XII Reunión de Trabajo en Procesamiento de la Información y Control, 16 al 18 de octubre de 2007

Linearly Implicit Discrete Event Methods for Stiff ODEs. Part II: Implementation

Gustavo Migoni[†] y Ernesto Kofman[‡]

[†]*Laboratorio de Sistemas Dinámicos FCEIA - UNR - CONICET. Riobamba 245 bis - (2000) Rosario*

Abstract— **This second part deals with the main practical issues of the linearly implicit quantization based integration (QBI) methods defined in the companion paper. The translation of the new algorithms into a discrete event (DEVS) specification and its implementation in a DEVS simulation tool is discussed. The efficience of the methods is illustrated comparing the simulation of two examples with the classic methods implemented by Matlab/Simulink.**

Keywords— **Stiff System Simulation, Quantization Based Integration, DEVS**

I INTRODUCTION

When simulating a continuous system on a digital computer, some quantity must always be discretized. Traditionally, this has always been the time variable. Almost all numerical ODE solvers currently on the market use approximations of either extrapolation polynomials (explicit algorithms) or interpolation polynomials (implicit algorithms) relying on past values of states and state derivatives. Most algorithms operate synchronously, i.e., they use a single simulation run-time clock and update all state variables together [1].

In order to guarantee both accuracy and numerical stability, most of these algorithms furthermore employ a step-size control strategy, i.e., a heuristic scheme built on top of the numerical ODE solver that balances the needs for efficiency with those of accuracy and stability [1].

Yet, time slicing is not the only way how continuous systems can be discretized. Another approach relies on state quantization, thereby leaving the time variable continuous. Methods based on Quantized State Systems (QSS) belong to this class of algorithms. QSS integrators operate naturally in an asynchronous fashion, i.e., each state variable is updated independently of the others, and step size control is an intrinsic feature of these algorithms, i.e., there is no need for a heuristic scheme to be added to the base algorithm to accomplish step-size control.

QSS algorithms can be proved to remain numerically stable (by adjusting the step size), and they offer excellent accuracy control.

Yet, most QSS algorithms introduced in the past [5, 3, 4] were explicit algorithms. They were therefore not well suited for the simulation of stiff systems. They remained numerically stable at the expense of utilizing excruciatingly small step sizes, as any explicit algorithm is expected to require in this situation.

Recently, a new QSS algorithm of the implicit kind was developed [6, 8]. The method, called BQSS (after Backward QSS), was able to efficently integrate many stiff systems. Yet, whereas the algorithm is implicit, it still can be implemented without a need for Newton iteration. The reason is that any state can only assume one of two next values. If a state variable currently assumes a value of q_j, the next value of that state variable must be either $q_j + \Delta q_j$, or $q_j - \Delta q_j$. Hence all possible next values can be *enumerated* without an open-ended search.

However, BQSS has some drawbacks: it is only first order accurate and it introduces an extra perturbation term that can lead to the appearence of spurious equilibrium points.

Both problems were solved with the definition of LIQSS and LIQSS2 methods in the companion paper [7]. The first order accurate LIQSS method follows the idea of BQSS, but avoids the presence of the mentioned perturbation term using a linearly implicit idea to find the state values where some derivatives cross by zero. Then, LIQSS2 combines that idea with the principles of the QSS2 method, resulting in a second order accurate method.

In this work we discuss the implementation of the LIQSS methods as discrete event algorithms in terms of the DEVS formalism [10], and its programming in a DEVS simulation software called PowerDEVS [9]. Then, we introduce some simulation examples and compare the performance of the new methods with the classic algorithms implemented in Matlab/Simulink.

The paper is organized as follows: Section II recalls the principles of DEVS and the implementation of QSS methods as discrete event models. Then, Section III discuss the main implementation issues of the new LIQSS methods. Finally, Section IV introduces the simulation examples and Section V presents conclusions and future work.

II DEVS and Quantization Methods

This section gives a brief introduction to the DEVS formalism and the basic principles of QBI methods.

A Formalismo DEVS

A DEVS model [10] processes an event input trajectory, and, based on that trajectory and the initial state, produces an event output trajectory.

The behavior of an atomic DEVS model is formally defined by the structure:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta) \tag{1}$$

where

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

Each possible state s ($s \in S$) has an associated time advance given by the *time advance function* ($ta(s) \to \mathbb{R}_0^+$). $ta(s)$ is a non–negative number indicating how long the system remains in a given state in absence of input events.

The, if at time t_1 the state takes the value s_1, after $ta(s_1)$ time units (i.e., at time $t_1 + ta(s_1)$) the system performs an internal transition going to a new state $s_2 = \delta_{\text{int}}(s_1)$. Function 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 $y_1 = \lambda(s_1)$. Function λ ($\lambda : S \to Y$) is called *output function*. Functions ta, δ_{int} and λ defined the autonomous behavior of a DEVS model.

When an input event arrives, the state changes instantaneously. The new state depends not only on the input event value but also on the previous state and the elapsed time since the last transition. If the model arrives to state s_3 at t_3 and an input event arrives at time $t_3 + e$ with a value x_1, the new state is calculated as $s_4 = \delta_{\text{ext}}(s_3, e, x_1)$ (notice that $e < ta(s_3)$). 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 *external transition function*. During an external transition no output event is produced.

DEVS models can be coupled and the result of the coupling defines an equivalent atomic DEVS model.

B DEVS and QSS

Given an equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2}$$

with $\mathbf{x} \in \Re^n$, $\mathbf{u} \in \Re^m$, the QSS method [5, 7] approximates it by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) \tag{3}$$

where the components q_j and x_j are related by hysteretic quantization functions. Consequently, the *quantized variables* $q_j(t)$ follow piecewise constant trajectories.

Each component of Eq.(3) can be thought as the coupling of two elementary subsystems. A static one,

$$d_j(t) = f_j(q_1, \cdots, q_n, u_1, \cdots, u_m) \tag{4}$$

and a dynamical one

$$q_j(t) = Q_j(x_j(\cdot)) = Q_j(\int d_j(\tau) d\tau) \tag{5}$$

where Q_j is the hysteretic quantization function (it is not a function of the instantaneous value $x_j(t)$, but a functional of the trajectory $x_j(\cdot)$).

Since the components $u_j(t)$ and $q_j(t)$ are piecewise constant, the output of Subsystem (4), i.e., $d_j(t)$, will be piecewise constant. This way, both subsystem have input and output piecewise constant trajectories that can be represented by sequences of events.

Then, Subsystems (4) and (5) define a relation between their input and output sequances of events. Consequently, equivalent DEVS models can be found for these systems, called *static functions* and *quantized integrators* respectively [5].

The second order accurate QSS2 method can be implemented in the same way of QSS. However, the trajectories are now piecewise linear instead of piecewise constant. Thus, the events carry two numbers that indicate the initial value and the slope of each segment. Also, the static functions and quantized integrators are modified with respect to those of QSS so they can take into account the slopes.

III LIQSS Implementation

This section discussed the implementation of LIQSS methods as DEVS models.

A LIQSS Definition

The LIQSS method was defined in the companion paper [7].

Given a state variable $x_j(t)$, LIQSS uses two hysteretic quantization functions: one from below ($\underline{q}_j(t) \leq x_j(t)$) and the other from above ($\overline{q}_j(t) \geq x_j(t)$). Both quantization functions are defined so that they never differ from x_j in more than $2\Delta Q_j$.

The quantized variable q_j is chosen equal to either $\underline{q}_j$ or $\overline{q}_j$, according to the direction of $\dot{x}_j(t)$. When $\dot{x}_j > 0$ we use $q_j = \overline{q}_j$, and viceversa.

When $\dot{x}_j = f_j(\mathbf{q}, \mathbf{u})$ depends on q_j, it could happen that the sign of the derivative changes when we evaluate f_j using each possibility, i.e., $f_j|_{\underline{q}_j} > 0$ and $f_j|_{\overline{q}_j} < 0$. Thus, we cannot find a correct value for q_j.

However, in that situation, continuity in f_j ensures that a value $\hat{q}_j$ exists, with $\underline{q}_j < \hat{q}_j < \overline{q}_j$, such that $f_j|_{\hat{q}_j} = 0$.

LIQSS then makes $q_j = \tilde{q}_j \approx \hat{q}_j$, where $\tilde{q}_j$ is calculated assuming that f_j depends linearly on q_j (i.e., LIQSS is a linearly implicit algorithm).

Formally, given System (2), the LIQSS method approximates it by Eq.(3), where each q_j is defined by the following function

$$q_j(t) = \begin{cases} \underline{q}_j(t) \text{ if } f_j(\mathbf{q}(t), \mathbf{u}(t))(\overline{q}_j(t) - x_j(t)) \leq 0 \\ \overline{q}_j(t) \text{ if } f_j(\mathbf{q}(t), \mathbf{u}(t))(\overline{q}_j(t) - x_j(t)) \geq 0 \\ \quad \wedge f_j(\mathbf{q}(t), \mathbf{u}(t))(\overline{q}_j(t) - x_j(t)) > 0 \\ \tilde{q}_j(t) \text{ otherwise} \end{cases} \tag{6}$$

with

$$\underline{q}_j(t) = \begin{cases} \underline{q}_j(t^-) - \Delta Q_j \\ \quad \text{if } x_j(t) - \underline{q}_j(t^-) \leq 0 \\ \underline{q}_j(t^-) + \Delta Q_j \\ \quad \text{if } x_j(t) - \underline{q}_j(t^-) \geq 2 \cdot \Delta Q_j \\ \underline{q}_j(t^-) \text{ otherwise} \end{cases} \tag{7}$$

$$\overline{q}_j(t) = \underline{q}_j(t) + 2\Delta Q_j \tag{8}$$

$$\tilde{q}_j(t) = \begin{cases} \overline{q}_j(t) - \frac{1}{A_{jj}} \cdot f_j(\overline{\mathbf{q}}^j(t), \mathbf{u}(t)) & \text{if } A_{jj} \neq 0 \\ q_j(t^-) & \text{otherwise} \end{cases} \tag{9}$$

where $\overline{\mathbf{q}}^j(t)$ is equal to $\mathbf{q}(t^-)$ except for the j–th component, where it is equal to $\overline{q}_j$ and $A_{j,j}$ is the j, j component of the Jacobian matrix evaluated in $\overline{\mathbf{q}}^j$, i.e.,

$$A_{jj} = \left. \frac{\partial f_j}{\partial x_j} \right|_{\overline{\mathbf{q}}^j, u(t^-)} \tag{10}$$

B DEVS Implementation of LIQSS

The main difference between LIQSS and QSS is the way in which $\mathbf{q}$ is obtained from $\mathbf{x}$. Eq.(6) says that q_j not only depends on x_j but also on $\mathbf{q}$.

In QSS, the changes in q_j are produced when x_j differes from it in ΔQ_j. Similarly, in LIQSS q_j changes when x_j reaches q_j. However, changes in q_j can trigger changes in other quantized variables because of (6). In the same way, changes in some component of u_j can also change some quantized variable.

Following the idea of the DEVS implementation of QSS, i.e., by the coupling of *quantized integrators* and *static functions*, but taking into account the mentioned differences, we can obtain a DEVS model for the LIQSS algorithm.

The structure of this implementation is shown in Fig.1.

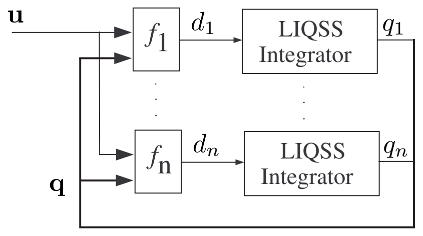

Figure 1: Block Diagram of LIQSS

Since the trajectories of $u_j(t)$ and $q_j(t)$ are piecewise contant, the static funcions are the same than those of QSS.

Then, we only need to define the LIQSS integrators so that they calculate q_j according to the definition of LIQSS.

In order to build the DEVS model of the LIQSS quantized integrator, we shall first analyze its behavior.

Let us suppose that in time t the state x_j reaches the value of q_j with a positive slope ($\dot{x}_j(t^-) > 0$). Then, the upper and lower quantization functions $\overline{q}_j$ and $\underline{q}_j$ must be updated (increasing by ΔQ_j). In this situation we first try with an output value $q_j(t) = \overline{q}_j = q_j(t^-) + \Delta Q_j$.

If, due to the feedback, we receive an input event with $d_j = \dot{x}_j(t) < 0$, then we are in the situation where we need to calculate the value $\hat{q}_j$ that provokes $\dot{x}_j = 0$, i.e., we estimate $\tilde{q}_j$ using Eq.(9), that in this case takes the form

$$\tilde{q}_j = \overline{q}_j - \frac{1}{A_{jj}} \cdot d_j$$

and we set $q_j(t) = \tilde{q}_j$ (supposing that $A_{j,j} \neq 0$, otherwise we just use $q_j = \overline{q}_j$).

The value of $A_{j,j}$ can be easily estimated using the values of $\dot{x}_j(t^-)$, $\dot{x}_j(t)$, $q_j(t^-)$ and $\overline{q}_j(t)$.

It could happen that the integrator then receives (also at time t) another event with a nonzero value (because of the error in the calculation of $\hat{q}_j$). In this case we shall not calculate any further value for q_j at time t.

In the opposite case (when x_j reaches q_j from above) we proceed in an analogous way.

The other case in which q_j changes and the quantized integrator must provoke output events is when a change in the sign of the derivative is received. Suppose that x_j was going up towards $q_j = \overline{q}_j$ and at time t (due to the change in some quantized variable or input), an event with $d_j = \dot{x}_j(t) < 0$ is received.

Then, the integrator must send the new output value $q_j(t) = \underline{q}_j$ so that x_j goes towards q_j. In this case, it can also happen that, due to the feedback, a new event with $\dot{x}_j < 0$ is received at time t and we are again in the situation where we need to calculate $\hat{q}_j$. In this case, we proceed exactly as before, calculating $\tilde{q}_j$ and ignoring any further change of sign at time t.

As before, the case where x_j is initially going down to q_j is completely analogous to the one described above.

In any situation, after calculating q_j, it result easy to schedule the next output event time. It can be calculated as:

$$\sigma_j = \begin{cases} (\overline{q}_j(t) - x_j(t))/d_j & \text{if } d_j > 0 \\ (x_j - \underline{q}_j(t))/d_j & \text{if } d_j < 0 \\ \infty & \text{otherwise} \end{cases}$$

The behavior described for the quantized integrator can be easily translated in a DEVS model. Notice that at each event time the quantized integrator provoke a maximum of two output events. This fact, combined with the property that ensures that $\mathbf{q}$ changes a finite number of times proven in Theorem 1 of [7], guarantees the legitimacy of the DEVS implementation (i.e., the simulation always perform a finite number of steps in any finite interval of time).

C Second Order LIQSS

The second order accurate LIQSS2 method defined in the companion paper [7] is similar to QSS2 but combines it with the idea of LIQSS. Like QSS2, the quantized variable and the state derivative trajectories $q_j(t)$ and $\dot{x}_j(t)$ are piecewise linear. Thus, the state trajectories $x_j(t)$ are piecewise parabolic.

Like LIQSS, this method uses two quantization functions, one from below $\underline{q}_j(t)$ and the other from above $\overline{q}_j(t)$, but these functions are piecewise linear instead of piecewise constant. The quantized variable q_j is equal to one or the other trajectory according to the sign of the second state derivative $\ddot{x}_j$ so that x_j goes towards q_j.

When the sign of $\ddot{x}_j$ changes when we start a new segment of $q_j(t)$, an intermediate slope $\hat{m}_j$ between the old and the new slope exists for which $\ddot{x}_j = 0$. Thus, the LIQSS2 looks for $\hat{m}_j$ in a linear way. Moreover, the algorithm also looks for the initial point $\hat{q}_j$ that makes $\dot{x}_j = \hat{m}_j$. That way, the quantized variable goes parallel to the state and the fast oscillations that appear in stiff systems are eliminated.

Formally, given the system (2), the LIQSS2 method approximates it by (3), where each component q_j is defined as:

$$q_j = \begin{cases} \overline{q}_j(t) & \text{if } \ddot{x}_j(t) > 0 \vee \\ & (\ddot{x}_j(t) = 0 \wedge \dot{x}_j(t) > m_j) \\ \underline{q}_j(t) & \text{if } \ddot{x}_j(t) < 0 \vee \\ & (\ddot{x}_j(t) = 0 \wedge \dot{x}_j(t) <= m_j) \\ \widetilde{q}_j(t) & \text{otherwise} \end{cases}$$

with

$$\begin{aligned} \underline{q}_j(t) &= \begin{cases} x_j(t_0) - \Delta Q_j \quad \text{if } t = t_0 \\ \underline{q}_i(t^-) + \Delta Q_j \\ \quad \text{if } (x_j(t) = \underline{q}_j(t^-) + 2\Delta Q_j \\ \underline{q}_i(t^-) - \Delta Q_j \\ \quad \text{if } (x_j(t) = \underline{q}_j(t^-) \\ \underline{q}_j(t_j) + m_j \cdot (t - t_j) \ \text{ otherwise} \end{cases} \\ \overline{q}_j(t) &= \underline{q}_j(t) + 2 \cdot \Delta Q_j \\ \widetilde{q}_j(t) &= \begin{cases} \dfrac{m_j(t) - \ddot{x}(t^-)}{A_{j,j}} + q_j(t^-) \\ \quad \text{if } A_{j,j} \neq 0 \\ q_j(t^-) \quad \text{otherwise} \end{cases} \end{aligned}$$

and

$$m_j = \begin{cases} m_j(t^-) \text{ if } q_j(t^-) = q_j(t) \\ \dot{x}_j(t^-) \text{ if } (\ddot{x}_j(t) \cdot \ddot{x}_j(t^-) > 0 \vee A_{j,j} = 0) \\ \quad \wedge q_j(t^-) \neq q_j(t) \\ m_j(t^-) - \dfrac{\ddot{x}_j(t^-)}{A_{j,j}} \text{ otherwise} \end{cases}$$

D DEVS Implementation of LIQSS2

The simulation scheme for LIQSS2 is the same than before (Fig.1), but now the trajectories are piecewise linear an parabolic.

Since the quantized variable trajectories of LIQSS2 are, as in QSS2, piecewise linear, the static functions are the same of QSS2.

The main difference between QSS2 and LIQSS2 is the way in which the quantized state variable trajectories are calculated in the integrator.

Each segment of the quantized variable trajectory can be characterized by an initial point q_j and a slope mq_j. In the same way, the state derivative can be characterized by the pair (d_j, md_j). Thus, each input and output event of the quantized integrator will carry two numbers.

Let us then analyze the behavior of the resulting DEVS model. Suppose that at time t the state x_j reaches either $\overline{q}_j$ or $\underline{q}_j$ with $\ddot{x}_j(t^-) > 0$. Using the fact that x_j is piecewise parabolic, we know both $\dot{x}_j(t^-)$ and $\ddot{x}_j(t^-)$. Thus, we set the new segments of $\overline{q}_j$ and $\underline{q}_j$ with slope $m_q = \dot{x}_j(t^-)$ and initial values $x_j + \Delta Q_j$ and $x_j - \Delta Q_j$ respectively. Then, as $\ddot{x}_j(t^-) > 0$ we select $q_j(t) = \overline{q}_j(t)$.

It could happen that, due to the feedback, at time t we then receive an event with slope $md_j(t) = \ddot{x}_j(t) < 0$. Thus, an intermediate output slope $\hat{m}q_j$ between the old and the new one exists that provokes the situation $md_j = \ddot{x}_j = 0$. If $A_{j,j} \neq 0$, this slope can be estimated as

$$\hat{m}q_j(t) \approx \widetilde{m}q_j(t) = mq_j(t^-) - \frac{\ddot{x}(t^-)}{A_{j,j}} \qquad (11)$$

We also search the value $\hat{q}_j(t)$ that makes $md_j = \dot{x}_j(t) = \hat{m}q_j(t)$:

$$\hat{q}_j(t) \approx \widetilde{q}_j(t) = \frac{\widetilde{m}q_j(t) - \ddot{x}(t^-)}{A_{j,j}} + x_j(t^-)$$

Thus, we update the slopes of $\underline{q}_j$ and $\overline{q}_j$ to the value $\widetilde{m}q_j$ and we output an event with the pair $(\widetilde{q}_j, \widetilde{m}q_j$.

Like the case of LIQSS, if we receive another event at time t we do not produce any further output event.

Another situation in which q_j changes is when an event with a change in the sign of the second derivative is received. Suppose that x_j was moving towards q_j with $\ddot{x}_j > 0$ and at time t an event is received with $md_j = \ddot{x}_j < 0$ (due to the change is some other quantized or input variable).

Thus, we first try with $q_j = \underline{q}_j(t)$ and we update the slope $mq_j = \dot{x}_j(t^-)$. If, due to feedback, we receive another event at time t with $md_j > 0$, we must search for the value $\hat{m}q_j$ that makes $md_j = 0$, and we repeat what we did from Eq.(11).

Once q_j is calculated, we must schedule the next event time. The time to the next event is given by the first crossing of x_j with either $\overline{q}_j$ or $\underline{q}_j$. This can be calculated as the minimum positive solution σ_j of the following equations

$$\begin{aligned} x_j(t) + u_j(t) \cdot \sigma_j + \frac{1}{2} mu_j(t)\sigma_j^2 &= \overline{q}_j \\ x_j(t) + u_j(t) \cdot \sigma_j + \frac{1}{2} mu_j(t)\sigma_j^2 &= \underline{q}_j \end{aligned}$$

Similarly to the case of LIQSS, $A_{j,j}$ can be estimated as:

$$A_{j,j}(t) = \frac{md_j(t^-) - md_j}{mq_j(t^-) - mq_j(t)}$$

All these ideas can be easily translated into the DEVS model of the LIQSS2 quantized integratior.

E PowerDEVS Implementation

PowerDEVS [9] is a software tool for DEVS simulation. It has a graphical editor that permits build block diagrams of DEVS models. PowerDEVS libraries contain all the blocks needed to implement the QSS methods, including quantized integrators, static functions, source terms and blocks for discontinuity handling.

The DEVS models corresponding to both LIQSS methods was added to the generic quantized integrator that implementes the QSS methods. Now, this block permits selecting between the following methods: QSS, QSS2, QSS3, BQSS, CQSS, LIQSS and LIQSS2. This block also permits selecting the quantum and the initial state value.

The following section illustrates the usage of these new methods in PowerDEVS.

IV EXAMPLES

This section aimed to introduce some examples which shows the performance advantages of the method in the simulation of non linear and linear stiff systems. The examples were simulated in PowerDEVS and compared with Matlab/Simulink results.

A Second Order Linear System

In the first part of this article [7] the following system was presented:

$$\begin{aligned} \dot{x}_1 &= 0.01x_2 \\ \dot{x}_2 &= -100x_1 - 100x_2 + 2020 \end{aligned} \tag{12}$$

with initial conditions $x_1(0) = 0$ and $x_2(0) = 20$

The system was first simulated using LIQSS and LIQSS2 methods with quantum $\Delta Q_i = 1$. Then, the simulations were repeated decreasing the quantum 10, 100 and 1000 times. The following table shows the number of steps performed for each method using the mentioned quantization:

[h] ΔQ_i	LIQSS1			LIQSS2		
	N°x_1	N°x_2	Total N°	N°x_1	N°x_2	Total N°
1	21	25	46	8	17	24
0.1	201	203	404	20	39	59
0.01	2006	2026	4032	60	126	186
0.001	20064	28174	48238	186	391	577

This table shows that the number of steps performed by LIQSS linearly varies with the quantization, while the number of steps in LIQSS2 grows approximately with the square root of the quantum reduction.

Figure 2 shows the simulation results using Simulink ODE15s with tolerance 10^{-3} and PowerDEVS with $\Delta Q_i = 0.001$ (the difference between both methods cannot be appreciated with the naked eye).

The simulation time could not be evaluated under PowerDEVS using $\Delta Q_i = 0.001$ (it was too small in order to be accurately measure). Thus, we compared the simulation times using quantum $\Delta Q_i = 0.0001$. The simulation in Simulink takes 0.078 seconds (ODE15s, in accelerated mode, with tolerance 10^{-3}), while in PowerDEVS it takes 0.015 seconds.

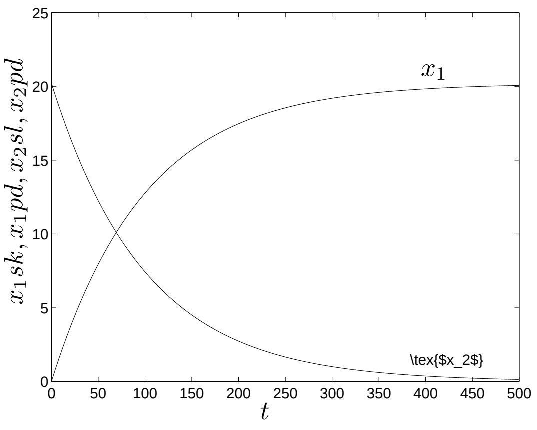

Figure 2: Linear Stiff System Simulation

The global error bound (Equation (23) of [7]) ensures that the error in the LIQSS2 simulation is always less than $2 \cdot 10^{-4}$ in x_1 and $6 \cdot 10^{-4}$ in x_2.

Figure (3) shows the simulation error in PowerDEVS with quanta $\Delta Q_i = 0.0001$. Comaring it with the theoretical bound, we see that the last one is a bit conservative.

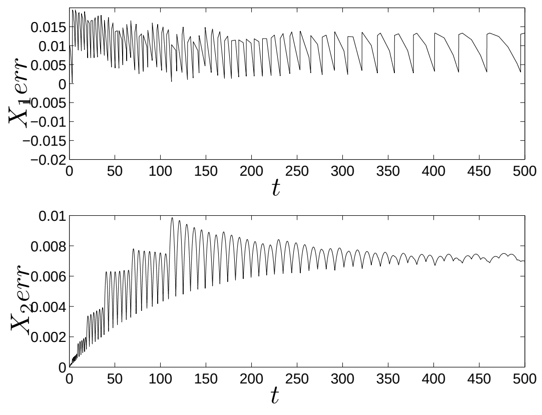

Figure 3: Error

B Van der Pol Oscillator

The problem consists of a second order differential equation proposed by B. van der Pol in the 1920's, that describes the behavior of nonlinear vacuum tube circuits. It has two periodic solutions, the constant solution, $z(t) = 0$, which is unstable, and a nontrivial periodic solution that correspond to an attractive limit cycle. The equation depends on a parameter that weights the importance of the nonlinear part of the equation. The corresponding state equations are:

$$\begin{aligned} \dot{x}_1(t) &= x2 \\ \dot{x}_2(t) &= (1 - x_1^2) \cdot \mu - x_1 \end{aligned} \tag{13}$$

We fixed the parameter $\mu = 1000$, what gives rise to a stiff problem that is often used as a test problem for stiff ODEs solvers [2].

The model was then built in PowerDEVS (Fig.4)

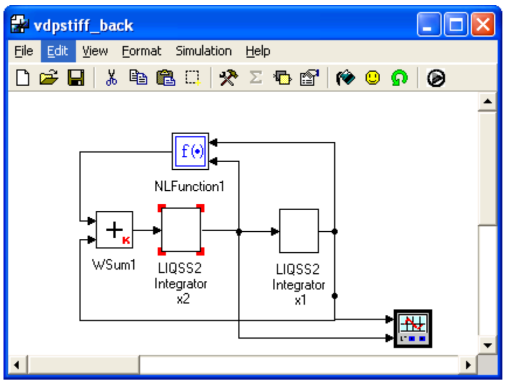

Figure 4: PowerDEVS model of the Van der Pol oscillator

For the simulation, we used initial conditions $x_1(0) = 2$, $x_2(0) = 0$ and quantization $\Delta Q_1 = 0.001$, $\Delta Q_2 = 1$. We simulate the system with LIQSS2 until $t_f = 4000$. The results are shown in Figure 5. The total number of steps was 2159 (838 in x_1 and 1321 in x_2). The simulation takes 0.031 seconds.

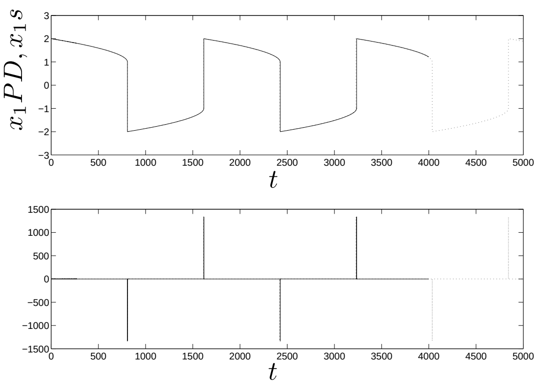

Figure 5: Van der Pol oscillator simulation

The same system was simulated using different Matlab/Simulink methods. The best results were obtained with ODE15s. In order to obtain similar results to the ones given by PowerDEVS the tolerance error must be set to 10^{-14}. Using larger tolerances the results were qualitatively similar but with a significant phase error. In this case, the number of steps was 2697 and the simulation takes 0.056 seconds.

It must be taken into account as we compare the results, that the order of LIQSS2 is smaller that the one of ode15s.

Finally, the system was simulated again using LIQSS2 but with quanta $\Delta Q_1 = 0.0001$ and $\Delta Q_2 = 0.1$ (ten times smaller). The number of steps performed result 4148(1975 in x_1 and 2173 in x_2) and the simulation takes 0.047 seconds. Comparing this result whit the previous one, it can be seen that the number of steps was increased just in a factor of 2 while the quantized levels were reduced in a factor of ten. This put in evidence the second order nature of the LIQS2 method

V CONCLUSIONS

We presented two novel QBI methods that are able to integrate stiff systems based on linearly implicit principles. The fact that the methods do not call for iterations permit achieving and important reduction of computational costs when compared with traditional implicit discrete time methods.

Future work must be done in order to develop higher order methods (following the idea of QSS3 for instance). It is also important to establish which kind of stiff system can be simulated with LIQSS methods.

References

[1] F.E. Cellier and E. Kofman. *Continuous System Simulation*. Springer, New York, 2006.

[2] W. H. Enright and J. D. Pryce. Two (fortran) packages for assessing initial value methods. *(ACM) Transactions on Mathematical Software*, 13(1):1–27, 1987.

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

[4] E. Kofman. A Third Order Discrete Event Simulation Method for Continuous System Simulation. *Latin American Applied Research*, 36(2):101–108, 2006.

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

[6] E. Kofman, G. Migoni, and F.E. Cellier. Integración por Cuantificación de Sistemas Stiff. Parte I: Teoría. In *Proceedings of AADECA 2006*, Buenos Aires, Argentina, 2006.

[7] G. Migoni and E. Kofman. Linearly Implicit Discrete Event Methods for Stiff ODEs. Part I: Theory. Technical Report LSD0107, LSD–UNR, 2007. Enviado a RPIC 2007.

[8] G. Migoni, E. Kofman, and F.E. Cellier. Integración por Cuantificación de Sistemas Stiff. Parte II: Aplicaciones. In *Proceedings of AADECA 2006*, Buenos Aires, Argentina, 2006.

[9] Esteban Pagliero, Marcelo Lapadula, and Ernesto Kofman. PowerDEVS. Una Herramienta Integrada de Simulación por Eventos Discretos. In *Proceedings of RPIC'03*, volume 1, pages 316–321, San Nicolas, Argentina, 2003.

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