A THIRD ORDER DISCRETE EVENT METHOD FOR CONTINUOUS SYSTEM SIMULATION
Part II: Applications

Ernesto KOFMAN[†]

[†]*Laboratorio de Sistemas Dinámicos. FCEIA – UNR – CONICET.*
Riobamba 245 bis – (2000) Rosario – Argentina
Email: kofman@fceia.unr.edu.ar

***Abstract*— This work discusses practical issues and applications of the QSS3 method introduced in the companion paper. The method rules are translated into a discrete event model within the DEVS formalism and implemented in a simulation software. The method is then tested with the simulation of two relatively complex hybrid systems and the results are then compared with all Matlab/Simulink ODE solvers. These experiments show a noticeable reduction of the computational costs.**

***Keywords*— Hybrid systems, ODE integration, Discrete Event Systems.**

I INTRODUCTION

Quantization based methods for numerical integration of ordinary differential equations exhibit theoretical and practical features which, in some cases, constitute important advantages over classic numerical methods.

Starting from the idea of quantizing the state space instead of discretizing the time (Zeigler and Lee, 1998), the addition of hysteresis to the quantization resulted in the formalization of the first numerical method of this kind (Kofman and Junco, 2001). Performing a first order approximation, QSS showed some strong theoretical and practical qualities but it was limited in accuracy.

Its second order successor QSS2 (Kofman, 2002) partially solved this problem exhibiting now very important advantages in discontinuous systems (Kofman, 2004).

However, being a second order method, QSS2 sensibly increases the computational costs when an important accuracy is requested. Moreover, as the number of steps depends on the square root of the quantization, its election is still quite critical.

These facts motivated the development of a third order method called QSS3 that was introduced in the companion paper (Kofman, 2005). There, it was shown that the number of steps depends on the inverse of the cubic root of the quantum, which means that the quantum can be decreased without a significant increment of the computational costs. Thus, the accuracy can be improved and the quantization choice is not as critical as before.

This paper treats the practical issues of the QSS3 method. Like QSS and QSS2, this new method transforms a set of differential equations into a discrete event model within the DEVS formalism framework (Zeigler, 1976; Zeigler *et al.*, 2000). In the three cases, the resulting DEVS model can be divided in quantized integrators and static functions.

Taking into account the relationship between the piecewise parabolic trajectories at the input and output of quantized integrators and static functions deduced in (Kofman, 2005), the corresponding DEVS atomic models are built. After programming the resulting models in PowerDEVS (Pagliero and Lapadula, 2002), the QSS3 method can be applied in the same way than QSS or QSS2, i.e., building the system block diagram.

Finally, the method is used in the simulation of two complex hybrid examples and the results are compared with what is obtained using all the methods implemented in Matlab/Simulink. This will show that QSS3 method is a particularly efficient algorithm for accurate simulation of strongly discontinuous systems.

II QUANTIZATION AND DEVS

QSS, QSS2 and QSS3 methods produce simulation models which cannot be expressed by difference equations as classic discrete time algorithms. As it was already mentioned, the simulation models can be represented by DEVS.

A DEVS formalism

The DEVS formalism was developed by Zeigler in the mid–seventies (Zeigler, 1976; Zeigler *et al.*, 2000). 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* $ta(s)$ ($ta(s) : S \to \mathbb{R}_0^+$). 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} ($\delta_{\text{ext}} : S \times \mathbb{R}_0^+ \times X \to S$) 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.

B DEVS and Quantized State Systems

Given a set of state equations of the form

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1, \cdots, x_n, u_1, \cdots, u_m) \\ &\vdots \\ \dot{x}_n(t) &= f_n(x_1, \cdots, x_n, u_1, \cdots, u_m) \end{aligned}$$

The QSS method approximates it by

$$\begin{aligned} \dot{x}_1(t) &= f_1(q_1, \cdots, q_n, u_1, \cdots, u_m) \\ &\vdots \\ \dot{x}_n(t) &= f_n(q_1, \cdots, q_n, u_1, \cdots, u_m) \end{aligned} \tag{1}$$

where q_i and x_i are related by a *hysteretic quantization function* (Kofman and Junco, 2001). Consequently, q_i is piecewise constant.

We can think each equation in (1) as the coupling of two elementary subsystems. A *static* one

$$d_i(t) = f_i(q_1, \cdots, q_n, u_1, \cdots, u_m) \tag{2}$$

and the *dynamic*

$$q_i(t) = Q_i(x_i(\cdot)) = Q_i(\int d_i(\tau)d\tau) \tag{3}$$

where Q_i is the hysteretic quantization function (it is not a function of the point $x_i(t)$ but a functional over the trajectory $x_i(\cdot)$).

Provided that the components u_i are piecewise constant, the output of subsystem (2), i.e., d_i, is piecewise constant. Thus, both subsystems have piecewise constant input and output trajectories.

A piecewise constant trajectory has the form

$$v(t) = v_k \quad t_k \leq t < t_{k+1} \tag{4}$$

and it can be represented by a sequence of events with the value v_k at time t_k.

The subsystems (2) and (3) implicitly define a relationship between their equivalent input and output sequences of events. Thus, equivalent DEVS models to these subsystems can be found. These DEVS models are called *static functions* and *quantized integrators* respectively.

Similar remarks can be done with respect to the second order method QSS2. Here, as the trajectories are piecewise linear instead of piecewise constant, the events take the value in $\mathbb{R}^2$ with the initial value and the slope of each section of line.

The mentioned DEVS atomic models can be found in (Kofman, 2004).

III DEVS MODELS OF QSS3

In the companion paper (Kofman, 2005) it was proven that static functions (2) and quantized integrators (3) have piecewise parabolic input and output trajectories (in the nonlinear case the higher order terms were discarded to that goal).

There, the relationship between those inputs and outputs was deduced. Now, these relations will be expressed in terms of the DEVS formalism.

A Third order quantized integrator

The piecewise parabolic input trajectory of the quantized integrator[1] –$d(t)$– can be represented by a sequence of events with values in $\mathbb{R}^3$, a triplet (d_k, m_{d_k}, p_{d_k}).

The state variable, $x(t)$, is the integral of $d(t)$ and can be calculated from the previous sequence.

The quantized variable, $q(t)$, constitutes the output of the subsystem and, like the input, it can be characterized by a sequence of events (q_j, m_{q_j}, p_{q_j}).

According to the definition of a second order quantization function (Kofman, 2005), $q(t)$ starts a new parabolic section when it differs from $x(t)$ in Δq (see Fig.1). In those event times, $q(t)$ takes the value and the first and second derivative of $x(t)$.

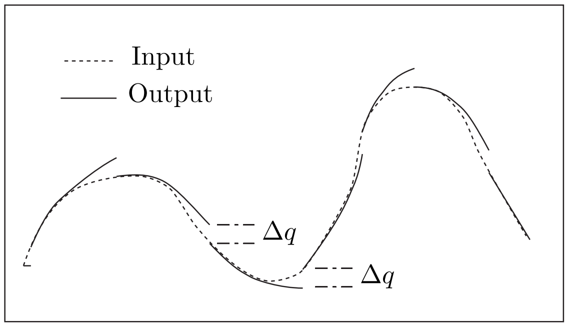

Figure 1: Input and output of a second order quantizer

The DEVS model was then built so that the state keeps the values of d, x and q. When d starts a new section of parabola (input events) or q starts a new section of parabola (output events), x, q and d are updated and variable σ (time advance, or time to next output event) is recalculated. Being the time to the next change in the quantized variable, σ is calculated as the elapsed time until q and x differ by Δq.

Calling s, $\tilde{s}$ and $\hat{s}$ to the current state, the state after an internal transition and the state after an external transition respectively, so that

$$\begin{aligned} s &= (d, m_d, p_d, x, q, m_q, p_q, \sigma) \\ \tilde{s} &= (\tilde{d}, \tilde{m_d}, \tilde{p_d}, \tilde{x}, \tilde{q}, \tilde{m_q}, \tilde{p_q}, \tilde{\sigma}) \\ \hat{s} &= (\hat{d}, \hat{m_d}, \hat{p_d}, \hat{x}, \hat{q}, \hat{m_q}, \hat{p_q}, \hat{\sigma}) \end{aligned}$$

the DEVS model can be expressed by

$$\begin{aligned} & M_{QI3} = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\ & X = \mathbb{R}^3 \times \{\text{inport}\}; \quad S = \mathbb{R}^7 \times \mathbb{R}_0^+ \infty, \\ & Y = \mathbb{R}^3 \times \{\text{outport}\}, \\ & \delta_{\text{int}}(s) = \tilde{s}; \quad \delta_{\text{ext}}(s, e, v, m_v, p_v, port) = \hat{s} \\ & \lambda(s) = (\tilde{q}, \tilde{m_q}, \tilde{p_q}, \text{outport}); \quad ta(s) = \sigma, \end{aligned}$$

[1]the sub–index i was eliminated to simplify the notation

with

$$\begin{aligned} \tilde{d} &= d + m_d\sigma + p_d\sigma^2, \quad \tilde{m_d} = m_d + 2p_d\sigma \\ \tilde{x} &= x + d \cdot \sigma + \frac{m_d}{2}\sigma^2 + \frac{p_d}{3}\sigma^3 \\ \tilde{q} &= \tilde{x}; \quad \tilde{m_q} = \tilde{d}; \quad \tilde{p_q} = \tilde{m_d} \\ \tilde{\sigma} &= \begin{cases} \sqrt[3]{\dfrac{3\Delta q}{\tilde{p_d}}} & \text{if} \quad \tilde{p_d} \neq 0, \\ \infty & \text{otherwise,} \end{cases} \end{aligned}$$

and

$$\begin{aligned} \hat{d} &= v; \quad \hat{m_d} = m_v; \quad \hat{p_d} = p_v \\ \hat{x} &= x + d \cdot e + \frac{m_d}{2}e^2 + \frac{p_d}{3}e^3 \\ \hat{q} &= q + m_q e + p_q e^2, \quad \hat{m_q} = m_q + p_q e \end{aligned}$$

Finally, $\hat{\sigma}$ is the least positive solution of

$$|\frac{p_v}{3}\sigma^3 + (\frac{m_v}{2} - \hat{p_q})\sigma^2 + (v - \hat{m_q})\sigma + \hat{x} - \hat{q}| = \Delta q \quad (5)$$

It can be easily seen that in this DEVS model x is the integral of d and q is the quantized version of x. Thus, this model is equivalent to a subsystem like (3) provided that d_i is piecewise parabolic.

B Static functions

Defining $v \triangleq (x_1, \cdots, x_n, u_1, \cdots, u_n)$, the linear version of (2) can be rewritten as

$$d_i = \sum_{j=1}^{N \triangleq n+m} c_j \cdot v_j \quad (6)$$

Since $v_j(t)$ are piecewise parabolic, they can be represented by sequences of events with values (v_{j_k}, $m_{v_{j,k}}$, $p_{v_{j,k}}$). As is was seen in (Kofman, 2005), the output $d_i(t)$ is also piecewise parabolic and the corresponding sequence of events (d_{i_k}, $m_{d_{i,k}}$, $p_{d_{i,k}}$) can be calculated with (6) using the fact that the derivatives satisfy the same equation.

The DEVS model keeps the input values v_j, m_{v_j}, p_{v_j} as well as the output d_i, m_{d_i}, p_{d_i}. The model has N input ports and when an input event arrives by the port j, the corresponding v_j, m_{v_j} and p_{v_j} take the input event value, while the rest of the input values are updated according to the elapsed time. The output values is then calculated with (6) and the time advance is set to 0 so that an output event occurs. After the corresponding internal transition, the time advance is set to ∞ so that no output event occur until the next event arrives.

Defining $s \triangleq (v, m_v, p_v, d, m_d, p_d, \sigma)$ the DEVS model can be expressed by

$$\begin{aligned} & M_{ST} = (X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where} \\ & X = \mathbb{R}^3 \times \{\text{inport}_1, \cdots, \text{inport}_N\}, \\ & S = (\mathbb{R}^{N+1})^3 \times \mathbb{R}_0^+ \infty, \quad Y = \mathbb{R}^3 \times \{\text{outport}\}, \\ & \delta_{\text{int}}(s) = (v, m_v, p_v, d, m_d, p_d, \infty) \\ & \delta_{\text{ext}}(s, e, u, m_u, p_u, port) = (\hat{v}, \hat{m_v}, \hat{p_v}, \hat{d}, \hat{m_d}, \hat{p_d}, 0) \\ & \lambda(s) = (d, m_d, p_d, \text{outport}), \quad ta(s) = \sigma, \end{aligned}$$

where, if $j = port$,

$$(\hat{v}_j, \hat{m}_{vj}, \hat{p}_{vj}) = (u, m_u, p_u)$$

and, when $j \neq port$

$$\begin{aligned} \hat{v}_j &= v_j + m_{v_j} e + p_{v_j} e^2 \\ \hat{m}_{vj} &= m_{v_j} + p_{v_j} e, \quad \hat{p}_{vj} = p_{v_j} \end{aligned}$$

The output is calculated as

$$\hat{d} = \sum_{j=1}^{N} c_j \hat{v}_j, \;\; \hat{m}_d = \sum_{j=1}^{N} c_j \cdot \hat{m}_{vj}, \;\; \hat{p}_d = \sum_{j=1}^{N} c_j \cdot \hat{p}_{vj} \tag{7}$$

This DEVS model can be also used for nonlinear functions. The only change that has to be made is in the calculations of (7), where the partial derivatives must be considered according to what was developed in (Kofman, 2005).

C Input signals

The incorporation of input signals in QSS3 does not differ from QSS and QSS2. The only difference is that now we are allowed to consider piecewise parabolic approximations, which can reduce the number sections (and the number of events) with respect to the piecewise constant and linear signals of QSS and QSS2.

Thus, the corresponding *signal sources* of QSS3 will be just DEVS generators which provoke events with the successive values of $u_i(t)$ (and their first and second derivatives).

D Discontinuity handling

Discontinuities can be managed in a similar way to QSS2. There, the discontinuity conditions were predicted by looking at the piecewise parabolic evolution of the state $x(t)$ (Kofman, 2004). Here, taking into account that a smaller quantization will be used, it is convenient to observe directly the quantized variable $q(t)$ in order to avoid solving a cubic equation.

Some examples including blocks with discontinuity detection capabilities will be discussed in Section IV.

E PowerDEVS implementation

The DEVS models described (quantized integrator, static functions and some generators and different discontinuous models) were programmed as new blocks of the simulator PowerDEVS (Pagliero and Lapadula, 2002) and organized in a new library called *qss3*. In that way, QSS3 simulations can be performed by using these blocks to draw the system block diagram.

The features of PowerDEVS permit to change the parameters like the quantum and initial state of the integrators and the coefficients of static functions by double clicking at the corresponding blocks.

Thus, the implementation of the QSS3 simulation becomes transparent to the end user of PowerDEVS.

The next section presents simulation results obtained with the mentioned library of PowerDEVS, showing also the way in which the software environment is used.

IV EXAMPLES

A DC motor with PWM control

We consider in this example a DC motor with constant field, described by the equations

$$\begin{aligned} \frac{dia}{dt} &= \frac{1}{L_a}(U_a(t) - R_a i_a - k_m \omega) \\ \frac{d\omega}{dt} &= \frac{1}{J}(k_m i_a - b_m \omega - \tau(t)) \end{aligned}$$

where $i_a(t)$ and $\omega(t)$ are the armature current and the angular speed of the motor.

The inputs $U_a(t)$ and $\tau(t)$ are the armature voltage and load torque. The parameters L_a, R_a, k_m, J and b_m are the armature inductance and resistance, the motor constant, the inertia, and the friction coefficient respectively.

A typical strategy to control the motor speed is called pulse width modulation. The motor speed is compared with a desired reference and then the armature voltage switches from a positive value $(+V)$ to a negative value $(-V)$ so that the duration of the resulting pulses is proportional to the error.

A way of achieving this is comparing the error with a triangular waveform (carrier wave) applying $+V$ or $-V$ according to the sign of the difference.

Using the parameters corresponding to a real DC motor: $L_a = 0.003$, $R_a = 0.05$, $k_m = 6.783$, $J_m = 15$, and $b_m = 0.005$ we simulated the response of the control system to a speed reference which goes from 0 to 60 with a rising time of 2 seconds. The DC motor is initially unloaded $(tau(0) = 0)$ and a step of 2500 is applied in $t = 3$. The triangular waveform was set with a frequency of $1000Hz$ and an amplitude of 1.1

The PowerDEVS model is shown in Fig.2. There, the blocks *QSS3 Integrator* and *QSS3 Linear* correspond to the third order quantized integrator and the linear static function described in the previous section. The *Triangular* and *Step* blocks are simple DEVS generators and the *Saturation* block bounds the output between 1 and -1 (so that the error does not becomes greater than the triangular wave, limiting in that way the maximum duty cycle).

The *SwitchTraj* block compares the error and the triangular wave, provoking events with values V or $-V$ when they become equal to each other. This block predicts when the trajectories cross using the fact that they are piecewise parabolic. Thus, the discontinuities are exactly detected and handled.

For the simulation, a quantum of 0.001 was used in both variables in order to appreciate the oscillations of the speed and current.

The QSS3 method completed the simulation of the first 5 seconds of the evolution after 7029 and 27572

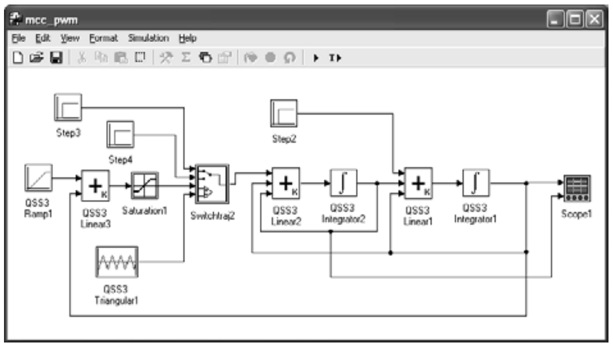

Figure 2: PowerDEVS model of a PWM control.

steps in the integrators which calculate ω and i_a respectively. Additionally, the block which produces the triangular waveform provoked 10000 events (i.e. 2 events by period during 5 seconds of simulation) and consequently, the switch produced 10000 commutations. In that way, there were a total of 54600 events. 34600 events corresponded to the continuous part and 20000 to the discrete part.

The simulation in PowerDEVS took 1.37 seconds on 450MHz PC under Windows 98.

The results are shown in Figures 3–5.

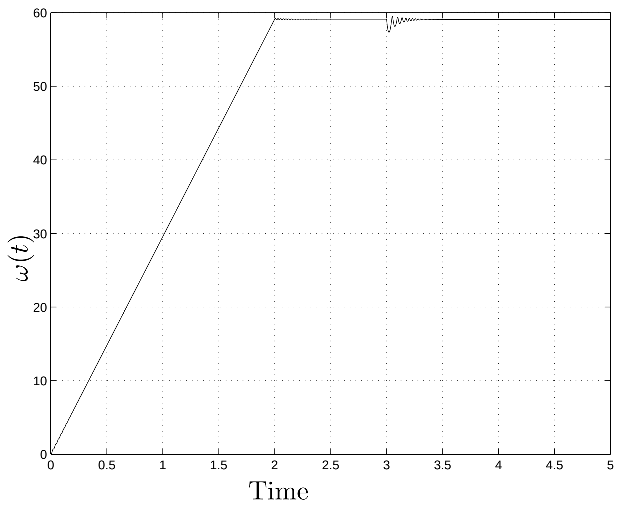

Figure 3: Motor speed.

The experiment was repeated using the QSS2 method with the same quanta. Now, the number of steps in the integrators was 17644 and 124021 in ω and $\imath_a$ respectively. The addition of the 10000 events in the discrete subsystem gives a total of 161665 events. PowerDEVS needed 3.18 seconds to simulate the system on the same computer than before.

We tried to simulate the same system with Simulink. The best results were obtained with ode23s. In order to get a qualitatively good result we needed to set the relative tolerance to $2 \cdot 10^{-10}$ and the absolute tolerance to 10^{-7}. The number of steps was 186593 and the simulation took 21.75 seconds.

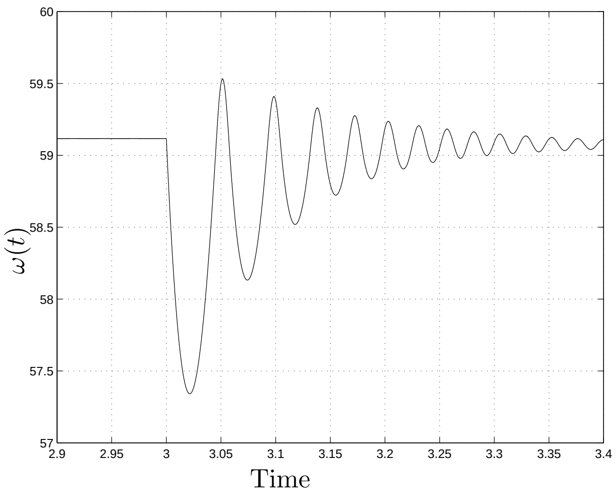

Figure 4: Motor speed (transient after the torque step).

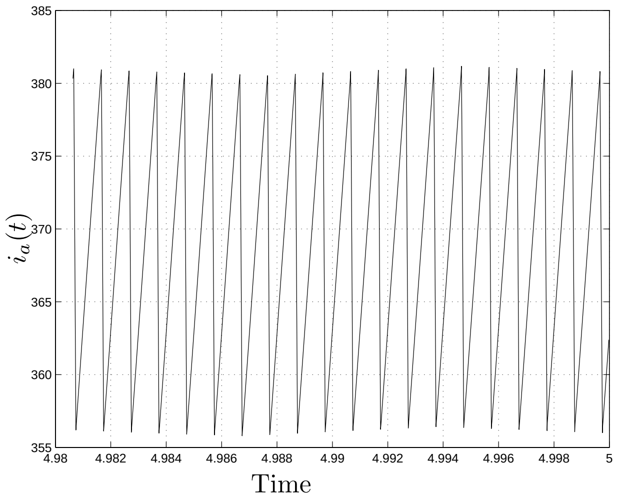

Figure 5: Armature current (final oscillations).

The number of steps performed by QSS3 was less than the third part of the steps of ode23s. Besides this, each step in QSS3 only involves a few calculations. The 10000 events given by the triangular wave generator are only seen by the switch model which decides when to apply $+V$ or $-V$.

Similarly, the 10000 events given by the switch are only seen by the integrator which calculates $i_a(t)$. The 27572 steps of the integrator which calculates $i_a(t)$ are only seen by itself and the other integrator. The only *expensive* events are the 7029 given by the integrator of $\omega(t)$ which are seen by both integrators and the discrete subsystem.

This clearly explains the fact that the simulation with PowerDEVS is much faster –about 15 times– than the simulation with Matlab.

B A ball bouncing downstairs

Consider a ball moving in two dimensions (x and y) bouncing downstairs. 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).

A possible model is given by the set of equations

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

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 commutations (*state events*) are produced when x and y verify $y = \text{int}(h+1-x)$.

The system was simulated in PowerDEVS with the QSS3 method. The block diagram was built in a similar way to the previous example.

In this case, a quantum of 0.00001 was chosen for the vertical position and 0.001 in the other variables. The initial conditions where $x(0) = 0.575$, $y(0) = 10.5$, $v_x(0) = 0.5$ and $v_y(0) = 0$ and the first 10 seconds of the system evolution were simulated.

The QSS3 method performed 464, 346, 10 and 6 steps in the integrators which calculate y, v_y, x and v_x respectively. Thus, there were a total of 826 steps. PowerDEVS took about 0.033 seconds to complete the simulation on a 450 MHz PC running under Windows 98. The results are shown in Fig.6.

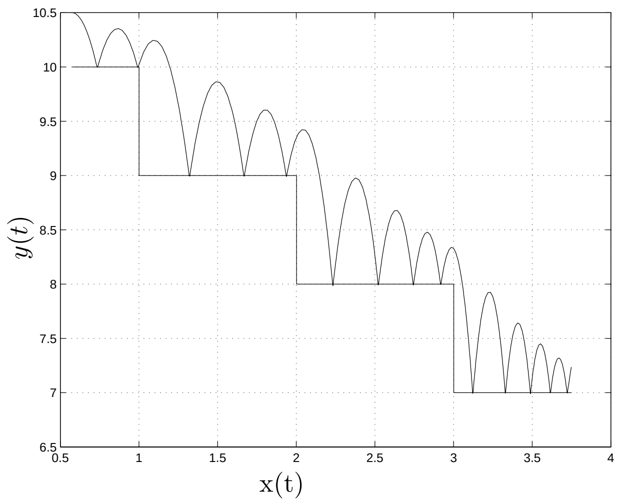

Figure 6: Ball bouncing downstairs (y vs. x).

The simulation with QSS2 and the same parameters takes a total of 9346 steps which are performed in 0.11 seconds on the same computer.

The simulation with variable step methods requires using a relative and absolute tolerance under 7×10^{-7}. Otherwise, the methods skip events. The best results with Matlab are obtained with the ode23s method which performs 2960 steps and takes 0.27 seconds (also on the same computer). Again, a noticeable reduction of simulation time was observed.

V CONCLUSIONS

The DEVS implementation and some simulation results of the QSS3 method were presented. The simulations showed that QSS3 is an efficient algorithm for accurate numerical integration of discontinuous systems. In the cases analyzed, the simulation time was drastically reduced with respect to what can be obtained with Matlab/Simulink.

Besides testing the method in more examples, future work should also consider the case of Differential Algebraic Equations, since it was only analyzed for QSS and QSS2 (Kofman, 2003).

QSS3 enlarged the family of quantization based integration methods. Although a fourth order method can be deduced from the same principles, the need of solving a fourth order equation might increase computational costs without improving considerably what QSS3 achieves.

REFERENCES

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

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

Kofman, Ernesto (2003). Quantization–Based Simulation of Differential Algebraic Equation Systems. *Simulation* **79**(7), 363–376.

Kofman, Ernesto (2004). Discrete Event Simulation of Hybrid Systems. *SIAM Journal on Scientific Computing* **25**(5), 1771–1797.

Kofman, Ernesto (2005). A Third Order Discrete Event Simulation Method for Continuous System Simulation. Part I: Theory. Technical Report LSD0501. LSD, UNR. Submitted to RPIC'05. Available at www.fceia.unr.edu.ar/∼kofman.

Pagliero, Esteban and Marcelo Lapadula (2002). Herramienta Integrada de Modelado y Simulación de Sistemas de Eventos Discretos. Diploma Work. FCEIA, UNR, Argentina.

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

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

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