligned} \tag{10}$$

We shall consider the initial conditions $x_1(0) = x_2(0) = 10$ and the quanta $\Delta q_1 = 0.2$, $\Delta q_2 = 0.1$.

Initially we have $q_1(t) = q_2(t) = 10$ and then $\dot{x}_1 = \dot{x}_2 = -10$. Thus, the state trajectories are $x_1(t) = x_2(t) = 10 - 10 \cdot t$.

The first quantized integrator (QI) schedules its first event for $t_{11} = 0.2/10$, while the second one does it for $t_{21} = 0.1/10$. Then the first step is performed by the second one at t=0.01. After that step we have[3] $q_2(t) = 9.9 - 10 \cdot (t - t_{21})$ which then results in $\dot{x}_2(t) = -9.9 + 10 \cdot (t - t_{21})$.

It follows that $x_2(t) = 9.9 - 9.9 \cdot (t - t_{21}) + 5 \cdot (t - t_{21})^2$, and then $|q_2(t) - x_2(t)|$ will become 0.1 when $t - t_{21} = 0.13177$. In that way, the next step at x_2 is scheduled for $t_{22} = 0.14177$.

This first step does not affect the first QI since $\dot{x}_1$ does not depend on q_2.

The second step is performed by the first QI at time $t_{11} = 0.02$. After this step it results that $q_1(t) = 9.8 - 10 \cdot (t - t_{11})$ and then $\dot{x}_1 = -9.8 + 10 \cdot (t - t_{11})$. Thus, we have $x_1(t) = 9.8 - 9.8 \cdot (t - t_{11}) + 5 \cdot (t - t_{11})^2$.

The next step in the first QI is then scheduled for the time $t_{12} = 0.201$, in which x_1 differs from q_1 in 0.2. This second step does not change the second QI because $\dot{x}_2$ does not depend on q_1.

The third step is then performed at the second QI at time 0.14177. This new step is similar to the first one (the first QI is not affected).

These first calculations show steps at $t = 0.01$, $t = 0.02$, and $t = 0.14177$. In those times we have $x_1(0.01) = 10$, $x_2(0.01) = 9.9$, $x_1(0.02) = 9.8$, $x_2(0.02) = 9.8015$, $x_1(0.1477) = 8.6807$, $x_2(0.1477) = 8.6823$.

Suppose now that system (9) is solved by any discrete time method with time steps at t_{21}, t_{11}, t_{22}, etc. In that routine both variables will be treated in the same way, and then the values of x_1 and x_2 will be identical to each other. Thus, the solution given by QSS2 (in which x_1 is different to x_2) cannot be obtained by any discrete time algorithm.

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

[3] $q_2(t) = x_2(t_{21}) + \dot{x}_2(t_{2_1}^-) \cdot (t - t_{21})$

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 (3).

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 [15] and [9], not only show theoretical features but also allow one to derive rules for the choice of the quantization.

Let us define $\Delta x(t) = q(t) - x(t)$. Then (4) can be rewritten as

$$\dot{x}(t) = f[x(t) + \Delta x(t), u(t)]. \tag{11}$$

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 differs from the original SES (3) only 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[4]).
- In stable and decoupleable LTI systems the QSS and QSS2 simulation trajectories never differ from the solutions of (3) in more than a bound, which can be calculated using a closed formula that depends on the quantum adopted (global 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.

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

Finally, the existence of a calculable global error bound is probably the most important property of quantization–based methods.

Given an LTI system

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{12}$$

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

$$|\tilde{\phi}(t) - \phi(t)| \leq |V||\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda||V^{-1}|\Delta q, \tag{13}$$

[4]In fact, we should not talk about stability. What we ensure is ultimate boundedness of the solutions [8].

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 or matrix and the symbol "$\leq$" in (13) means that each component of the error $|e(t)|$ is always less than or equal to the corresponding component of the bound calculated on the right–hand side of the inequality.

Inequality (13) 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 yields very important theoretical properties:

- the existence of a linear relationship between the global error bound and the quantum;
- the fact that the error is always bounded, and unstable results cannot be obtained in the simulation of LTI stable systems.

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 mentioned properties still hold. However, Equation (13) now becomes

$$|\tilde{\phi}(t) - \phi(t)| \leq |V||\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda||V^{-1}|\Delta q + |V||\mathbb{R}\mathrm{e}(\Lambda)^{-1}V^{-1}B|\Delta u, \tag{14}$$

where Δu is a vector with the quanta adopted in each input component. Inequality (14) can be easily deduced from [10].

2.6. QSS and QSS2 simulation of DAE systems. There are many continuous systems where an explicit ODE formulation cannot be easily obtained [5]. 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 (DAE) systems.

The basic idea for an efficient treatment of DAE systems consists in applying the ODE solver rules directly on the original DAE [7]. This is the idea followed to simulate DAE systems of index 1 with quantization-based methods[5] in [12].

A time invariant DAE can be written as

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

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

where $z(t)$ is a vector of algebraic variables. The vectors x_r and u_r are reduced versions of x and u, respectively.

Equation (15b) 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 (15) into

$$\dot{x}(t) = f[q(t), u(t), z(t)], \tag{16a}$$

$$0 = g[q_r(t), u_r, z(t)]. \tag{16b}$$

[5]Higher index problems have not been treated yet with QSS methods. Currently, the only alternative is to apply Pantelides' algorithm [18] to reduce the index before simulating. Anyway, the QSS method in some index 2 problems seems to adopt a simple switching behavior, as shown in [14].

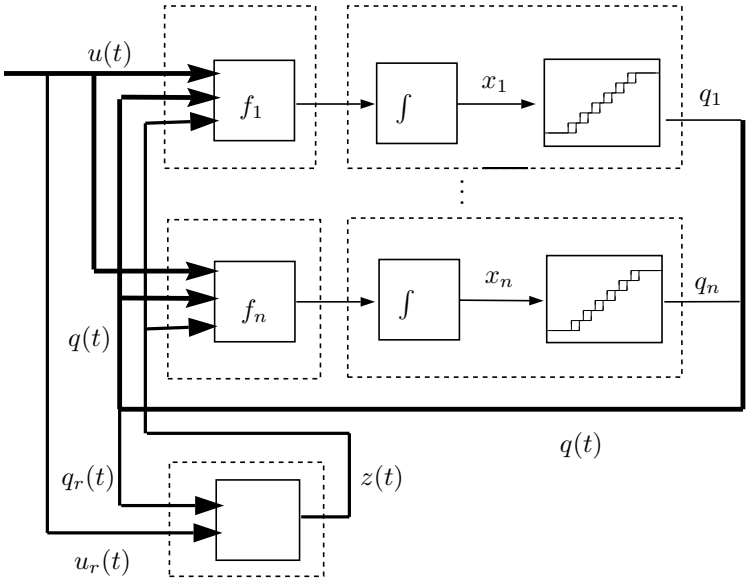

FIG. 5. *Coupling scheme for the QSS simulation of (15)*

Here, the iterations should only be performed to solve Eq.(16b) when the components of q_r or u_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 (16a) can be represented by quantized integrators and static functions, as 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 come from DEVS signal generators, the algebraic variables should be calculated by solving the restriction (16b). 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

$$g(v,z) = g(v_1,\ldots,v_m,z_1,\ldots,z_k) = 0 \tag{17}$$

for the QSS case can be written as

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

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's iteration or some other iteration rules to find the solution of (17) using an initial value z.

When the size of z (i.e., k) is greater than 1, the output events of model M_5 contain 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 [12].

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 the generation of incorrect event sequences, because the nonsmoothness violates the theoretical assumptions on which solvers are founded [1]. 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 has been being studied since Cellier's thesis [4], and many works can be found in the recent literature (see, for instance, [19, 24, 20, 6]).

Although the event detection introduces considerable improvements in simulation, it requires performing iterations, and it modifies the way in which the solvers work. Moreover, the event scheduling in the discrete subsystem must be simultaneously handled, and this fact can add a new problem in the presence of complex discrete dynamics.

All these difficulties disappear with the use of quantization–based integration methods. On the 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 subsystem by a DEVS model.

3.1. Continuous subsystem 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 [23, 2, 3, 1]).

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

$$\dot{x}(t) = f[x(t), u(t), z(t), m(t)], \tag{18a}$$

$$0 = g[x_r(t), u_r(t), z(t), m(t)], \tag{18b}$$

$m(t)$ being 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 (18b) has a solution for each value of $m(t)$ (which implies that the system (18) always has index 1).

Independently of the way in which $m(t)$ is calculated, the simulation submodel 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 (18) into

$$\dot{x}(t) = f[q(t), u(t), z(t), m(t)], \tag{19a}$$

$$0 = g[q_r(t), u_r(t), z(t), m(t)] \tag{19b}$$

with the definitions introduced in (16). 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.

From the point of view of a quantized integrator or a static function, a change in $m(t)$ (a discontinuity) is seen exactly in the same way as a change in a quantized variable $q_i(t)$ (a step). Thus, each discontinuity adds only a single step to the continuous subsystem integration.

3.2. Discrete subsystem representation. One of the most important features of DEVS is its capability to represent all kinds of discrete systems. Taking into account that the continuous subsystem is approximated by a DEVS model, it is convenient to represent also the discrete behavior by another DEVS model. Thus, both DEVS models can be directly coupled to build a unique DEVS model which approximates the whole system.

In the presence of only time events, the DEVS model representing the discrete subsystem 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 QIs work, the events will be processed by the continuous part as soon as they come out from the generator without needing to modify 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 greater advantage: The state trajectories are perfectly known for all time. Moreover, they are piecewise linear or piecewise parabolic functions, and so 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 calculate the trajectory $m(t)$. Since the state trajectories are only known inside the QIs, 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 involves only the manipulation of the polynomial coefficients.

Using these ideas, the simulation model of a hybrid system like (18) using the QSS or QSS2 method will be a coupled DEVS with the structure shown in Figure 6.

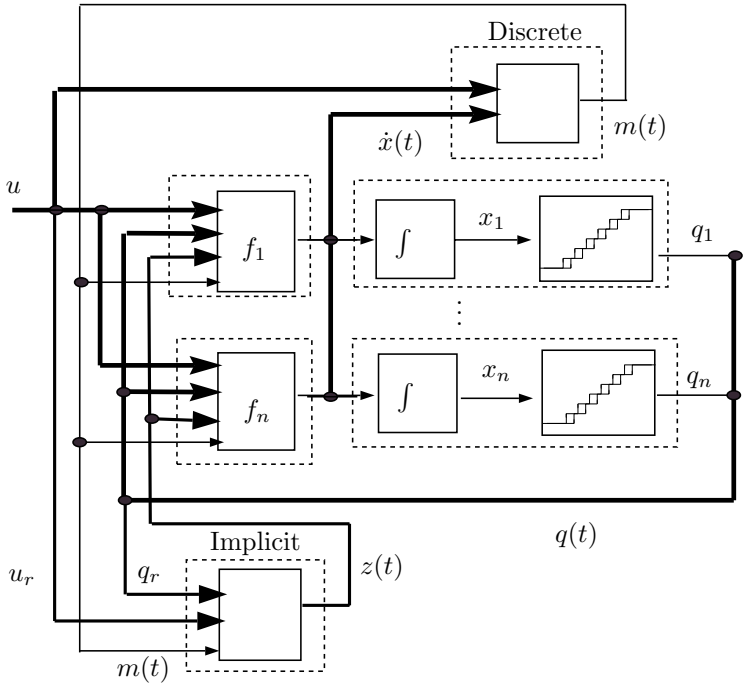

FIG. 6. *Coupling scheme for the QSS simulation of a hybrid system*

The discrete subsystem 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 produces 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 its dynamic is. Taking into account this remark, the scheme of Figure 6 can simulate any system like (18) in interaction with any discrete model.

There are cases in which this scheme can be simplified. As was mentioned before, in the absence of state events 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.

The simulation of the coupled DEVS model of Figure 6 behaves in a similar way to Algorithm 1. The only difference is that now, to calculate the minimum time–to–the–next–event, the discrete subsystem is also taken into account. When that subsystem performs the internal transition (i.e., when a model discontinuity takes

place), it provokes an event which carries the new value of $m(t)$. Thus, it is just processed as any other step (i.e., a change in the value of a QI).

In conclusion, the use of QSS and QSS2 simplifies the event location and permits treating each discontinuity as a single simulation step where no restart strategy is needed.

4. Examples and results. In order to show the uses and the advantages of quantization–based integration methods in hybrid systems, we shall introduce three simple examples. The first and the second are ruled by time events, while the last one contains state events.

Further examples of hybrid systems simulation with the QSS method can be found in [13] and [11], where continuous plants with asynchronous controllers were simulated.

4.1. DC—AC inverter circuit. Consider the inverter circuit of Figure 7.

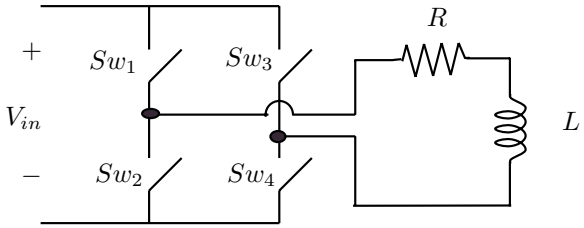

FIG. 7. *DC-AC full bridge inverter*

The set of switches can take two positions. In the first one switches 1 and 4 are closed and the load receives a positive voltage. In the second position switches 2 and 3 are closed and the load receives a negative voltage.

The system can be represented by the following differential equation:

$$\frac{d}{dt}i_L = \frac{1}{L}(-R \cdot i_L + s_w \cdot V_{in}), \tag{20}$$

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 the 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.

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.

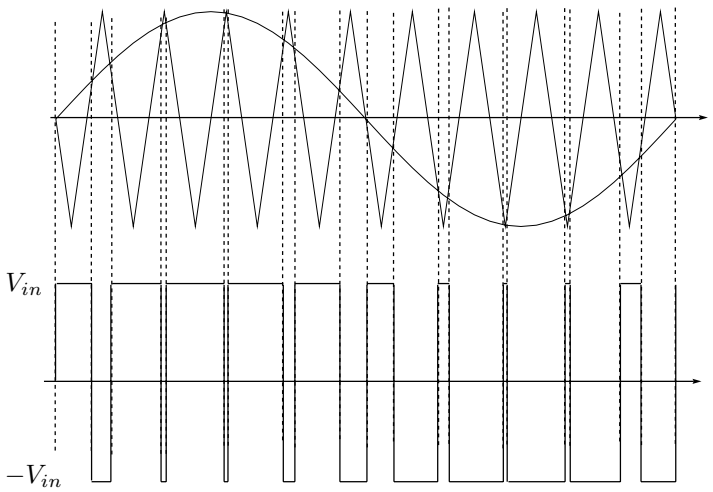

FIG. 8. *Pulse width modulation*

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

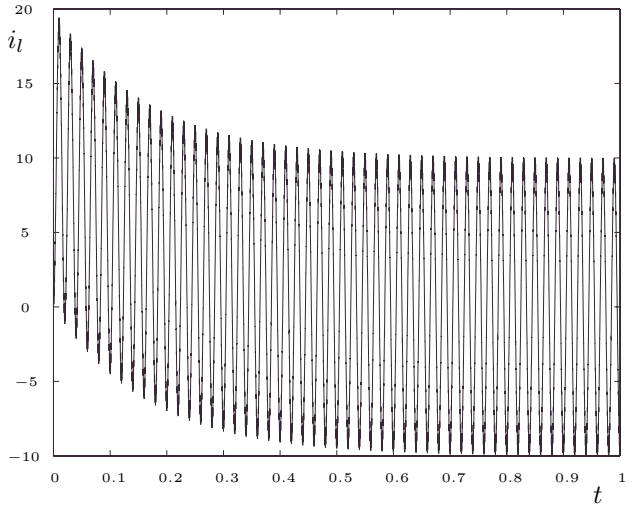

FIG. 9. *Load current with PWM*

The final time of the simulation was 1 second, and so 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 QI. Thus, the total number of steps was 6300.

In this case, since the commutations do not produce any structural change (they affect only the input voltage sign), (13) 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).

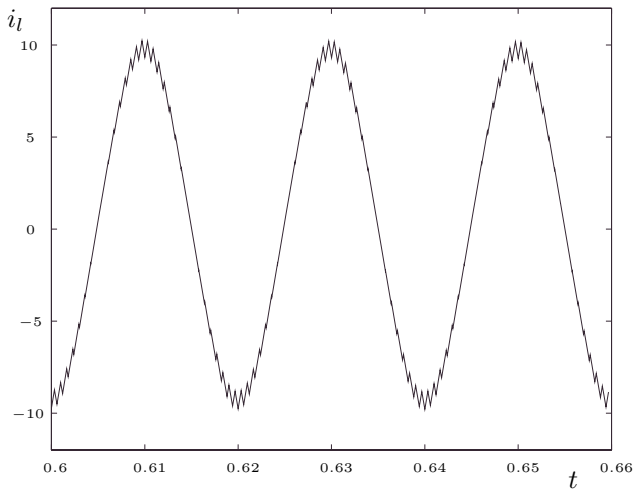

FIG. 10. *Detail of the permanent regime load current*

The same system was simulated with all the discrete time methods implemented in Simulink. The fixed step ode5 algorithm (5th order) needed more than 50,000 steps to obtain an acceptable result.

Using variable–step methods the result was even worse. Only ode23s[6] gave acceptable results with about 76,000 steps. The absolute and relative tolerances had to be reduced to 10^{-9} and 10^{-10} to obtain that result (the simulation with bigger tolerances gave wrong results).

The simulations were repeated with variable–step methods enforcing additional calculations at the event times. Although the results were significantly improved, the simulation required more than 20,000 steps using the same tolerance[7] as QSS2 (ode23 gave the best results now).

Anyway, this trick –enforcing calculations at predetermined time instants– cannot be used in general cases since the event times may not be 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.

The execution of the QSS2 simulation in this example takes about 8.5 milliseconds[8] on a 466MHz PC under MS Windows 98 using a Borland C++ compiler. The Matlab execution of the compiled ode23 (with 20,000 steps) takes 480 milliseconds on the same computer. Despite this quite unfair comparison (Matlab–compiled routines are not as efficient as flat c codes), the execution time of QSS2 is really fast.

In fact, each step of QSS2 involves only:

- two function evaluations (in higher order systems only some components should be evaluated);
- the actualization of the value of x (a second order polynomial evaluation);
- the calculation of the next step time (solving a quadratic equation).

[6]The methods used by Simulink are described in [21], and a description about the way they handle discontinuities can be found in [22].

[7]The absolute and relative tolerances were 10^{-2} and 10^{-3}

[8]The routine was run 1000 times in 8.5 seconds.

Also taking into account that it does not discard values and does not use adaptive rules, the number of calculations made in a single step of QSS2 is much less than the number of calculation involved in the steps of any other variable–step method.

4.2. DC—AC inverter with surge protection. The circuit of Figure 11 is a modification of the previous example. Here, a resistor R_p and a nonlinear component were included to limit the voltage of the load resistor R.

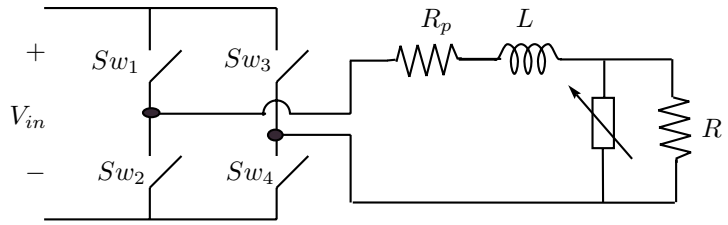

FIG. 11. *DC–AC inverter with surge protection*

It will be assumed that the nonlinear component has a varistor–like voltage–current relationship:

$$i(t) = k \cdot u(t)^{\alpha}.$$

Under this assumption the equation describing the system dynamics becomes a nonlinear DAE:

$$\frac{d}{dt} i_L = \frac{1}{L}(-R_p \cdot i_L - u + s_w \cdot V_{in}), \tag{21}$$

where u must verify

$$i_L - k \cdot u^{\alpha} - \frac{u}{R} = 0. \tag{22}$$

The parameters were chosen as in the previous example with $R_p = 0.01$, $k = 5^{-7}$, and $\alpha = 7$. The QSS2 simulation was completed again with 3100 internal transitions at the QI (i.e., 6300 steps). The output voltage (u_l) trajectory is shown in Figures 12–13.

The iterations to solve Equation (22) were performed with the secant method. Since that equation does not depend on s_w the secant method was invoked only after the QI internal transitions (i.e., 3100 times). That way, the method performed only 9598 iterations during the whole simulation.

The simulation with ode23 enforcing calculations at the event times needed, as before, more than 20,000 steps. However, the difference with QSS2 is more noticeable now: ode23 has to solve Equation (22) in all the steps (i.e. 20,700 times), while QSS2 has to solve it only 3100 times.

4.3. A ball bouncing downstairs. A typical discontinuous example is the bouncing ball. 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 in the air –with the presence of friction– and a different model in the floor (spring–damper).

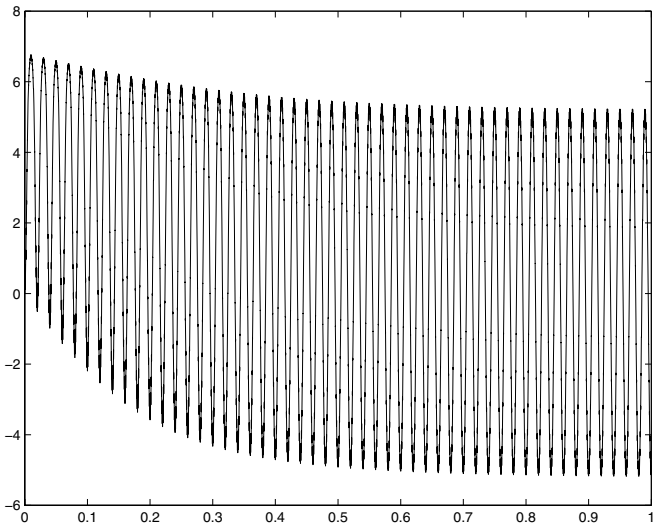

FIG. 12. *Load voltage with surge protection*

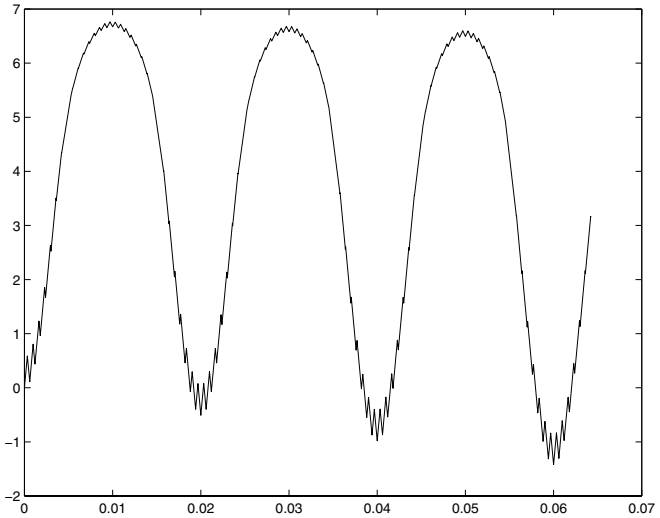

FIG. 13. *Load voltage with surge protection (startup)*

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. Function $\text{int}(h + 1 - x)$ gives the height of the floor at a given position (h is the height of the first step and steps of $1m$

by $1m$ are considered).

The *state events* are produced when x and y verify

$$y = \text{int}(h + 1 - x). \tag{23}$$

The resulting QSS2 simulation model is similar to the one shown in Figure 3 (without the implicit block). The QIs and static functions are just DEVS models like M_3 and M_4. The discrete subsystem receives the events with the derivatives of x and y and sends events when the event condition is achieved (to calculate that, it just has to find the roots of a second order polynomial).

The system was simulated with parameters $m = 1$, $k = 100,000$, $b = 30$, $ba = 0.1$, initial conditions $x(0) = 0.575$, $v_x(0) = 0.5$, $y(0) = 10.5$, $v_y = 0$, and quanta 0.001 in the horizontal position, 0.0001 in the vertical position, and 0.01 in the speeds.

In this case, the quantum in each variable was chosen equal to the desired absolute error. In decoupled systems, Eq.(13) implies that the global error bound in each variable is equal to the quantum. Although this is not true in the bouncing ball example (the system is not diagonal and the equation is discontinuous), that rule approximately works in many cases.

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.

Figures 14 and 15 show the simulation results.

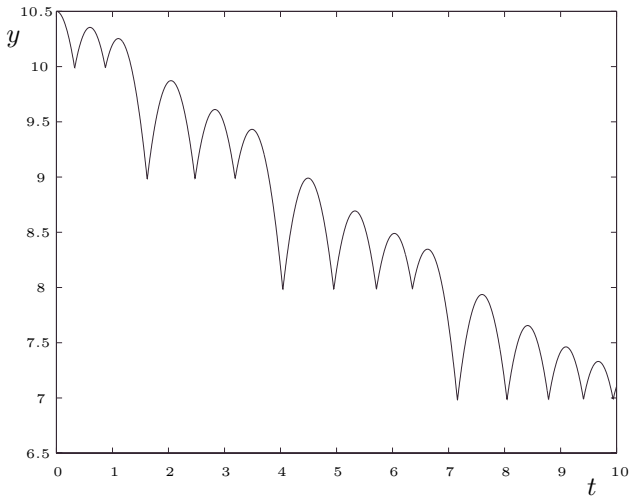

FIG. 14. *y vs. t in the bouncing ball example*

It is important to remark that each step involves very few calculations and the sparsity is well exploited. In fact, the internal transitions in x do 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 provoke only 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.

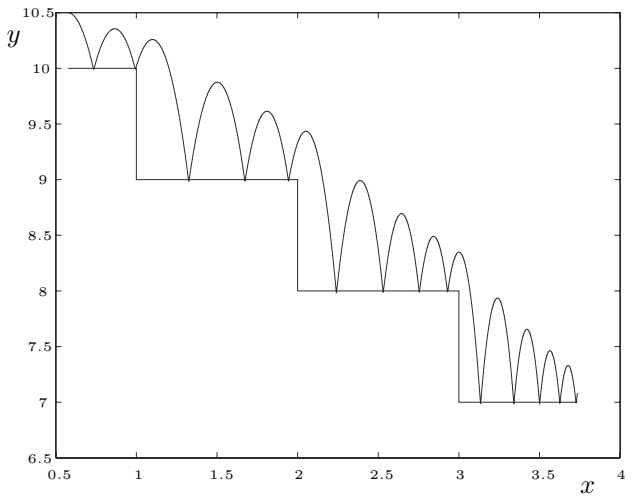

FIG. 15. *x vs. y in the bouncing ball example*

As a result, the discrete model receives 525 events and produces only 26 internal transitions (at the event occurrence times, i.e., two events per 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 10,000 steps (and each step involves calculations over all the integrators).

The best result was obtained with ode23s, which could obtain a similar trajectory with about 3000 steps. Despite the number of steps being almost the same, the calculations involved in each single step are much simpler in the QSS2 case.

The number of steps in ode23s is due to the need of a very small tolerance (the maximum absolute and relative tolerances were 1×10^{-9} and 7×10^{-7}, respectively). The use of bigger tolerances in this example (a relative tolerance of 8×10^{-7}, for instance) provokes event skipping problems for the chosen initial state, as shown in Figure 16.

An example of this problem was presented in [6], where the authors gave a solution based on decreasing the step size as well as the system trajectories approximate the discontinuity conditions.

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 produces an event at that time. The rest of the DEVS models (QIs, static and implicit functions) work without taking into account the presence of discontinuities, but they receive the events coming from the discrete part and treat them as if they were coming from an input generator or another QI. As a consequence, there are no extra calculations and there is no need of incorporating restarting routines or modifying the algorithm at all.

In this last example, the advantages of QSS2 are not as evident as in the previous one. In fact, QSS2 is quite limited because it is just a second order method and, in absence of discontinuities, does not work as well as other higher order algorithms. Thus, when there are not many events, what QSS2 improves with the efficient discontinuity handling is then lost during integration along the continuous part.

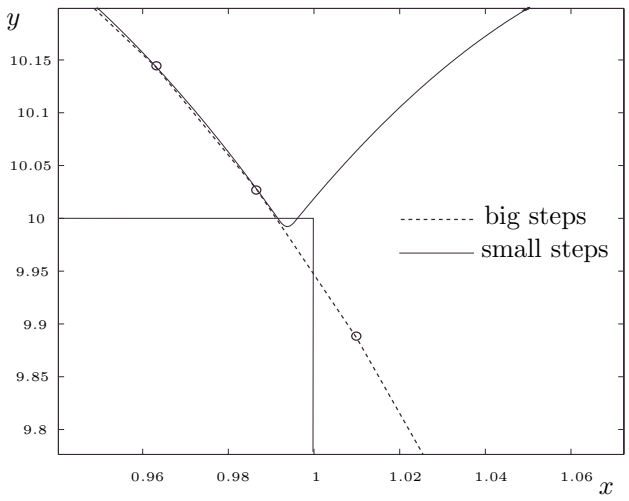

FIG. 16. *Event skipping in discrete time algorithms*

Anyway, there is another advantage in QSS and QSS2 connected to reliability. When the step size or the tolerance are increased in discrete time methods, we arrive at qualitatively wrong results (event skipping is just one example of this). However, QSS and QSS2 perform a correct simulation of the quantized system for any quantization adopted. When there are conditions which ensure the existence of some connection between the solution of the QSS(2) and the SES (as in the DC—AC inverter example) those qualitative incorrect results cannot be obtained in any case.

5. Conclusions. The use of DEVS and the QSS and QSS2 methods shows an efficient and very simple approach for the simulation of hybrid systems. The facilities to handle discontinuities constitute one of the most important advantages of these methodologies with respect to classic discrete time algorithms.

In the examples analyzed, the methods showed a superior performance to complex implicit, high order, and variable–step methods implemented in Simulink. Taking into account that QSS and QSS2 are very simple, low order, and explicit algorithms with fixed quantization size, it is promising to think what future more complex quantization–based methods might offer.

Besides these practical facts, the methods also show strong theoretical properties. In the DC—AC inverter example the result was good not only in terms of steps and accuracy but also in the existence of a calculated global error bound.

With respect to future work, it is necessary to extend the theoretical stability and error bound analysis to general discontinuous systems in order to establish conditions which ensure simulation correctness. 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 with fixed order continuous parts were considered. Variable–order cases can be easily taken into account and solved without introducing significant modifications to the methodology.

Finally, in the bouncing ball example, the use of a bigger quantum in the position while the ball is in the air would have resulted in an important reduction of the number of calculation without affecting the error (in this case, the number of transitions can

be reduced to approximately half).

This problem –related to the use of fixed quantization– is connected to another problem related to the quantum choice. Although there are practical and also theoretical rules, it is not easy to find the appropriate quantum.

These observations lead to the convenience of using logarithmic or even adaptive quantization. If such a result can be obtained together with the use of higher order approximations (a third order approximation QSS3 can be easily imagined), the quantization–based approximations may constitute an important alternative for the simulation of general hybrid systems.

REFERENCES

[1] P. Barton, *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, Anchorage, Alaska, 2000, IEEE, pp. 117–122.

[2] M. Branicky, *Stability of Switched and Hybrid Systems*, in Proc. 33rd IEEE Conf. Decision Control, Lake Buena Vista, FL, 1994, IEEE, pp. 3498–3503.

[3] J. Broenink and P. Weustink, *A Combined–System Simulator for Mechatronic Systems*, in Proceedings of the 10th European Simulation Multiconference (ESM96), Budapest, Hungary, 1996, Society for Computer Simulation, pp. 225–229.

[4] F. Cellier, *Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools.*, PhD thesis, Swiss Federal Institute of Technology, Zurich, Switzerland, 1979.

[5] ———, *Object-Oriented Modeling: Means for Dealing With System Complexity*, in Proc. 15th Benelux Meeting on Systems and Control, Mierlo, The Netherlands, 1996, pp. 53–64.

[6] J. Esposito, V. Kumar, and G. Pappas, *Accurate Event Detection for Simulating Hybrid Systems*, in HSCC, vol. 2034 of Lecture Notes in Computer Science, New York, 2001, Springer–Verlag, pp. 204–217.

[7] C. Gear, *The Simultaneous Numerical Solution of Differential–Algebraic Equations*, IEEE Trans. Circuit Theory, TC–18 (1971), pp. 89–95.

[8] H. Khalil, *Nonlinear Systems*, Prentice-Hall, Upper Saddle River, NJ, 2nd ed., 1996.

[9] E. Kofman, *A Second Order Approximation for DEVS Simulation of Continuous Systems*, Simulation, 78 (2002), pp. 76–89.

[10] ———, *Non Conservative Ultimate Bound Estimation in LTI Perturbed Systems*, in Proceedings of the Argentine Association of Automatic Control(AADECA 2002), Buenos Aires, Argentina, September 2002, AADECA, p. ID#87.

[11] ———, *Quantized–State Control of Linear Systems*, in Proceedings of the Argentine Association of Automatic Control(AADECA 2002), Buenos Aires, Argentina, September 2002, AADECA, p. ID#88.

[12] E. Kofman, *Quantization–Based Simulation of Differential Algebraic Equation Systems*, Simulation, 79 (2003), pp. 363–376.

[13] E. Kofman, *Quantized-State Control. A Method for Discrete Event Control of Continuous Systems.*, Latin American Applied Research, 33 (2003), pp. 399–406.

[14] E. Kofman and S. Junco, *Quantized Bond Graphs: An Approach for Discrete Event Simulation of Physical Systems*, in Proceedings of ICBGM'01, Phoenix, AZ, 2001, Society for Computer Simulation, pp. 369–374.

[15] ———, *Quantized State Systems. A DEVS Approach for Continuous System Simulation*, Transactions of the Society for Computer Simulation, 18 (2001), pp. 123–132.

[16] E. Kofman, J. Lee, and B. Zeigler, *DEVS Representation of Differential Equation Systems. Review of Recent Advances*, in Proceedings of the 2001 European Simulation Symposium (ESS '01), Marseille, France, 2001, Society for Computer Simulation, pp. 591–595.

[17] M. Otter and F. Cellier, *The Control Handbook*, CRC Press, Boca Raton, FL, 1996, ch. Software for Modeling and Simulating Control Systems, pp. 415–428.

[18] C. Pantelides, *The Consistent Initialization of Differential–Algebraic Systems*, SIAM Journal of Scientific and Statistical Computing, 9 (1988), pp. 213–231.

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

[20] T. Schlegl, M. Buss, and G. Schmidt, *Development of Numerical Integration Methods for*

Hybrid (Discrete-Continuous) Dynamical Systems, in Proceedings of Advanced Intelligent Mechatronics, Tokio, Japan, 1997, IEEE.

[21] L. Shampine and R. MW., *The MATLAB ODE Suite*, SIAM Journal on Scientific Computing, 18 (1997), pp. 1–22.

[22] L. Shampine and S. Thompson, *Event Location for Ordinary Differential Equations*, Computers and Mathematics with Applications, 39 (2000), pp. 43–54.

[23] J. Taylor, *Toward a Modeling Language Standard for Hybrid Dynamical Systems*, in Proc. 32nd IEEE Conference on Decision and Control, vol. 3, San Antonio, TX, 1993, IEEE, pp. 2317–2322.

[24] J. Taylor and D. Kebede, *Modeling and Simulation of Hybrid Systems in Matlab*, in Proc. IFAC World Congress, San Francisco, CA, July 1996, Elsevier Science.

[25] B. Zeigler, *Theory of Modeling and Simulation*, John Wiley & Sons, New York, 1976.

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

[27] B. Zeigler and J. Lee, *Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment*, in Proceedings of the SPIE 12th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls: Enabling Technology for Simulation Science, Proc. SPIE 3369, Bellingham, WA, 1998, pp. 49–58.