Quantization–Based Simulation of Differential Algebraic Equation Systems
Technical Report LSD0203

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

Abstract

This paper extends the use of first and second order Quantized State Systems (QSS and QSS2) to be applied in the simulation of Differential Algebraic Equation (DAE) systems. A DEVS model related to a generic DAE is developed and a new method is given for a block–by–block translation from a Block Diagram containing algebraic loops into a coupled DEVS representation of its QSS and QSS2 approximations. We show that the main advantages provided by the quantization–based methods in the simulation of Ordinary Differential Equations (ODE) –stability properties, reduction of computational costs, sparsity exploitation, etc.– are still verified in DAE. These advantages are illustrated and discussed in the simulation of two examples which show the main features of the methodology.

Keywords: DEVS, ODE simulation, DAE simulation, Quantized State Systems.

1 Introduction

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

However, many continuous systems do not lead naturally to an explicit ODE model (Cellier, 1996). Indeed, there are cases in which that representation does not exist. These systems, where the state equations are implicitely defined, are called Differential Algebraic Equations. The difficulty carried by DAE simulation is connected to the need of using some special techniques which include iterations or symbolic manipulation to solve the implicit equations involoved.

One of the most important results in the area come from the idea of combining the ODE solver rules with the system implicit equations in order to compute them together (with iterations or symbolic manipulation). This idea, due to Gear (Gear, 1971), established the basis for all the modern DAE simulation methods.

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

Since the end of the 90's, a completely different approach for ODE numerical simulation has been being developed. In this new approach, the time discretization is replaced by the state quantization and a DEVS simulation model (Zeigler et al., 2000) is obtained instead of a discrete time one.

The origin of these methods can be found in the definition of Quantized Systems (Zeigler and Lee, 1998). This idea was reformulated with the addition of hysteresis and it was formalized as a simulation method for general ODEs in (Kofman and Junco, 2001b) where the Quantized State Systems were defined. This new method was then improved with the definition of the Second Order Quantized State Systems (Kofman, 2002a). Despite their simplicity, the QSS and QSS2 methods satisfy some stability, convergence and error bound properties which are only shared by complex implicit discrete time algorithms.

From the computational cost point of view, the quantization based methods also offer some advantages. They can reduce the number of calculations, their parallel implementation is straightforward and they exploit structural properties like sparsity in a very efficient fashion.

In this work, we study the use of QSS and QSS2 in the simulation of DAE systems. Following Gear's idea of combining the implicit system equations with the method rules, we introduce the concept of implicitely defined QSS and QSS2 and we develop generic DEVS models which simulate them.

It is shown that this approach yields an advantage in exploiting structural system properties: The iterations which solve the implicit equations do not have to be applied at each step. They are only performed after changes in the variables related by those equations.

It is a fact that many DAE systems come from Block Diagram models of continuous systems. In this case, the implicit equations can take the form of algebraic loops. Although these systems can be written as general DAEs and simulated with implicitelty defined QSS, we develop an idea which allows the direct implementation of quantization–based methods without any symbolic manipulation. This new technique translates a block diagram –which can contain algebraic loops– into a DEVS model which performs the QSS (or QSS2) simulation.

The paper is organized as follows: Section 2 recalls the QSS and QSS2 methods and their main properties. Then, in Section 3 the application of the quantization based methods to DAE systems is presented. There, we explain the concept of implicitely defined QSS and QSS2 and we show how those systems can be simulated by DEVS models. Further, in Section 4 we study the mentioned problem in Block Diagrams and we develop a general methodology for its direct QSS (QSS2) simulation. Finally, in Section 5 we show the use and the advantages of the methdology in the simulation of two illustrative examples.

2 Quantization–Based Methods

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

2.1 QSS–Method

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

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

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

The QSS–method simulates an approximate system, which is called Quantized State System:

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

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

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 $b : \Omega \rightarrow \Omega$ be a mapping and let $q_i = b(x_i)$ where the trajectory q_i satisfies:

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

and

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

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

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

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

Based on Figure 2, we can build the simulation model as the coupling of n atomic DEVS models representing the integrators and quantizers (or *quantized integrators*) and other n atomic DEVS models which simulate the behavior of the static functions f_i.

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

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

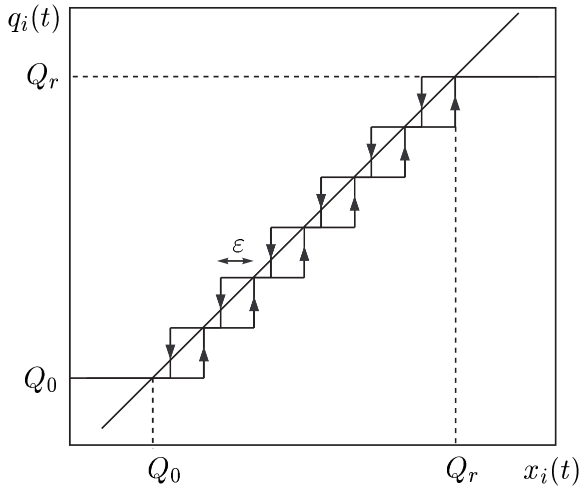

Figure 1: Quantization Function with Hysteresis

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

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

where

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

and

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

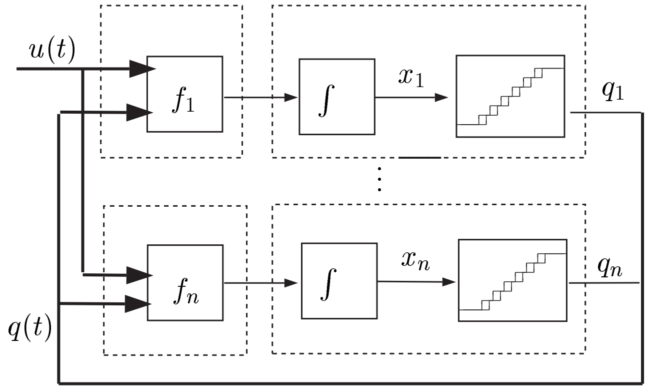

Figure 2: Block Diagram Representation of a QSS

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

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

where

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

Then, the QSS–method can be directly applied after choosing the quantization intervals and hysteresis width for each integrator and coupling the resulting DEVS models M_1 and M_2 according to Figure 2.

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

2.2 The QSS2-Method

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

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

The basic idea of QSS2 is the use of first–order quantization functions instead of the quantization function of Figure 1. Then, the simulation model can be still represented by (2) but now $q(t)$ and $x(t)$ have a different relationship. This new system is called Second Order Quantized State System or QSS2 for short.

The first–order quantization function can be seen as a function which gives a piecewise linear output trajectory, whose value and slope change when the difference between this output and the input becomes bigger than certain threshold. Figure 3 illustrates this idea.

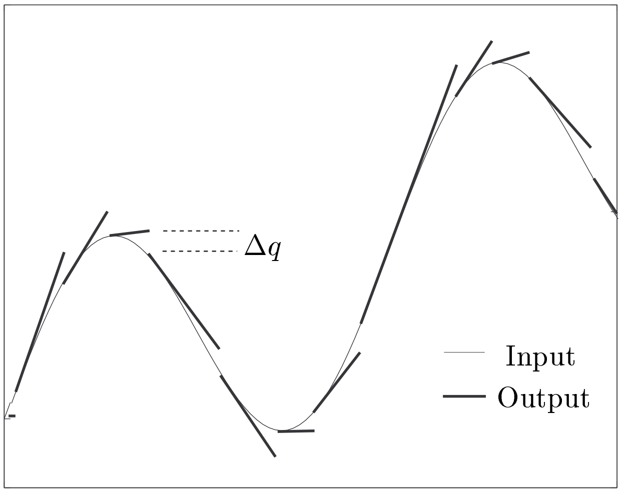

Figure 3: Input and Output trajectories in a *First Order* quantizer

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

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

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 are

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

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

QSS2 can be represented by DEVS, but the representation is only exact in LTI systems with piecewise linear input trajectories. The problem with nonlinear systems is that the trajectories are no longer piecewise linear after passing by a nonlinear function and then, the state derivative trajectories do not have any particualar form.

In a QSS2 coming from a LTI system the quantized variable and state derivative trajectories are piecewise linear and the state variable have piecewise parabolic trajectories. This knowledge allows building the DEVS model but now, the events should carry not only the new value of a trajectory but also its new slope.

Anyway, the idea behind the DEVS model construction is similar to the first order method. We represent it as a coupled DEVS model like the one shown in Figure 2, but the atomic models are redefined in order to obtain the new behavior.

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

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

where

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

and σ_2 can be calculated as the least positive solution of

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

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

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

with

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

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

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

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

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

2.3 Theoretical Properties of QSS and QSS2

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

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

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

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

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

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

Using 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 arbitrary small regions around the equilibrium points of that continuous system (Stability[2]).

- In stable and decoupleable LTI systems, the QSS and QSS2 simulation trajectories never differ from the true solutions in more than a bound which can be calculated using a closed formula which depends on the quantum adopted (Error Bound)

The Convergence Property ensures that an arbitrarily small error can be achieved by using an enough small quantization. A sufficient condition which guarantees 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 the choice of the quantum to be used in the different state variables. However, this is not very practical since it is based on a Lyapunov analysis and the algorithm requires the use of a Lyapunov function. Anyway, this is a very strong theoretical property since it holds for general nonlinear systems.

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

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

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

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

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

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

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

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

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

Finally, it should be mentioned that in most 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 appropiate quantizer. Although this input quantization introduces a new error, the properties we mentioned are still satisfied, but Equation (9) now becomes

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

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

[2]In order to be rigurous, we should not talk about stability. What we ensure is ultimate boundedness of the solutions (Khalil, 1996) since the quantized system trajectories usually finish in small oscillations.

3 QSS and QSS2 Simulation of DAE Systems

As we mentioned in the introduction, the basic idea for an efficient treatment of DAE system consists in applying the ODE solver rules directly on the original DAE.

A time invariant DAE can be written as

$$\tilde{f}(\dot{x}(t), x(t), u(t)) = 0 \tag{11}$$

Then, if we try to use the QSS or QSS2 method rules here, we should replace the state and input variables $x(t)$ and $u(t)$ by their quantized version $q(t)$ and $u_q(t)$. Thus, Equation 11 becomes

$$\tilde{f}(\dot{x}(t), q(t), u_q(t)) = 0 \tag{12}$$

Then, we can use iteration rules –like Newton– to obtain $\dot{x}$ from (12). In this work, we assume the index is 1 which means that we have enough equations to determine $\dot{x}$. If the index is higher, the Pantelides' algorithm (Pantelides, 1988) can be applied in order to reduce it to 1.

Once we have the state derivatives, they can be sent to the quantized integrators which perform the rest of the job, i.e. calculating the quantized variable trajectories. Each time a quantized variable changes, a new iteration process should be performed in order to recalculate $\dot{x}$.

It was already explained that QSS and QSS2 methods exploit the system sparsity reducing the computational costs. Now, we will show how we can make use of this in DAE problems.

Equation (12) is usually written as

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), z(t)) && \text{(13a)} \\ 0 &= g(x_r(t), u_r, z(t)) && \text{(13b)} \end{aligned}$$

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

Equation (13b) 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 tranforms (13) into

$$\begin{aligned} \dot{x}(t) &= f(q(t), u_q(t), z(t)) && \text{(14a)} \\ 0 &= g(q_r(t), u_{q_r}, z(t)) && \text{(14b)} \end{aligned}$$

and now, the iterations should be only performed to solve Eq.(14b) when the components of q_r or u_{q_r} change. When the dimension of x_r is significantly less than the dimension of x, i.e. when there are several state variables which do not influence on the loop, this fact represents an important advantage.

When Equation (14b) defines the value of z, we can see that system (14) defines something which behaves like a QSS or a QSS2. In fact, we can easily prove that the state and quantized variable trajectories correspond to QSS or QSS2. Moreover, the auxiliary variables z have piecewise constant and piecewise linear trajectories in QSS and QSS2 respectively.

Then, we say that the system (14) is an implicitely defined QSS or QSS2. What we have to say now is how an implicitely defined QSS (QSS2) like this can be translated into a DEVS model.

It is clear that (14a) can be represented by quantized integrators and static functions as we did in Section 2. The only difference now is the presence of the auxiliary variables z which act as inputs like

u_q. However, while the components of u_q are known and they may come from DEVS signal generators, the auxiliary variable values should be calculated by solving the restriction (14b).

Thus a new DEVS model shuld be built. This DEVS model must receive events with the values of q_r and u_{q_r} and then it has to calculate z. Figure 4 shows the new coupling scheme with the addition of a new DEVS model which calculates z.

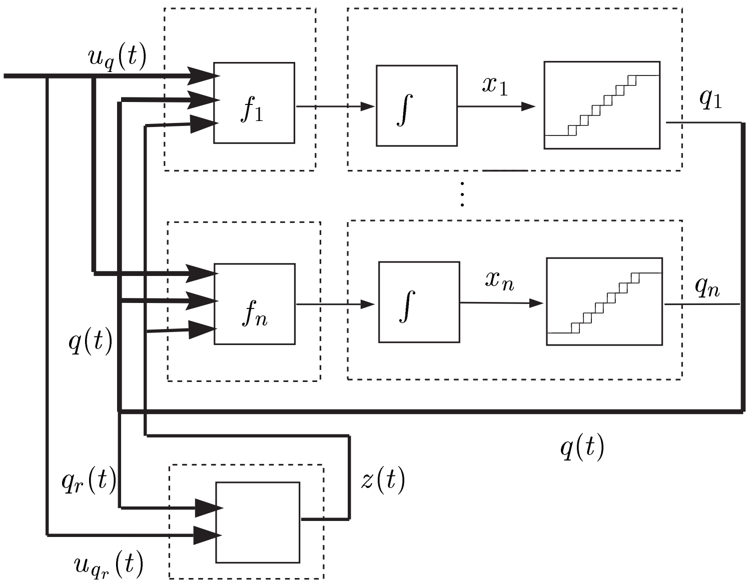

Figure 4: Coupling scheme for the QSS simulation of (13)

A DEVS model which solves a general implicit equation like

$$g(v, z) = g(v_1, \ldots, v_m, z_1, \ldots, z_k) = 0 \tag{15}$$

for the QSS case can be written as follows

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

where

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

and the function $h(v,z)$ returns the result of applying Newton iteration or some other iteration rules to find the solution of (15) using an initial guess z.

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

The idea for the QSS2–method is similar, but now the implicit equation should be rewritten as:

$$g[v(t),z(t)] = g[v_0 + m_v\Delta t, z_0 + m_z\Delta t] = 0 \tag{16}$$

which can be splitted into

$$g(v_0, z_0) = 0 \tag{17a}$$

and a first order approximation

$$\left.\frac{\partial g}{\partial v}\right|_{(v_0,z_0)} m_v + \left.\frac{\partial g}{\partial z}\right|_{(v_0,z_0)} m_z = 0 \tag{17b}$$

Then, the algorithm should iterate using Eq.(17a) to obtain z_0 and then, it must use this value in (17b) to calculate m_z.

The calculation of m_z can be done using symbolic manipulation (i.e. matrix inversion) or a new iteration process which will finish after two steps (because this is a linear equation). If the partial derivatives are not available, they can be estimated using coefficients as we did in Equation (6).

When z and $g(v,z)$ are scalar, Equation (17b) can be easily solved. In this case, we have

$$m_z = -\frac{\left.\frac{\partial g}{\partial v}\right|_{(v_0,z_0)} m_v}{\left.\frac{\partial g}{\partial z}\right|_{(v_0,z_0)}} \tag{18}$$

and then, a DEVS model can be built as follows

$M_6 = < X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta >$, where:
$X = \mathbb{R}^2 \times \{\text{inport}_1, ..., \text{inport}_m\}$
$S = \mathbb{R}^{3(m+1)} \times \mathbb{R}_0^+\infty$
$Y = \mathbb{R}^2 \times \{\text{outport}_1, ..., \text{outport}_m\}$
$\delta_{\text{int}}((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma) = ((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \infty)$
$\delta_{\text{ext}}((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma, e, u, mu, port) =$
$= ((\tilde{v}_1, \tilde{m}_{v_1}, \tilde{c}_1), ..., (\tilde{v}_m, \tilde{m}_{v_m}, \tilde{c}_m), (\tilde{z}, \tilde{m}_z, \tilde{c}_z), 0)$
$\lambda((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma) = (z, m_z, \text{outport})$
$ta((v_1, m_{v_1}, c_1), ..., (v_m, m_{v_m}, c_m), (z, m_z, c_z), \sigma) = \sigma$

with

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

and

$$\tilde{z} = h(\tilde{v}, z)$$

where h is the result of the iterations to solve Equation (17a) starting from the initial value z.

The coefficients which estimate the partial derivatives can be recalculated as:

$$\tilde{c}_j = \begin{cases} \dfrac{g(v + m_v e, \tilde{z})}{v_j + m_{v_j} e - \tilde{v}_j} & \text{if} \quad \mathit{port} = \text{inport}_j \wedge v_j + m_{v_j} e - \tilde{v}_j \neq 0 \\ c_j & \quad \text{otherwise} \end{cases} \tag{19}$$

$$\tilde{c}_z = \begin{cases} \dfrac{g(\tilde{v}, z + m_z e)}{z + m_z e - \tilde{z}} & \text{if} \quad z + m_z e - \tilde{z} \neq 0 \\ c_z & \quad \text{otherwise} \end{cases} \tag{20}$$

and finally, the new slopes are

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

and

$$\tilde{m}_z = -\frac{1}{\tilde{c}_z} \sum_{i=1}^{m} \tilde{m}_{v_i} \tilde{c}_i$$

If m is big, i.e. v has many components, the new output slope $\tilde{m}_z$ can be calculated with the equivalent formula:

$$\tilde{m}_z = \frac{c_z}{\tilde{c}_z} m_z + \frac{1}{\tilde{c}_z}(m_{v_j} c_j - \tilde{m}_{v_j} \tilde{c}_j)$$

where j is the index of the input port where the last event arrived, that is $\text{inport}_j = p$.

This last DEVS model is just a possible alternative to solve an implicit restriction like (16) taking into account the slopes. There are many other possibilities. When function g is linear, the DEVS model produces the exact output values z and m_z. Otherwise, it gives only a first order approximation.

4 Block–Oriented DEVS Simulation of DAE

As it was mentioned in the introduction, DAE systems of index 1 are often represented by Block Diagrams containing algebraic loops.

The circuit of Figure 5 for instance, can be modeled by the block diagram of Figure 6. In this block diagram, the bold lines indicate the presence of an algebraic loop.

The QSS–method can be implemented by transforming the integrators into DEVS models like M_1 (quantized integrators) and the static functions into their DEVS representation (DEVS models like M_2). Then, we can couple these DEVS models according to the coupling scheme of Figure 6.

In fact, that is what we did to convert system (1) into a DEVS representation of (2) (see Figure 2).

The block–by–block translation from a general block diagram representing a continuous system into a DEVS model may result in an inefficient simulation from the point of view of the computational costs. It is better to join all the static functions which calculates each derivative to arrive to a model

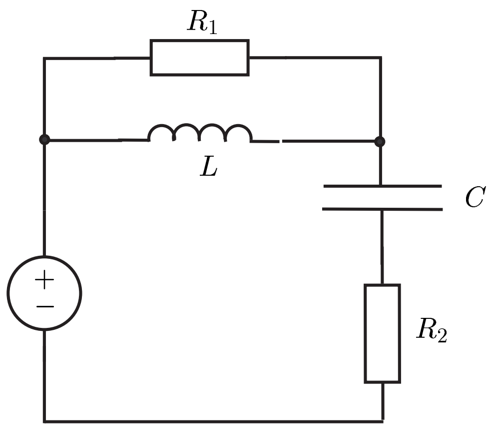

Figure 5: RLC circuit

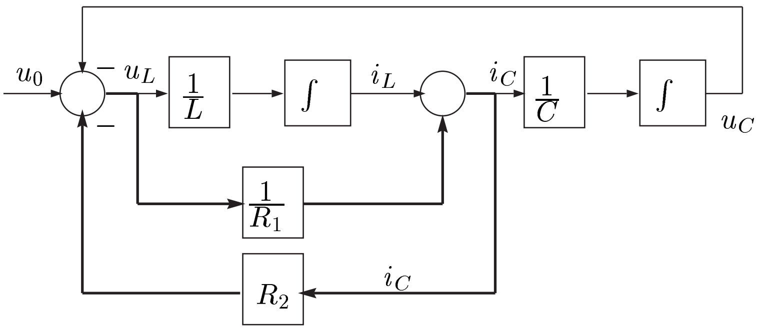

Figure 6: Block Diagram representation of the RLC circuit

like the one shown in Figure 2 reducing thus the traffic of events. However, the block–by–block translation is very simple and it does not require from any kind of symbolic manipulation.

Block Diagrams are a usual tool for representing differential equations and the available DEVS simulation programs do not offer tools to translate them into sets of equations like (1). Thus, if we do not want to perform the translation by hand and we do not have another automatic tool to generate a set of equations like (1), the block–by–block translation is the only possibility we have in order to apply the QSS–method.

However, if the Block Diagram of Figure 6 is translated into a coupled DEVS model a problem will appear. Due to the algebraic loop, the DEVS model will result illegitimate and when an event enters the loop, it will propagate forever through the static functions. Evidently, we need to put a new DEVS model in the loop so that it solves the involved implicit equation.

Suposse we decide to put that *loop–breaking* DEVS model before the gain R_2 as Figure 7 shows.

Then, the loop–breaking model will receive the values of i_C and it should send $\tilde{i}_C$. Each time this DEVS model sends an event, it receives a new event with the value i_C calculated by the static functions

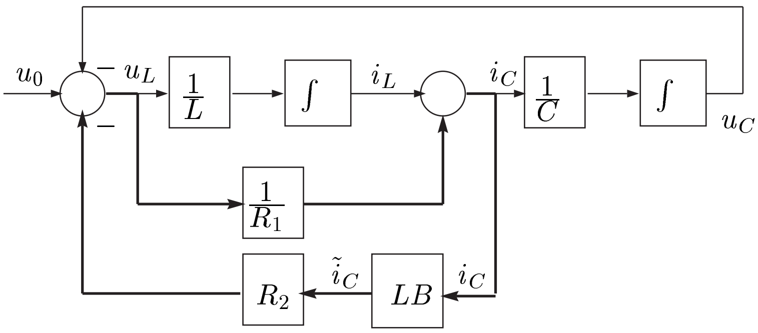

Figure 7: Addition of a Loop-Breaking model to the Block Diagram of Fig.6

belonging to the loop. If this value coincide with the value of $\tilde{i}_C$ that the model had previously sent, it means that the implicit equation was solved and it is unnecessary to send a new $\tilde{i}_C$. In that way, the simulation continues outside the loop until a new i_C arrives to the breaking loop model and the process is repeated.

This idea can solve our problem. However, it has not been said yet how the value of $\tilde{i}_C$ has to be calculated. Before giving an answer to this question the problem should be fromulated in a more formal and general fashion.

Let us call z the variable sent by the loop–breaking model. Then, when it sends an event with value z_1 it immediatly receives a new event with value $h(z_1)$ calculated by the static functions.

Thus, the model should calculate a new value for z, let us call it z_2, which should satisfy

$$h(z_2) - z_2 \triangleq g(z_2) \approx 0 \tag{21}$$

If it is not verfied, the process should be repeated sending a new value z_3.

It is clear that z_{j+1} must be calculated following some algorithm to find the solution of $g(z) = 0$. Taking into account that we do not know the derivative of $g(z)$, a good alternative to the Newton iteration is the use of the Secant–Method.

Using this method, the loop–breaking DEVS model can be represented as follows:

$$M_7 = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ where}$$
$$X = \mathbb{R} \times \{\text{inport}\}$$
$$Y = \mathbb{R} \times \{\text{outport}\}$$
$$S = \mathbb{R}^3 \times \mathbb{R}_0^+ \infty$$
$$\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(z_1, z_2, h_1, \sigma, e, x_v, p) = \begin{cases} (z_1, z_2, h_1, \infty) & \text{if } \ |x_v - z_2| < tol \\ (z_2, \tilde{z}, x_v, 0) & \text{otherwise} \end{cases}$$
$$\delta_{\text{int}}(s) = \delta_{\text{int}}(z_1, z_2, h_1, \sigma) = (z_1, z_2, h_1, \infty)$$
$$\lambda(s) = \lambda(z_1, z_2, h_1, \sigma) = (z_2, \text{outport})$$
$$ta(s) = ta(z_1, z_2, h_1, \sigma) = \sigma$$

with

$$\tilde{z} = \frac{z_1 \cdot x_v - z_2 \cdot h_1}{x_v - h_1 + z_1 - z_2} \tag{22}$$

The parameter *tol* is the maximum error allowed between z and h. Equation (22) is the result of applying the secant–method to approximate $g(z) = 0$ where $g(z)$ is defined according to (21). The iteration algorithm can be changed by modifying Eq. (22).

A loop–breaking DEVS model for QSS2 can be also obtained following similar ideas.

5 Examples

5.1 Block diagram–based simulation of the RLC circuit

The DEVS model corresponding to the Block Diagram of Figure 7 was built. Using parameters $C = L = R_1 = R_2 = 1$ the system step response was simulated.

The quantum size and hysteresis width were chosen equal to 0.01 in both integrators and the error tolerance *tol* in the loop–breaking was taken equal to 0.0001.

The simulation –whose result is shown in Figure 8– was completed after 112 and 72 internal transitions at each quantized integrator and a total of 378 iterations at the loop–breaking DEVS model.

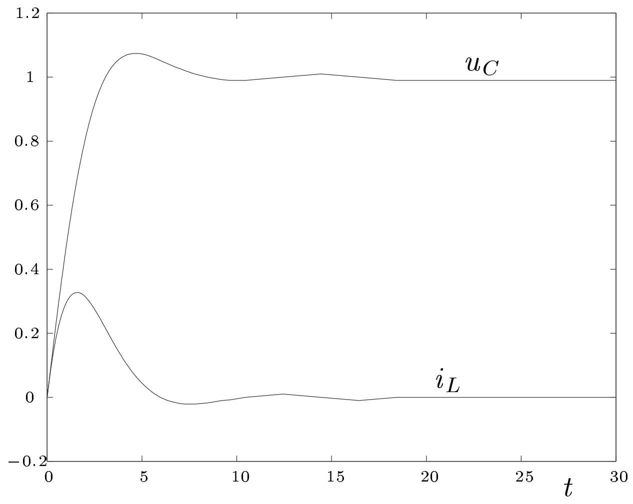

Figure 8: QSS simulation of the RLC circuit using a Loop–Breaking DEVS

In this case, due to the system linearity, during each step the secant–method arrives to the exact solution of $g(z) = 0$ performing only two iterations. This explains the fact that the total number of iterations in the loop–breaking model was twice the total number of steps at both quantized integrators.

The solution obtained is exactly the same which can be obtained if we solve symbolically the implicit equation and then we simulate the resulting ODE with the QSS method.

The main advantage shown by the methodology is its simplicity. We did not perform any calculation to carry on the simulation. We limited our work to connect models like M_1, M_2 and M_7 according to the block diagram of Figure 7.

5.2 QSS2 simulation of a Transmission Line with surge voltage protection

In (Kofman, 2002a) we simulated a transmission line model –taken from (Ismail et al., 1999)– with the QSS2 method.

Figure 9 show the mentioned model, which was modified with the addition of a load.

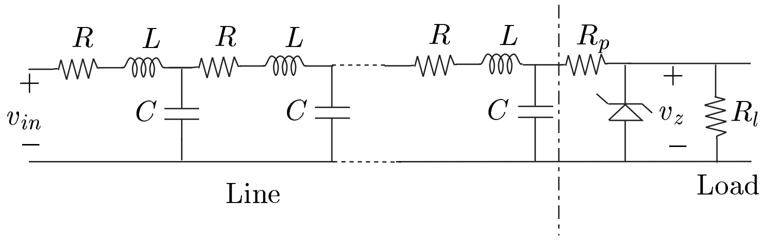

Figure 9: RLC Transmission Line with surge voltage protection

The load is composed by a resistor R_l –which may represent the gate of some electronic component– and a surge protection circuit formed by a zener diode and a resistor R_p. The zener diode satisfies the following nonlinear current–voltage function:

$$i_z = \frac{I_0}{1 - (v_z/v_{br})^m} \tag{23}$$

where m,v_{br} and I_0 are parameters which depend on different physical features.

If the transmission line is divided in five sections –as it is done in the cited references–, the following equations are obtained:

$$\begin{aligned}
\frac{di_1}{dt} &= \frac{1}{L}v_{in} - \frac{R}{L}i_1 - \frac{1}{L}u_1 \\
\frac{du_1}{dt} &= \frac{1}{C}i_1 - \frac{1}{C}i_2 \\
\frac{di_2}{dt} &= \frac{1}{L}u_1 - \frac{R}{L}i_2 - \frac{1}{L}u_2 \\
\frac{du_2}{dt} &= \frac{1}{C}i_2 - \frac{1}{C}i_3 \\
&\vdots \\
\frac{di_5}{dt} &= \frac{1}{L}u_4 - \frac{R}{L}i_5 - \frac{1}{L}u_5 \\
\frac{du_5}{dt} &= \frac{1}{C}i_5 - \frac{1}{R_pC}(u_5 - v_z)
\end{aligned}$$

Here, the state variables u_j and i_j represent the voltage and current in the capacitors and inductors respectively and the output voltage v_z is an algebraic variable which should satisfy

$$\frac{1}{R_p}u_5 - \left(\frac{1}{R_p} + \frac{1}{R_l}\right)v_z - \frac{I_0}{1 - (v_z/v_{br})^m} = 0 \tag{24}$$

Thus, we have a DAE which cannot be converted into an ODE by symbolic manipulation.

This system was simulated with the QSS2 method. The input v_{in} was a trapezoidal trajectory where each pulse has an amplitude of $2.5V$, a duration of $1ns$ and a rising time 100 times faster ($10ps$). The simulation result is shown in Figure 10.

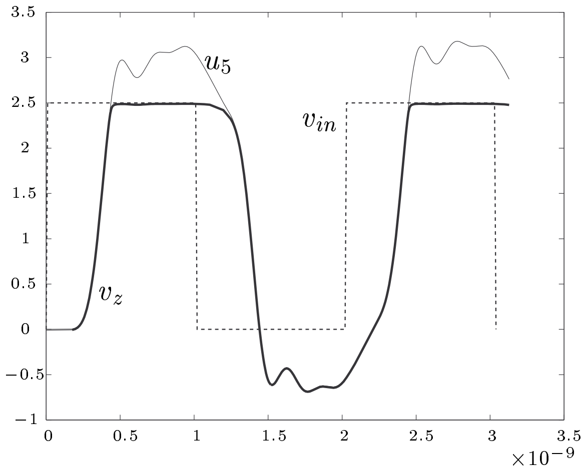

Figure 10: QSS2 simulation results in a RLC transmission line

The parameters used were $R = 80\Omega$, $C = 0.2pF$, $L = 20nH$, $R_p = 100K\Omega$, $R_l = 100M\Omega$, $I_0 = 0.1\mu A$, $v_{br} = 2.5V$ and $m = 4$. The quantization adopted was $\Delta u = 4mV$ in the state variables representing voltages and $\Delta i = 10\mu A$ in the state variables representing currents.

The simulation model consists of 10 first order quantized integrators (atomic DEVS models like M_3), 10 static functions (atomic DEVS models like M_4) and one implicit function to solve Equation (24), which was implemented using an atomic DEVS model like M_6.

The first $32ns$ of the simulation were completed after 2640 steps (between 200 and 316 steps at each integrator). The implicit model performed a total of 485 iterations with the Secant–method. The reason for this is that the quantized integrator which calculates u_5 only performed 200 internal transitions, and then the implicit model received only 200 external events. The Secant–method needs between two and three iterations to find the solution of Eq.(24) with the required tolerance (we used $tol = 1 \times 10^{-8}$) which explains the fact that the total number of iterations was 485 (between 400 and 600).

The advantages of the QSS2 method are evident in this example. In a discrete–time algorithm, the Secant–Method would have been invoked in all the steps while the QSS2 only called it after the changes in u_5 (about once every 13 steps). Thus, the presence of the implicit equation only adds a few calculations which do not affect significantly the total number of computations.

Moreover, if a fixed step algorithm were used, the step size should not be taken bigger than $1ps$ in order to capture properly the rising period. Then, it should perform at least 3200 steps involving calculations in all the state variables and iterations in the implicit equations. As it was already mentioned, the QSS2 only performed 2640 steps, each of them involving calculations in only three state variables and only 200 steps produced iterations in the implicit equation.

6 Conclusions

The QSS and QSS2 methods were extended to their use in the simulation of DAE systems. Similarly to what happens in ODE systems, the quantization–based approach applied to DAE shows some advantages over the discrete–time algorithms.

The most important advantage is probably the fact that the iterations to solve the implicit equations do not have to be performed in every step. In this way, the method avoids a big number of calculations. Taking into account this feature and the fact that each step only involves calculations in some state variables, we conclude that the quantization–based approximations may constitute a powerful tool to solve sparse DAE systems.

Given a block diagram which can have algebraic loops, a new method was developed which allows its block by block translation into a DEVS model which simulates the corresponding QSS or QSS2 approximation. Although the computational efficiency of this methodology is not optimal, it is a systematic methodology with straightforward implementation in any DEVS simulation environment which still exploits sparsity and the other advantages of quantization–based methods.

It can be easily seen that the errors in the calculation of implicit variables may be treated as extra perturbations, which will only increase the error bound without modifying stability properties. Thus, stability is not a problem in the QSS and QSS2 methods nor in ODE neither in DAE systems.

In future work, the higher index problems should be taken into account. Although it was mentioned that the Pantelides'algorithm can be applied to reduce the index to 1 and then the systems can be simulated following the developed methods, a more efficient way might be found.

In fact, when the QSS–method is applied to Bond Graphs with derivative causalities (i.e. index 2 or higher), the system adquires a switching behavior where the iterations are not necessary (Kofman and Junco, 2001a). If this idea is generalized to other higher index DAE systems, the quantization–based method would have an extra advantage over the discrete–time algorithms.

Although we mentioned that stability is not a problem and errors in the calculation of implicit variables only increase the error bound, this fact should be formally proven. It is also important to find a relationship between the tolerance in the implicit equation solution and the increment in the error bound.

Finally, the problem of the parallelization should be solved. The parallel implementation of QSS and QSS2 in ODE and DAE seems to be straightforward since the division into subsystems is clear. If we consider each pair composed by a quantized integrator and the corresponding static function as a subsystem, the traffic of messages between different subsystems is not big. Moreover, in QSS the value carried by a message differ from the previous in a quantum. Then, it can be transmitted using only one bit saying if the quantized variable increased or decreased. This fact may constitute a very important advantage in real–time simulation (in off–line simulation the message should also carry information about the time).

However, all these remarks require a deeper research in order to establish rules for the parallel implementation and to quantify the benefits obtained in terms of execution time reduction.

References

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

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

Ismail, Y., Friedman, E., and Neves, J. (1999). Figures of merit to characterize the importance of On-Chip inductance. *IEEE Trans. on VLSI Systems*, 7(4):442–449.

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

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

Kofman, E. (2002b). Non Conservative Ultimate Bound Estimation in LTI Perturbed Systems. Technical Report LSD0201, LSD-UNR. Submitted to AADECA'02. Available at www.eie.fceia.unr.edu.ar/~ekofman/.

Kofman, E. and Junco, S. (2001a). Quantized Bond Graphs: An Approach for Discrete Event Simulation of Physical Systems. In *Proceedings of ICBGM'01*, pages 369–374, Phoenix.

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

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

Pantelides, C. (1988). The Consistent Initialization of of Differential–Algebraic Systems. *SIAM Journal of Scientific and Statistical Computing*, 9(2):213–231.

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

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