Quantization–Based New Integration Methods for Stiff ODEs.

Gustavo Migoni[1],Ernesto Kofman[1] and François Cellier[2]

[1]Laboratorio de Sistemas Dinámicos. FCEIA – UNR
– CIFASIS– CONICET. Rosario, Argentina
[2]Institute of Computational Science, ETH Zurich, Switzerland

May 11, 2010

Abstract

The paper introduces new classes of numerical ODE solvers that base their internal discretization method on state quantization instead of time slicing. These solvers have been coined Quantized State System (QSS) simulators.

The main result of this work is a first order accurate QSS-based stiff system solver called Backward QSS (BQSS). The numerical properties of this new algorithm are being discussed, and it is shown that this algorithm exhibits properties that make it a potentially attractive alternative to the classical numerical ODE solvers. Some simulation examples illustrate the advantages of this method.

As a colateral result, a first order accurate QSS-based solver designed for solving marginally stable systems is sketched. This new method, called Centered QSS (CQSS), is successfully applied to a new difficult benchmark problem describing a high-order system that is simultaneously stiff and marginally stable

1 Introduction.

Practically all of today's commonly used numerical ODE solvers are based on similar principles. They all perform some kind of (either equidistant or non-equidistant) time slicing; they construct an interpolation polynomial that passes through a number of previously computed state and derivative values; and finally, they then use that interpolation polynomial to estimate the value of the state vector at the next discrete time instant, $\mathbf{x_{k+1}} = \mathbf{x}(t_{k+1})$.

Different ODE solvers vary in their approximation orders, i.e., in the number of past pieces of information that they use in the construction of the interpolation polynomial; they differ in which pieces of information they use in constructing the interpolation polynomial; they also differ in their method of time slicing, i.e., how often they compute new state values; and finally, whereas some algorithms are explicit in nature, i.e., make use of past and current information only, others are implicit, i.e., make use of the derivative value at the next time instant, $\dot{\mathbf{x}}(t_{k+1})$, as well.

Yet, all of these algorithms, be they linear or nonlinear methods, single-step or multi-step techniques, explicit or implicit approaches, attempt to answer the same question:

> Given current and past state and derivative information, which values shall the state variables assume at the next discrete time instant?

In this paper, we shall describe a set of numerical ODE solvers that are based on a totally different discretization method. Rather than making use of the concept of *time slicing* to reduce a continuous-time problem to a (in some way equivalent) discrete-time problem that can be solved on a digital computer using an algorithm, these methods employ the concept of *state quantization* for the same purpose.

Hence these methods attempt to provide an answer to a succinctly different question:

> Given that the current value of a state variable, x, is Q_i, where Q_i denotes one of an ordered increasing set of discrete values that the state variable may assume, when is the earliest time instant at which this state variable shall reach either the next higher or the next lower discrete level, $Q_{i\pm1}$?

This concept, first proposed by Zeigler [23, 24], constitutes the basis of a new class of numerical ODE solvers, namely the *Quantized State System (QSS)* solvers [17]. These algorithms have some striking properties in common:

1. QSS algorithms are intrinsically variable-step techniques. They adjust the time instant at which the state variable is re-evaluated to the speed of change of that state variable.

2. QSS methods are naturally asynchronous, i.e., different state variables update their state values separately and independently of each other at different instants of time.

3. QSS techniques are guaranteed to retain numerical stability. They automatically adjust the frequency at which future state values are being computed to meet the numerical stability requirements [11]. The quantization process can be treated as a bounded perturbation on the original ODE. Thus, nonlinear stability can be easily studied making use of Lyapunov functions [17]. These properties have a connection with those established by Dahlquist related to nonlinear stability of conventional numerical solvers. [2, 4].

4. QSS solvers offer a global rather than local error bound, i.e., numerical solutions obtained by QSS algorithms are guaranteed to never differ from the analytical solution by an amount larger than a computable finite value, at least when dealing with linear time-invariant analytically stable systems [11].

5. The QSS approach allows the definition of explicit solvers that are nevertheless stiffly stable [6], something that classical numerical ODE solvers can never do, as this paper shall demonstrate.

6. QSS algorithms offer intrinsically dense output, a feature that is particularly important for asynchronous methods [3].

7. QSS solvers are excellent at root solving, i.e., at simulating across heavily discontinuous models, as they occur frequently in engineering. Due to their dense output feature, their built-in root-solving method is explicit and doesn't require any iteration [12]. More precisely, QSS methods provide low–order polynomial trajectories, the roots of which can be found in a straightforward manner. Thus, the time of the zero–crossings can be calculated before they occur, and the corresponding state events can be scheduled and treated as simple time events.

8. QSS algorithms are good candidates for real-time simulation, because they can be easily implemented on parallel computer architectures (due to their asynchronous nature) [22]; because they can detect and locate discontinuities using explicit methods; and finally, because they allow to minimize the communication bandwidth between separate agents dealing with different subsystems [15], as a state change in any state variable gets communicated asynchronously to its neighbors and to its neighbors only by a single bit (sending a "1" means the state increases its discrete value to the next higher level; sending a "0" means the state decreases its value to the next lower level; and not sending anything means the state remains at the same level as before).

For all of those reasons, it is well worthwhile considering the paradigm shift that QSS methods call for.

Many practical dynamical systems encountered in either science or engineering are stiff, i.e., exhibit Jacobians with eigenvalues that are spread out along the negative real axis of the complex plane. Using traditional methods of time slicing for the numerical simulation of such systems, it results that implicit methods must be used, as all explicit methods invariably restrict the step size in order to guarantee numerical stability. More precisely, algorithms must be employed whose stability domains loop in the right-half complex plane. Some implicit methods exhibit such stability domains, whereas all explicit methods invariably show stability domains that loop in the left-half complex plane [3].

Another case that call for implicit methods is that of marginally stable systems. In these systems, the Jacobians have eigenvalues located near the imaginary axe.

The literature on time slicing methods for stiff and marginally stable systems contains hundreds of algorithms which provide different features and advantages [3, 7, 8]. However, all of them are invariantly implicit and need iterations at each step.

Implicit methods exhibit a serious drawback. They are not useful for real-time simulations, as the resulting Newton iteration cannot be guaranteed to converge within a fixed time interval. Hence the simulation of stiff systems in real time poses a hard problem, for which no good solutions have been found in the past.

Whereas a number of papers have already been written about QSS methods in general (non-stiff QSS solvers of orders 1 to 3 have been developed and have been reported in other publications [3, 11, 14, 17, 20, 21]), this paper is the first to deal with the issues of stiffness and marginal stability. It introduces two new QSS algorithms: BQSS, a stiff first-order ODE solver based on QSS technology, and CQSS, another first-order QSS solver designed for solving marginally stable systems.

Higher-order stiff and geometric QSS solvers are currently under development.

2 Quantization Based Integration.

This section introduces the QSS method and explains the difficulties it is having in dealing with stiff ODE systems.

2.1 QSS Method.

Given the system

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

with $\mathbf{x} \in R^n$ is the state vector, and $\mathbf{u}(t) \in R^m$ represents a known set of input trajectories, the QSS methods approximates it by

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

In this last system, $\mathbf{q}$ is the vector of quantized variables, whose components are related with those of the state vector $\mathbf{x}$ according to the following hysteretic quantization function:

$$q_j(t) = \begin{cases} q_j(t^-) + \Delta Q_j \text{ if } x_j(t^-) - q_j(t) \geq \Delta Q_j \\ q_j(t^-) - \Delta Q_j \text{ if } q_j(t^-) - x_j(t) \geq \varepsilon_j \\ q_j(t^-) \text{ otherwise} \end{cases}$$

ΔQ_j is called *quantum* and ε_j is the *hysteresis width.*

The quantum ΔQ_j is a given parameter that plays a role analogous to that of the step size h in conventional numerical integration methods.

The hysteresis width is usually chosen equal to the quantum, as this choice reduces oscillations without increasing the error [3]. Under this condition, $q_j(t)$

follows a piecewise constant trajectory that only changes when it differs from x_j by ΔQ_j. After each change, $q_j(t) = x_j(t)$.

Using the fact that $\mathbf{q}(t)$ is piecewise constant and provided that the input $\mathbf{u}(t)$ is also piecewise constant, it can be seen that $\mathbf{x}(t)$ is piecewise linear [17]. Consequently, the numerical solution of Eq.(2) is straightforward.

Notice also that $\mathbf{x}(t)$ provides a piecewise linear dense output for the state. Thus, the instants of time at which the trajectories cross a given threshold can be computed analytically. Due to this fact and due to the asynchronous nature of the method, QSS can efficiently handle discontinuities on the right hand side of Eq.(1) [12].

Second- and third-order accurate QSS methods are similar to the first-order algorithm, but they exhibit quadratic and cubic state trajectories, respectively. Although the root–solving problem in these cases carries a higher computational cost, the cost is still negligible in comparison with conventional iterative solutions.

2.2 QSS and Stiff Systems.

The system

$$\begin{aligned} \dot{x}_1(t) &= 0.01\, x_2(t) \\ \dot{x}_2(t) &= -100\, x_1(t) - 100\, x_2(t) + 2020 \end{aligned} \tag{3}$$

has eigenvalues $\lambda_1 \approx -0.01$ and $\lambda_2 \approx -99.99$, which indicates that the system is stiff.

The QSS method approximates this system as

$$\begin{aligned} \dot{x}_1(t) &= 0.01\, q_2(t) \\ \dot{x}_2(t) &= -100\, q_1(t) - 100\, q_2(t) + 2020 \end{aligned} \tag{4}$$

Considering initial conditions $x_1(0) = 0$, $x_2(0) = 20$ together with the quanta $\Delta Q_1 = \Delta Q_2 = 1$, the QSS numerical ODE solver performs the following steps:

At $t = 0$, we set $q_1(0) = 0$ and $q_2(0) = 20$. Then, $\dot{x}_1(0) = 0.2$ and $\dot{x}_2(0) = 20$. This situation remains unchanged until $|q_i - x_i| = \Delta Q_i = 1$.

The next change in q_1 is thus scheduled at $t = 1/0.2 = 5$, whereas the next change in q_2 is scheduled at $t = 1/20 = 0.05$.

Hence a new step is performed at $t = 0.05$. After this step, it results that $q_1(0.05) = 0$, $q_2(0.05) = 21$, $x_1(0.05) = 0.01$, $x_2(0.05) = 21$. The derivatives are $\dot{x}_1(0.05) = 0.21$ and $\dot{x}_2(0.05) = -80$.

The next change in q_1 is rescheduled to occur at $0.05 + (1 - 0.01)/0.21 = 4.764$, whereas the next change in q_2 is scheduled at $0.05 + 1/80 = 0.0625$. Hence, the next step is performed at $t = 0.0625$.

At $t = 0.0625$, it results that $q_1(0.0625) = 0$, $q_2(0.0625) = x_2(0.0625) = 20$, $x_1(0.0625) \approx 0.0126$, and the derivatives coincide with those found at time $t = 0$.

This behavior is cyclicly repeated until a change in q_1 occurs. That change occurs at $t \approx 4.95$, after 158 changes in q_2 oscillating back and forth between 20 and 21.

The simulation continues in the same way. Figs.1–2 show the evolution of $q_1(t)$ and $q_2(t)$ across 500 units of simulated time.

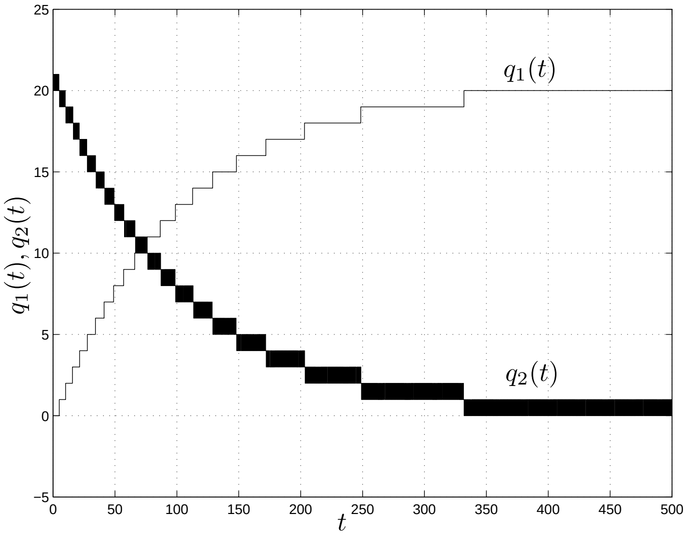

Figure 1: QSS Simulation

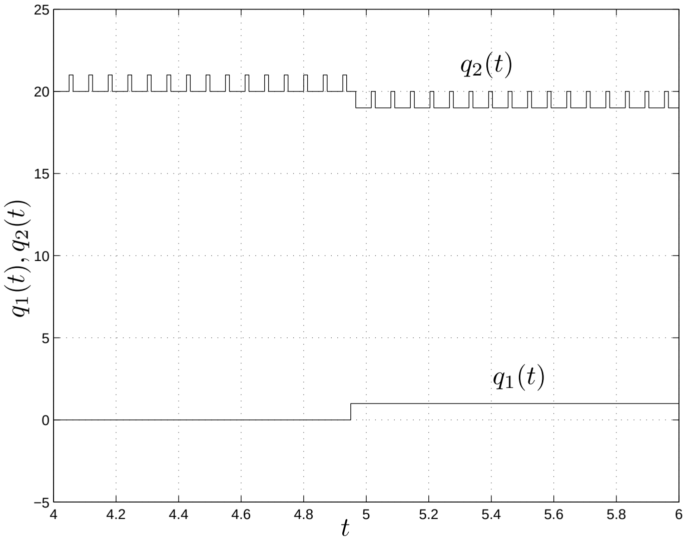

Figure 2: QSS Simulation (detail)

The fast oscillations of q_2 provoke a total of 15995 transitions in that variable, whereas q_1 only changes 21 times. Consequently, the total number of steps needed to complete the simulation is greater than 16000, i.e., the number of simulation steps is of the same order of magnitude as the 25000 steps that would be needed by the Forward Euler method for maintaining numerical stability.

Evidently, the QSS method is unable to efficiently solve System (3).

QSS, as well as other explicit schemes, can solve some very particular stiff problems in an efficient way. For instance, if we change the constant value 2020 by 2000 in Eq.(3) the number of steps is reduced to only 42. Another explicit method similar to QSS was reported to exhibit good performance in some stiff combustion models [19].

Yet, none of these methods can offer a decent performance in a general case. Only implicit methods can efficiently handle general stiff systems.

3 Backward QSS.

3.1 Basic Idea.

Efficient solution of stiff systems requires using implicit methods that evaluate the state derivatives at future instants of time.

This idea, when applied to QSS methods, would imply that the components of $\mathbf{q}(t)$ in (2) are quantized versions of future values of $x(t)$. In other words, given $x_i(t)$, $q_i(t)$ should be a quantized value in the neighborhood of $x_i(t)$, such that $x_i(t)$ evolves towards $q_i(t)$.

For the introductory example (3) using the same initial conditions and quantization as before, this idea would yield the following *simulation*:

At $t = 0$, we can choose either $q_2(0) = 19$ or $q_2(0) = 21$ depending on the sign of $\dot{x}_2(0)$. In both cases, it results that $\dot{x}_1(0) > 0$ and the *future* quantized value of x_1 is $q_1(0) = 1$.

If we choose $q_2(0) = 21$, it results that $\dot{x}_2(0) = -180 < 0$, and consequently, x_2 does not evolve towards q_2. On the other hand, choosing $q_2(0) = 19$ implies that $\dot{x}_2(0) = 20 > 0$, and once again, x_2 does not evolve towards q_2.

Hence it is not possible to choose q_2 such that x_2 moves towards q_2.

However, the fact that the sign of $\dot{x}_2$ changes taking $q_2 = 19$ and $q_2 = 21$ implies that there must exist a point, $\hat{q}_2$, in between those two values, for which $\dot{x}_2 = 0$.

In this case, we set arbitrarily $q_2 = 21$, but we let $\dot{x}_2 = 0$, as if q_2 had adopted the (unknown) value $\hat{q}_2$.

We could have chosen $q_2 = 19$ instead. We would have received another approximation in this way, but both approximations remain bounded, as shall be shown in due course.

The next change in q_1 is thus scheduled at $t = 1/0.21 \approx 4.762$, whereas the next change in q_2 is scheduled at $t = \infty$.

Hence the next step is evaluated at $t = 4.762$. Here, it results that $x_1 = 1$ and $x_2 = 20$. Then, $q_1(4.762) = 2$ (because $\dot{x}_1 > 0$). We evaluate again $\dot{x}_2$ for $q_2 = 19$ and $q_2 = 21$. Now the value is negative in both cases. Thus, the correct value is $q_2(4.762) = 19$ since in that way, x_2 moves towards q_2.

With these new values of q_1 and q_2, it results that $\dot{x}_1 = 0.19$ and $\dot{x}_2 = -80$. The next change in q_1 is then scheduled at $t = 4.762 + 1/0.19 = 10.025$, whereas the next change in q_2 is scheduled at $t = 4.762 + 1/80 = 4.774$. Thus, the next step is performed at $t = 4.774$, when x_2 reaches q_2.

The calculations continue in the same way. The algorithm is similar to that of QSS. The difference is that we try to choose q_i such that x_i evolves towards q_i. When this is not possible, this means that there must exist a point near x_i, for which $\dot{x}_i = 0$. In that case, we enforce that condition but, instead of calculating that point, we keep the previous value of q_i.

Figure 3 shows the result of this simulation that took 21 changes in q_1 and 22 changes in q_2. For $t = 354.24$, the algorithm reaches a stable situation where the changes in both variables are scheduled at $t = \infty$.

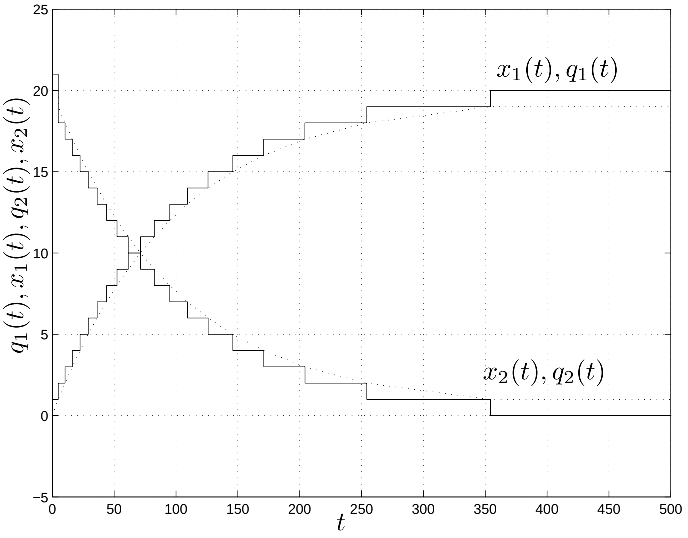

Figure 3: BQSS Simulation

This is the basic idea that defines the BQSS method. For each state variable x_i, we use two quantization functions, one from above and the other from below x_i. Then, q_i takes its value equal to one or the other function according to the sign of the derivative $\dot{x}_i$.

In the case analyzed, this idea worked very well. The algorithm solved the stiff system (3) using 43 steps only, which equals the performance of any implicit method (although the error might be large as a first order approximation is being used).

However as already known from the explicit QSS methods, the use of quantization without hysteresis may provoke infinitely fast oscillations [3, 17]. For instance using the same idea as before with the system

$$\begin{aligned}\dot{x}_1(t) &= 0.5\,x_1 + x_2(t)\\ \dot{x}_2(t) &= -x_1(t) + 0.5\,x_2(t)\end{aligned} \tag{5}$$

with $\Delta Q_i = 1$ and initial conditions $x_1(0) = 0.1$, $x_2(0) = -0.01$, we obtain a solution where the changes in q_1 and q_2 occur faster and faster, and the oscillation frequency goes to infinity.

The solution to this problem, just like in the case of the explicit QSS methods, is to add hysteresis to the upper and lower quantization functions.

3.2 BQSS Definition.

Given the system (1), the BQSS method approximates it by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t)) + \boldsymbol{\Delta}\mathbf{f} \tag{6}$$

where the components of $\mathbf{q}$ are chosen from the set:

$$q_j(t) \in \{\underline{q}_j(t), \overline{q}_j(t)\} \tag{7}$$

with:

$$\underline{q}_j(t) = \begin{cases} \underline{q}_j(t^-) - \Delta Q_j \\ \qquad \text{if } x_j(t) - \underline{q}_j(t^-) \leq 0 \\ \underline{q}_j(t^-) + \Delta Q_j \\ \qquad \text{if } x_j(t) - \underline{q}_j(t^-) \geq \varepsilon_j + \Delta Q_j \\ \underline{q}_j(t^-) \text{ otherwise} \end{cases} \tag{8}$$

$$\overline{q}_j(t) = \begin{cases} \overline{q}_j(t^-) + \Delta Q_j \\ \qquad \text{if } \overline{q}_j(t^-) - x_j(t) \leq 0 \\ \overline{q}_j(t^-) - \Delta Q_j \\ \qquad \text{if } \overline{q}_j(t^-) - x_j(t) \geq \varepsilon_j + \Delta Q_j \\ \overline{q}_j(t^-) \text{ otherwise} \end{cases} \tag{9}$$

In other words, $q_j(t)$ is chosen from a set of two values denoting a lower and an upper bound. These bounds by themselves are being calculated from their own previous values.

We furthermore choose $q_j(t)$ such that $x_j(t)$ approaches $q_j(t)$:

$$f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) > 0 \tag{10}$$

and if either none or both of the two possible values $\underline{q}_j(t)$ and $\overline{q}_j(t)$ satisfy the condition (10), then there must exist a vector $\hat{\mathbf{q}}^{(j)}(t)$, such that $f_j(\hat{\mathbf{q}}^{(j)}(t), \mathbf{u}(t)) = 0$:

$$\exists \hat{\mathbf{q}}^{(j)}(t) \mid f_j(\hat{\mathbf{q}}^{(j)}(t), \mathbf{u}(t)) = 0 \tag{11}$$

where each component of the vector $\hat{\mathbf{q}}^{(j)}(t)$ satisfies:

$$|x_i(t) - \hat{q}_i^{(j)}(t)| < \Delta Q_i + \varepsilon_i \tag{12}$$

Finally, we choose the increments:

$$\Delta f_j = \left\{ \begin{array}{l} 0, \text{ if } f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j - x_j) > 0 \\ -f_j(\mathbf{q}(t), \mathbf{u}(t)), \text{ otherwise} \end{array} \right. \tag{13}$$

i.e., the increments are either zero, if a unique consistent evolution has been found, or alternatively, the increments are chosen such that the corresponding derivatives are set to zero.

Like in QSS, ΔQ_j is called the quantum, and ε_j is the hysteresis width. Contrary to QSS, it is here convenient to choose smaller values of $\varepsilon_j < \Delta Q_j$. The reason is that for such values of ε_j, the two hysteresis loops from above and from below do not intersect. Also contrary to QSS, the oscillation frequency of BQSS does not grow linearly with decreasing values of ε_j. In the current implementation, BQSS sets $\varepsilon_j = 0.01 * \Delta Q_j$.

Notice that the definition of q_j is implicit. Moreover, it can yield more than one solution. However, we know that $q_j(t)$ can only assume one of two values: $\underline{q}_j(t)$ or $\overline{q}_j(t)$.

At first glance, we may think that all combinations of possible values q_j for all components must be evaluated in order to find a correct vector $\mathbf{q}$.

However, it shall be shown that this is not necessary and that $\mathbf{q}$ can in fact be obtained explicitly.

3.3 Explicit calculation of q.

The main difference between BQSS and QSS is the way in which $\mathbf{q}$ is obtained from $\mathbf{x}$, as Eqs.(10) and (11) imply that the values of the different components are interrelated.

In QSS, changes in q_j are produced when x_j differs from q_j by ΔQ_j. In BQSS, changes are provoked when x_j reaches q_j. In addition, a change in q_j may provoke changes in other quantized variables due to Eqs.(10) and (11). Also, changes in some component of $\mathbf{u}$ can produce changes in some quantized variables.

Thus, events might be provoked either by changes in the inputs or because a state variable reached the corresponding quantized variable. After any of those changes, the main difficulty seems to be finding a consistent set of values for $\mathbf{q}$ that satisfy Eqs.(7)–(12).

We shall introduce an algorithm to obtain –in a simple and explicit way– a value of the $\mathbf{q}$ vector that satisfies the aforementioned equations.

In the algorithm, $\mathcal{D} = \{\dots, i, \dots\}$ is the set of sub–indices of the functions f_i already evaluated. Similarly, $\mathcal{A}$ is the set of sub–indices of the q_i that are going to change their values. Both sets are initially empty.

Algorithm 1.

1.a. If an input changes ($u_j(t) \neq u_j(t^-)$):

> The sub–indices of the functions f_i that depend on u_j are included in the set $\mathcal{D}$.

```
> For each $i \in \mathcal{D}$:
    - Define $\widetilde{\mathbf{q}}^{(i)} \triangleq \mathbf{q}(t^-)$.
    - If $f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) \cdot (\widetilde{q}_i^{(i)} - x_i) < 0$
        . Include $i$ in $\mathcal{A}$
        . Define
```

$$q_i(t) \triangleq \begin{cases} \overline{q}_i(t) & \text{if } f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) > 0 \\ \underline{q}_i(t) & \text{if } f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) < 0 \end{cases} \tag{14}$$

```
    - Otherwise, $q_i(t) \triangleq q_i(t^-)$
1.b. If $x_i$ reaches $q_i$:
> Include $i$ in $\mathcal{A}$ and $\mathcal{D}$.
> Define $\widetilde{\mathbf{q}}^{(i)} \triangleq \mathbf{q}(t^-)$.
> Calculate $q_i(t)$ according to Eq.(14)
2. While $\mathcal{A} \neq \phi$
> Let $j$ be the smallest element of $\mathcal{A}$.
> Define $\mathcal{B}$ as the set of sub–indices $i \notin \mathcal{D}$ so that $f_i$ depends on $q_j$.
> For each $i \in \mathcal{B}$:
    – Define $\widetilde{\mathbf{q}}^{(i)}$ according to
```

$$\widetilde{q}_k^{(i)} = \begin{cases} q_k(t) & \text{if } k \in \mathcal{D} \\ q_k(t^-) & \text{if } k \notin \mathcal{D} \end{cases} \tag{15}$$

```
    – If $f_i(\widetilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) \cdot (\widetilde{q}_i^{(i)} - x_i) < 0$
        . Include $i$ in $\mathcal{A}$.
        . Calculate $q_i(t)$ according to Eq.(14).
    – Otherwise, $q_i(t) \triangleq q_i(t^-)$.
> Add the elements of $\mathcal{B}$ to the set $\mathcal{D}$ and remove $j$ from $\mathcal{A}$.
3. For every $i \notin \mathcal{D}$, leave $q_i(t) = q_i(t^-)$.
```

This algorithm always finds a value for $\mathbf{q}$. We shall prove below that it satisfies the definition provided in Section 3.2. Notice that the sub–indices can be included in set $\mathcal{D}$ only once. Thus, every component of function $\mathbf{f}$ is being evaluated at most once.

Theorem 1 *Consider the BQSS approximation of Eq.*(6) *and suppose that* $\mathbf{x}(t)$ *and* $\mathbf{q}(t^-)$ *are known, and they satisfy Eqs.*(7)*–*(12)*. Suppose further that either* $u_i(t) \neq u_i(t^-)$ *or* $x_i(t) = q_i(t^-)$ *for some sub–index* i*. Then, Algorithm 1 always finds a value of* $\mathbf{q}(t)$ *that satisfies Eqs.*(7)*–*(12)*.*

Proof. For every $i \notin \mathcal{D}$, we define $\widetilde{\mathbf{q}}^{(i)} \triangleq \mathbf{q}(t^-)$.

We shall prove that an arbitrary component $q_j(t)$ satisfies Eqs.(7)–(12).

Note that the only values that $q_j(t)$ can take are $q_j(t^-)$, $\underline{q}_j(t)$ and $\overline{q}_j(t)$. In the latter two cases, Eq.(7) is automatically satisfied.

In the first case, when $q_j(t) = q_j(t^-)$, it can be seen that $\underline{q}_j(t) = \underline{q}_j(t^-)$ and $\overline{q}_j(t) = \overline{q}_j(t^-)$. Otherwise, $x_j(t)$ would have reached its quantized variable $q_j(t^-)$ and we would not have assigned $q_j(t) = q_j(t^-)$ (item 1.b.). This ensures that Eq.(7) is always satisfied.

To prove that Eqs.(10)–(12) are satisfied, we must show that if $f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) \leq 0$ then $\exists \hat{\mathbf{q}}^{(j)}$ such that $f_j(\hat{\mathbf{q}}^{(j)}, \mathbf{u}(t)) = 0$ and the components of $\hat{\mathbf{q}}$ satisfy (12).

Notice that if $q_j(t) \neq q_j(t^-)$, the new value is calculated according to Eq.(14). Thus taking into account Eqs.(8)–(9), it will be true that

$$f_j(\widetilde{\mathbf{q}}^{(j)}, \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) \geq 0 \tag{16}$$

On the other hand if we set $q_j(t) = q_j(t^-)$, it is because Eq.(16) is satisfied. Consequently, Eq.(16) is always satisfied.

Taking this equation into account, if $f_j(\mathbf{q}(t), \mathbf{u}(t)) \cdot (q_j(t) - x_j(t)) \leq 0$, then

$$f_j(\widetilde{\mathbf{q}}^{(j)}(t), \mathbf{u}(t)) \cdot f_j(\mathbf{q}(t), \mathbf{u}(t)) \leq 0$$

and from the mean value theorem, there exists $\hat{\mathbf{q}}^{(j)}$ between $\widetilde{\mathbf{q}}^{(j)}(t)$ and $\mathbf{q}(t)$ such that $f_j(\hat{\mathbf{q}}^{(j)}, \mathbf{u}) = 0$.

Eqs.(8) and (9) ensure that

$$|\overline{q}_i - x_i| \leq \Delta Q_i + \varepsilon_i\,; \ \ |\underline{q}_i - x_i| \leq \Delta Q_i + \varepsilon_i \tag{17}$$

The components $\widetilde{q}_i^{(j)}$ can only adopt the values $\overline{q}_i(t)$, $\underline{q}_i(t)$, or $x_i(t)$ (item 1.b). Then

$$\begin{array}{l} |\widetilde{q}_i^{(j)} - x_i(t)| \leq \Delta Q_i + \varepsilon_i \\ |q_i - x_i(t)| \leq \Delta Q_i + \varepsilon_i \end{array} \tag{18}$$

From this equation and since $\hat{q}_i^{(j)}$ is between $q_i(t)$ and $\widetilde{q}_i^{(j)}(t)$, it results that

$$|\hat{q}_i^{(j)} - x_i(t)| \leq \Delta Q_i + \varepsilon_i \tag{19}$$

which concludes the proof □

4 Theoretical Properties of BQSS.

This section studies fundamental properties of the BQSS method. We first show that the BQSS method performs a finite number of steps within any finite interval of time. Then, we analyze the stability and global error bound properties.

4.1 Legitimacy of BQSS.

The following theorem ensures that the BQSS method cannot exhibit oscillatory behavior with a frequency approaching infinity, i.e., the algorithm always performs a finite number of steps within any finite interval of time. An algorithm that satisfies this property is called *legitimate*.

Theorem 2 *Suppose that function* $\mathbf{f}$ *in Eq.*(1) *is bounded in a domain* $D \times D_u$*, where* $D \subset \mathbb{R}^n$*,* $D_u \subset \mathbb{R}^m$ *and assume that the trajectory* $\mathbf{u}(t) \in D_u$ *is piecewise constant. Then,*

1. *Any solution* $\mathbf{x}(t)$ *of Eq.*(6) *is continuous while* $\mathbf{q}(t)$ *remains in* D*.*
2. *The trajectory* $\mathbf{q}(t)$ *is piecewise constant while it remains in* D*.*

Proof. The proof of 1. is straightforward since, according to Eq.(6), the derivative of $\mathbf{x}$ is bounded.

Concerning 2. it is clear that every component q_j can only assume values of the form $k \cdot \Delta Q_j$. However to prove that $\mathbf{q}$ is piecewise constant, it is necessary to ensure that it only experiences a finite number of changes in any finite interval of time.

Let (t_1, t_2) be an arbitrary interval of time in which $\mathbf{q}(t)$ remains in D. We shall prove that, within this interval, $\mathbf{q}(t)$ undergoes a finite number of changes.

The assumptions of the theorem ensure that $\mathbf{f}(\mathbf{q}, \mathbf{u})$ is bounded and, taking into account Eqs.(6) and (13), positive constants m_j exist such that, for $t \in (t_1, t_2)$

$$|\dot{x}_j(t)| \leq m_j; \text{ for } j = 1, \ldots, n.$$

Let $t_c \in (t_1, t_2)$ and suppose that $\overline{q}_j(t_c^-) \neq \overline{q}_j(t_c^+)$. According to Eq.(9), this situation cannot be repeated until $|x_j(t) - x_j(t_c)| \geq \epsilon_j$. Thus, the minimum time interval between two discontinuities in $\overline{q}_j(t)$ is

$$t_j = \frac{\epsilon_j}{m_j}$$

Then, calling $\overline{n}_j$ the number of changes of $\overline{q}_j(t)$ in the interval (t_1, t_2), it results that

$$\overline{n}_j \leq (t_2 - t_1)\frac{m_j}{\epsilon_j}$$

It can be easily seen that $\underline{q}_j$ will perform a maximum number of changes bounded by the same expression.

Since $\mathbf{u}(t)$ is piecewise constant, it will perform a finite number of changes n_u in the interval (t_1, t_2).

The definition of q_j ensures that it can only take the values $\overline{q}_j(t)$ or $\underline{q}_j(t)$. In addition, it can only change if these variables change or if there is a change in some other quantized or input variable ($q_i(t)$ or $u_i(t)$) such that the restrictions of Eqs.(10) and (11) hold. In conclusion, changes in $q_j(t)$ are linked to changes in some $\overline{q}_i(t)$, $\underline{q}_i(t)$ or $u_i(t)$. Thus, the total number of changes will be equal to or less than the sum of all the changes in those variables, i.e.,

$$n_j \leq n_u + 2(t_2 - t_1)\sum_{i=1}^{n} \frac{m_i}{\epsilon_i}$$

which is a finite number. □

4.2 Perturbed Representation.

Stability and error bound properties of QSS methods can be analyzed considering that Eq.(2) can be viewed as a perturbed version of the original system of Eq.(1), where the perturbations are bounded by the quantization in use. We shall see that something similar occurs with BQSS.

Each component of Eq.(6) can be written as

$$\dot{x}_i(t) = f_i(\mathbf{q}(t), \mathbf{u}(t)) + \Delta f_i \tag{20}$$

Defining

$$\overset{*}{\mathbf{q}}^{(i)}(t) = \begin{cases} \mathbf{q}(t) & \text{if } \Delta f_i = 0 \\ \hat{\mathbf{q}}^{(i)}(t) & \text{otherwise} \end{cases}$$

and using Eqs. (10)–(13), Eq.(20) can be rewritten as

$$\dot{x}_i = f_i(\overset{*}{\mathbf{q}}^{(i)}(t), \mathbf{u}(t)) \tag{21}$$

Defining $\mathbf{\Delta x}^{(i)}(t) \triangleq \overset{*}{\mathbf{q}}^{(i)} - \mathbf{x}(t)$ and replacing it in Eq.(21), it results that

$$\dot{x}_i(t) = f_i(\mathbf{x}(t) + \mathbf{\Delta x}^{(i)}(t), \mathbf{u}(t)) \tag{22}$$

where

$$|\Delta x_j^{(i)}(t)| \leq \Delta Q_j + \varepsilon_j \tag{23}$$

because from Eqs.(7), (8), and (9) it results that

$$|q_j(t) - x_j(t)| \leq \Delta Q_j + \varepsilon_j$$

and from Eq.(12) we have that

$$|\hat{q}_j^{(i)}(t) - x_j(t)| \leq \Delta Q_j + \varepsilon_j$$

4.3 Stability and global error bound.

In the linear time–invariant case, Eq.(1) can be written as:

$$\dot{\mathbf{x}}(t) = A\,\mathbf{x}(t) + B\mathbf{u}(t) \tag{24}$$

and the BQSS approximation is

$$\dot{\mathbf{x}}(t) = A\,\mathbf{q}(t) + B\mathbf{u}(t) + \mathbf{\Delta f}(t) \tag{25}$$

For a stable system of this type[1], the following theorem shows the existence of a global error bound:

Theorem 3 *Assume that matrix A is Hurwitz[2]. Let $\phi(t)$ be the solution of Eq.(24) with initial condition $\phi(0)$, and let $\tilde{\phi}(t)$ be a solution of Eq.(25) with the same initial condition. Let $\mathbf{e}(t) \triangleq \phi(t) - \tilde{\phi}(t)$. Then for all $t \geq 0$, it results that[3]*

$$|\mathbf{e}(t)| \leq |V| \cdot |\mathbb{R}e(\Lambda)^{-1} V^{-1}| \cdot |A| \cdot (\mathbf{\Delta Q} + \varepsilon) \tag{26}$$

where $\Lambda = V^{-1} A\, V$ is the spectral decomposition of A, and $\mathbf{\Delta Q}$ and ε are the quantization and hysteresis width vectors in Eq.(25).

Proof. According to Eq.(22), the i–th component of Eq.(25) can be rewritten as

$$\dot{x}_i(t) = A_i(\mathbf{x}(t) + \mathbf{\Delta x}^{(i)}(t)) + B_i\,\mathbf{u}(t)$$

Defining $d_i(t) \triangleq A_i\, \mathbf{\Delta x}^{(i)}(t)$, we can write

$$\dot{x}_i(t) = A_i\,\mathbf{x}(t) + d_i + B_i\,\mathbf{u}(t) \tag{27}$$

From Eq.(23) and the definition of d_i it results that

$$|d_i(t)| \leq |A_i| \cdot (\mathbf{\Delta Q} + \varepsilon) \tag{28}$$

Resuming vector notation, Eq.(27) can be written as:

$$\dot{\mathbf{x}}(t) = A\,\mathbf{x}(t) + B\,\mathbf{u}(t) + \mathbf{d}(t) \tag{29}$$

with

$$|\mathbf{d}(t)| \leq |A| \cdot (\mathbf{\Delta Q} + \varepsilon) \tag{30}$$

Replacing $\mathbf{x}(t)$ in Eq.(24) by $\phi(t)$ and in Eq.(29) by $\tilde{\phi}(t)$ and subtracting the two equations from each other, we obtain:

$$\dot{\mathbf{e}}(t) = A\,\mathbf{e}(t) + \mathbf{d}(t) \tag{31}$$

with $\mathbf{e}(0) = 0$.

When A is Hurwitz and diagonalizable, Theorem 3 of [13] establishes the validity of Eq.(26) from Eqs.(30) and (31). The same result can be derived for the non–diagonalizable case from Theorem 3.3 of [16]. □

Corollary 1 *If matrix A is Hurwitz, the BQSS numerical approximation gives ultimately bounded results, i.e., it ensures* practical stability

Corollary 2 *The global error bound has a linear dependence on the quantization*

[1] The definitions and main theoretical properties of QSS methods hold for both, linear and nonlinear cases. However, unconditional practical stability and error bound only apply for linear time invariant systems [11, 3]. In BQSS the situation is the same. All the definitions and properties hold for both, linear and nonlinear systems, except this one.

[2] A square matrix is called Hurwitz when all its eigenvalues have negative real part. A system like Eq.(24) is stable iff matrix A is Hurwitz.

[3] The symbol "$\leq$" denotes here a componentwise inequality between two vectors. Similarly, "$|\cdot|$" denotes the matrix or vector of componentwise computed absolute values of the matrix or vector elements.

5 Towards F-Stability.[2]

The second–order linear time–invariant system

$$\begin{aligned}\dot{x}_1(t) &= \alpha\, x_1(t) + \omega\, x_2(t)\\ \dot{x}_2(t) &= -\omega\, x_1(t) + \alpha\, x_2(t)\end{aligned}\tag{32}$$

has conjugate complex eigenvalues $\lambda_{1,2} = \alpha \pm i\omega$. It is asymptotically stable for $\alpha < 0$, marginally stable for $\alpha = 0$, and unstable for $\alpha > 0$.

Figure 4 shows the result of simulating this system with QSS1 and BQSS methods, for the parameters $\omega = 1$, $\alpha = 0$ (a marginally stable case) and initial states $x_1 = 4$, $x_2 = 0$.

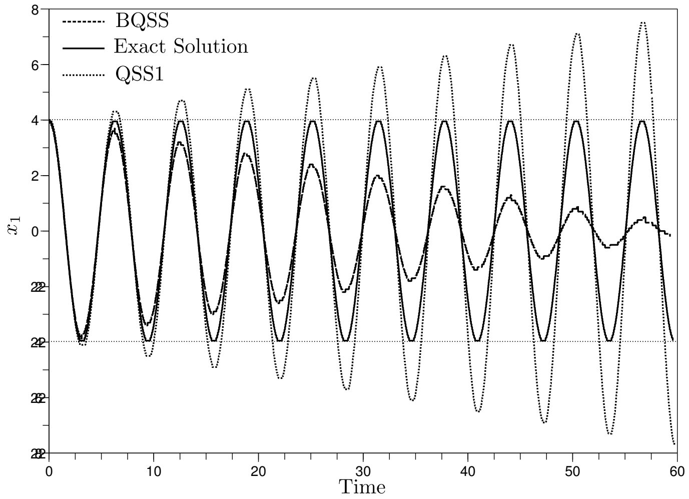

Figure 4: QSS and BQSS simulation of a marginally stable system

Both methods give qualitatively wrong results, as they are not F–stable. QSS and BQSS methods integrate an approximate system of the form

$$\begin{aligned}\dot{x}_1(t) &= \alpha\,(x_1(t) + \Delta x_1(t)) + \omega\,(x_2(t) + \Delta x_2(t)) + \Delta f_1\\ \dot{x}_2(t) &= -\omega\,(x_1(t) + \Delta x_1(t)) + \alpha\,(x_2(t) + \Delta x_2(t)) + \Delta f_2\end{aligned}\tag{33}$$

The perturbation terms Δx_j are bounded according to

$$|\Delta x_j| \le \Delta Q_j + \varepsilon_j < 2\Delta Q_j \tag{34}$$

The quantities Δf_j are normally zero, except in BQSS, where they can adopt a value that brings the derivative to zero. According to the definition, Δf_1 can be nonzero only when

$$\alpha\,\hat{q}_1(t) + \omega\,\hat{q}_2(t) = 0 \tag{35}$$

[2] F-stable numerical ODE solvers are defined as to have their border of stability coincide with the imaginary axis of the complex $\lambda \cdot h$ plane [3]. F-Stability was also referred to as precise A-stability [18]

with

$$|\hat{q}_i - x_i| \leq \Delta Q_i + \varepsilon_i < 2\Delta Q_i, \tag{36}$$

In this case:

$$-\Delta f_1 = \alpha(x_1 + \Delta x_1) + \omega(x_2 + \Delta x_2)$$

From the last equation and Eq.(35), we can write

$$\Delta f_1 = \alpha(\hat{q}_1 - x_1 - \Delta x_1) + \omega(\hat{q}_2 - x_2 - \Delta x_2)$$

Then, using Eqs.(34) and (36), we obtain

$$|\Delta f_1| < 4|\alpha|\Delta Q_1 + 4|\omega|\Delta Q_2 \tag{37}$$

A similar analysis concludes that

$$|\Delta f_2| < 4|\omega|\Delta Q_1 + 4|\alpha|\Delta Q_2 \tag{38}$$

We shall analyze the stability of (33) using the Lyapunov candidate

$$V(x) \triangleq \frac{1}{2}(x_1^2 + x_2^2). \tag{39}$$

with the time derivative

$$\dot{V}(x) = \alpha(x_1^2 + x_2^2) + x_1(\alpha\Delta x_1 + \omega\Delta x_2 + \Delta f_1) + x_2(\alpha\Delta x_2 - \omega\Delta x_1 + \Delta f_2)$$

Thus,

$$\begin{aligned}\alpha\dot{V}(x) &= \alpha^2(x_1^2 + x_2^2) + x_1\alpha(\alpha\Delta x_1 + \omega\Delta x_2 + \Delta f_1) + x_2\alpha(\alpha\Delta x_2 - \omega\Delta x_1 + \Delta f_2)\\ &\geq \alpha^2\|x\|^2 - 5|\alpha| \cdot \|x\|(|\alpha| + |\omega|)(\Delta Q_1 + \Delta Q_2).\end{aligned}$$

Then, provided that $\alpha \neq 0$ and

$$\|x\| > 5\left(1 + \left|\frac{w}{\alpha}\right|\right)(\Delta Q_1 + \Delta Q_2) \tag{40}$$

it results that $\alpha\dot{V}(x) > 0$.

When $\alpha < 0$, $\dot{V}(x)$ is negative on all level surfaces where Eq.(40) holds. Then, both methods, QSS and BQSS, give *practically stable* results. This is, the solutions approach a bounded region around the origin.

Similarly when $\alpha > 0$, $\dot{V}(x)$ is positive on the same level surfaces, and the Euclidean norm of x grows monotonically, provided that the initial condition satisfies (40).

Thus, both methods are numerically stable for eigenvalues in the open left-half plane and unstable for eigenvalues in the open right-half plane.

Let us now analyze the case $\alpha = 0$, i.e., when the eigenvalues are located on the imaginary axis. In this case, it can be seen that $\Delta f_i = 0$ in BQSS whenever $\|x\| > 2\Delta Q_1 + 2\Delta Q_2$. The reason is that the derivative of x_1 only depends on the sign of q_2 and the derivative of x_2 only depends on the sign of q_1. In this way, we can always find q_1 and q_2 in the direction of the corresponding derivatives.

Then, for $\|x\|$ large enough, it results that

$$\dot{V}(x) = x_1\omega\Delta x_2 - x_2\omega\Delta x_1 \tag{41}$$

Using (33) to substitute the terms $x_1\omega$ and $x_2\omega$ we obtain

$$\begin{aligned}\dot{V}(x) &= (-\dot{x}_2 - \omega\Delta x_1)\Delta x_2 - (\dot{x}_1 + \omega\Delta x_1)\Delta x_1 \\ &= -\dot{x}_1\Delta x_1 - \dot{x}_2\Delta x_2\end{aligned}$$

In BQSS, the definition ensures that $\dot{x}_i\Delta x_i \geq 0$. Moreover, the situation $\dot{x}_i\Delta x_i = 0$ is only possible when $\dot{x}_i = 0$. Thus, the simulation with BQSS of marginally stable systems will produce practically stable results as V (and hence $\|x\|$) decreases with the time.

For QSS, we need to evaluate V between two instants of time

$$V(t_b) - V(t_a) = \int_{t_a}^{t_b} \dot{V}(x)dt = \int_{t_a}^{t_b} (-\dot{x}_1\Delta x_1 - \dot{x}_2\Delta x_2)dt \triangleq -\Delta V_1 - \Delta V_2$$

Let us call $t_1, t_2, \cdots, t_m$ the instants of time in the interval (t_a, t_b) where x_1 reaches quantization levels. The first term ΔV_1 can be expressed as

$$\Delta V_1 = \int_{t_a}^{t_1} \dot{x}_1\Delta x_1 dt + \sum_{k=1}^{m-1} \int_{t_k}^{t_{k+1}} \dot{x}_1\Delta x_1 dt + \int_{t_m}^{t_b} \dot{x}_1\Delta x_1 dt$$

Since $\Delta x_1 = q_1 - x_1$ and $q_1(t)$ remains constant between t_k and t_{k+1} we can calculate

$$\int_{t_k}^{t_{k+1}} \dot{x}_1(q_1(t_k) - x_1(t))dt = \int_{x_1(t_k)}^{x_1(t_{k+1})} (q_1(t_k) - x_1)dx_1$$

If t_k and t_{k+1} are instants of time at which x_1 crosses the same quantization value, the integral is 0. Otherwise, $x_1(t_{k+1}) - x_1(t_k) = \pm\Delta Q_1$ and

$$\begin{aligned}\int_{t_k}^{t_{k+1}} \dot{x}_1\Delta x_1(t)dt &= q_1(t_k)(x_1(t_{k+1}) - x_1(t_k)) - \frac{1}{2}(x_1(t_{k+1})^2 - x_1(t_k)^2) \\ &= (x_1(t_{k+1}) - x_1(t_k)) \cdot (q_1(t_k) - \frac{x_1(t_{k+1}) + x_1(t_k)}{2})\end{aligned}$$

In QSS, we have $q_1(t_k) = x_1(t_k)$, and the integral evaluates to $-\Delta Q_1^2/2$. Hence QSS subtracts this constant value from ΔV_1 between successive steps, except when x_1 changes its direction. A similar analysis concludes that ΔV_2 decreases by $-\Delta Q_2^2/2$ between successive steps in x_2. In conclusion, V grows over time, and we obtain an unstable result.

In order to achieve F–stability, we require that the integral in the last equation becomes zero. This can be achieved by modifying the QSS method such that $q_i(t_k) = 0.5(x_i(t_{k+1}) + x_i(t_k))$, i.e., by taking the mean value between the QSS and BQSS values for q.

This very simple idea defines a new method that shall be called Centered QSS or CQSS method. Figure 5 shows the simulation with CQSS of the marginally stable system simulated in Fig.4.

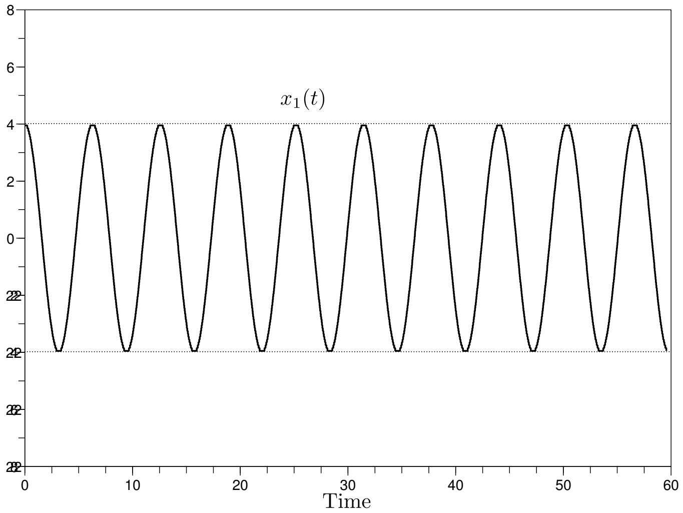

Figure 5: CQSS simulation of a marginally stable system

6 Examples.

The new BQSS and CQSS methods were programmed in PowerDEVS, a DEVS simulation engine that already implemented the QSS family of numerical integration methods [1]. In this section we report the simulation results of four examples of increasing complexity.

6.1 Linear second order stiff system.

Consider again the introductory example of Eq.(3), with initial conditions $x_1(0) = 0$, $x_2(0) = 20$. This time, we wish to solve the system until $t = t_f = 1000$.

We first repeated the introductory experiment using BQSS with a quantum $\Delta Q_1 = \Delta Q_2 = 1$. Then, we simulated the system twice more while reducing the quantum first by a factor of 10, and finally by a factor of 100.

The first simulation took 21 and 22 steps in x_1 and x_2, respectively. Using $\Delta Q_i = 0.1$, the number of steps was 204 in each variable. For $\Delta Q_i = 0.01$, we observed 2022 and 2038 steps in x_1 and x_2.

In all three cases, the simulation arrived at a stable situation where the algorithm does not perform any further step before $t = 500$.

It can be seen that the number of steps in each variable is given by the division of the corresponding signal amplitude by the quantum.

Figure 6 shows the results with $\Delta Q_i = 1$ (solid lines) and $\Delta Q_i = 0.1$ (dotted lines). We did not include the one corresponding to $\Delta Q_i = 0.01$ because it cannot be distinguished from the latter with the naked eye.

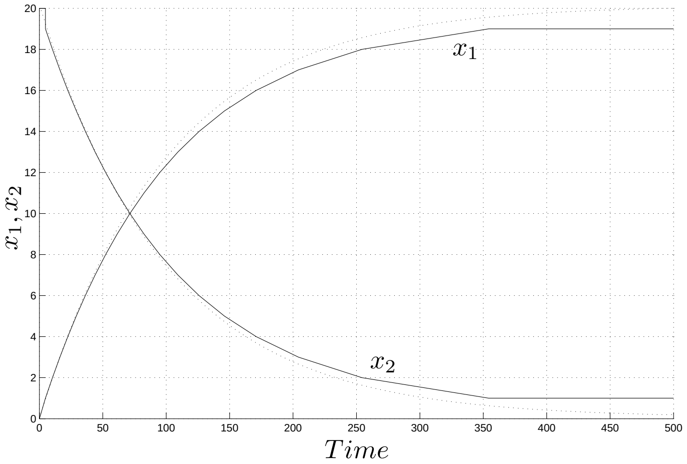

Figure 6: BQSS simulation of Eq.(3)

The theoretical error bounds, according to Theorem 3, for $\Delta Q_i = 1$ are

$$|e_1| \leq 3.004;\ |e_2| \leq 5.001 \tag{42}$$

while for $\Delta Q_i = 0.1$, they are 10 times smaller, and for $\Delta Q_i = 0.01$, they are 100 times smaller.

Integration Method	**Max,Mean Error**	**Number of Steps**	**Function f_i evaluations**
Power DEVS BQSS ($\Delta Q_{1,2} = 1$)	1.06,0.34	43	65
PowerDEVS BQSS ($\Delta Q_{1,2} = 0.1$)	0.199,0.034	408	612
PowerDEVS BQSS $\Delta Q_{1,2} = 0.01$)	0.0191,0.0033	4060	6098

Table 1: Results for the linear second order system

Table 6.1 summarizes the results. As in can be seen from comparing the first column, the analytical error bound turned out to be conservative in this example.

As the BQSS method does not introduce oscillations the number of changes performed in each variable can be obtained as the division between the signal amplitude and the quantum. This fact is corroborated in this example.

When the trajectories are not monotonic, the number of steps performed can be obtained adding the number of steps at each monotonic segment. This is equivalent to divide the signal activity [9] by the quantum.

6.2 Nonlinear third order stiff system.

The following system is a stiff standard test problem for ODE solvers [5]:

$$\begin{aligned} \dot{x}_1 &= -0.013x_1 - 1000x_1x_3 \\ \dot{x}_2 &= -2500x_2x_3 \\ \dot{x}_3 &= -0.013x_1 - 1000x_1x_3 - 2500x_2x_3 \end{aligned} \tag{43}$$

We consider initial conditions $x_1(0) = 1$ $x_2(0) = 1$ $x_3(0) = 0$.

In order to solve this system with BQSS, we first selected a quantization of $\Delta Q_1 = 0.01$, $\Delta Q_2 = 0.01$, and $\Delta Q_3 = 5 \times 10^{-8}$, together with a final time of $t_f = 500$.

Figure 7 shows the results. The simulation took 517 steps (100 in x_1, 102 in x_2, and 315 in x_3), arriving at a stable situation at time $t = 419.66$. PowerDEVS finished the calculations after 24 ms on a 2 GHz PC running Linux.

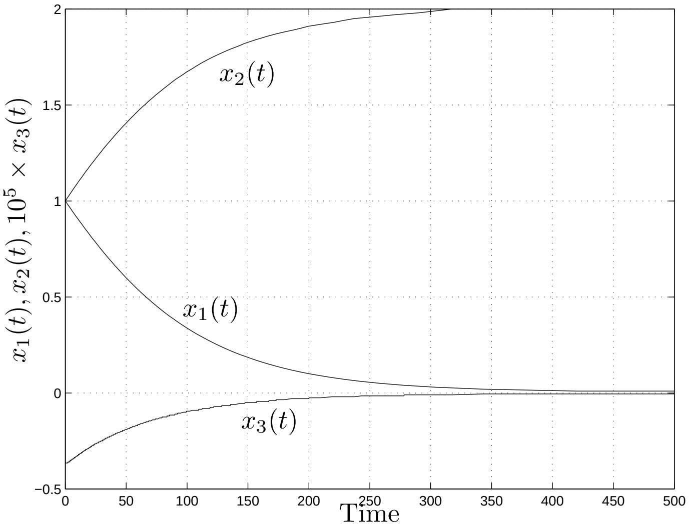

Figure 7: BQSS simulation of the system of Eq.(43)

These results obtained with BQSS have almost the same absolute error than those obtained with Matlab® ode15s and ode23s methods using a relative tolerance of 10^{-3}. Yet, BQSS requires less time to finish the calculations.

By reducing the quantum 10 times, the number of function evaluations grows about 10 times. However, the integration time increases less than three times. This suggest that in the first case, PowerDEVS spends most of the time with the initialization procedure.

Integration Method	**Mean Error**	**Number of Steps**	**Function f_i evaluations**	**CPU Time [sec]**
PowerDEVS BQSS ($\Delta Q_{1,2} = 0.01$ $\Delta Q_3 = 5 \cdot 10^{-8}$)	0.014	517	1349	0.024
PowerDEVS BQSS ($\Delta Q_{1,2} = 0.001$ $\Delta Q_3 = 5 \cdot 10^{-9}$)	$7.2 \cdot 10^{-4}$	4953	12860	0.057
PowerDEVS CQSS $\Delta Q_{1,2} = 0.01$ $\Delta Q_3 = 5 \cdot 10^{-8}$)	0.011	522	1363	0.026
Matlab ODE15s (To=10^{-3})	$7.1 \cdot 10^{-5}$	69	> 621	0.167
Matlab ODE23s (To=10^{-3})	$8.2 \cdot 10^{-5}$	59	> 531	0.088

Table 2: Results for the third order system

We then repeated the experiment with the F–stable CQSS method. The results obtained were similar (522 steps), but this time around, a stable situation was not reached. The simulation ended in slow oscillations around the final values of the three state variables.

Had we continued the simulation to a longer final time, CQSS would have produced more steps while BQSS performance remains identical. BQSS evidently performs better than CQSS in stiff systems, while CQSS performs better in marginally stable systems.

6.3 80th order marginally stable stiff nonlinear system.

The following system of equations represents a lumped model of a lossless transmission line where $L = C = 1$, with a nonlinear load at the end:

$$\begin{aligned}
\dot{\phi}_1(t) &= u_0(t) - u_1(t) \\
\dot{u}_1(t) &= \phi_1(t) - \phi_2(t) \\
&\vdots \\
\dot{\phi}_j(t) &= u_{j-1}(t) - u_j(t) \\
\dot{u}_j(t) &= \phi_j(t) - \phi_{j+1}(t) \\
&\vdots \\
\dot{\phi}_n(t) &= u_{n-1}(t) - u_n(t) \\
\dot{u}_n(t) &= \phi_n(t) - g(u_n(t))
\end{aligned} \tag{44}$$

We consider an input pulse entering the line:

$$u_0(t) = \begin{cases} 10 & \text{if } 0 \leq t \leq 10 \\ 0 & \text{otherwise} \end{cases} \tag{45}$$

and a nonlinear load:

$$g(u_n(t)) = (10000 \cdot u_n)^3 \tag{46}$$

We also set zero initial conditions $u_i = \phi_i = 0,\ i = 1, \ldots, n$.

We consider 40 LC sections (i.e., $n = 40$), which results in an 80^{th} order system. Linearization around the origin ($u_i = \phi_i = 0$) shows that the system is marginally stable (the linearized model does not have any damping term). However, a more careful analysis concludes that the system is also stiff (the nonlinear load adds a fast mode when u_n grows).

We decided to solve the system of Eqs.(44)–(46) using the F–stable CQSS method. To this end, we started with quanta of $\Delta Q_i = 0.1$ for all state variables except for u_n, where we applied $\Delta Q = 1 \times 10^{-4}$.

The quantum in QSS methods plays the role of the absolute tolerance. We reduced it at the last state because we wanted to observe the evolution of that variable with a bigger resolution since its value will be very close to zero.

To obtain the first 500 seconds of simulated time, CQSS performed roughly 6500 transitions in each of the state variables. The results of the first 100 seconds of the simulation are shown in Fig.8.

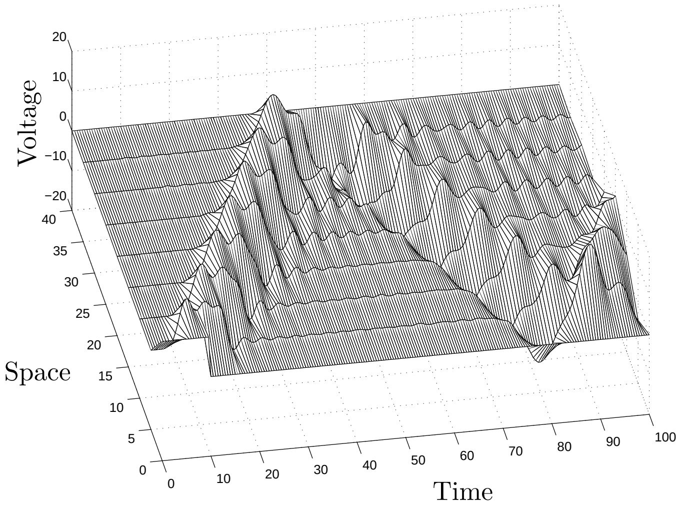

Figure 8: CQSS simulation of the system of Eqs.(44)–(46)

Figure 9 shows the voltage at the 35^{th} section of the transmission line (i.e., near the end with the load).

The results obtained with CQSS are similar to those obtained with the ode15s method of Matlab®, selecting a small error tolerance.

Regarding computational costs, CQSS (with $\Delta Q_i = 0.1$) completed the 500 seconds of simulated time in 8.66 seconds, performing around 6500 steps for each state variable. Every step changes the value of two components of the

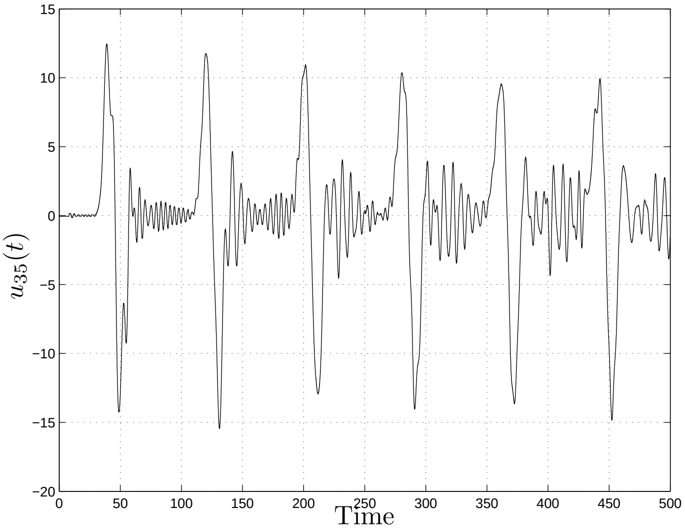

Figure 9: CQSS simulation of the system of Eqs.(44)–(46)

Integration Method	**Mean error**	**Number of Steps**	**Function f_i evaluations**	**CPU Time [sec]**
PowerDEVS CQSS ($\Delta Q_i = 0.1$)	0.133	497526	995052	3.94
PowerDEVS CQSS ($\Delta Q_i = 0.2$)	0.290	263506	527012	2.15
PowerDEVS CQSS ($\Delta Q_i = 0.5$)	0.833	144171	288342	1.23
Matlab ODE23t (To=10^{-3})	0.706	5551	>444080	8.80
Matlab ODE15s (To=10^{-3})	1.1498	5192	>415360	9.52

Table 3: Errors and work amount comparation for the transmission line

right hand side function of Eq.(44). Consequently, every component of that function is evaluated about 13000 times. In other words, the entire simulation involves about 13000 full function evaluations.

We then repeated the experiment with quanta of $\Delta Q_i = 0.2$. This reduced the number of calculations by roughly a factor of two (the simulation took 4.55 seconds), and the difference with the previous results can be hardly appreciated with the naked eye.

Using quanta of $\Delta Q_i = 0.5$, we observed an error of the order of 1.0 in the voltages, but the qualitative behavior is the same as before. In this case, the

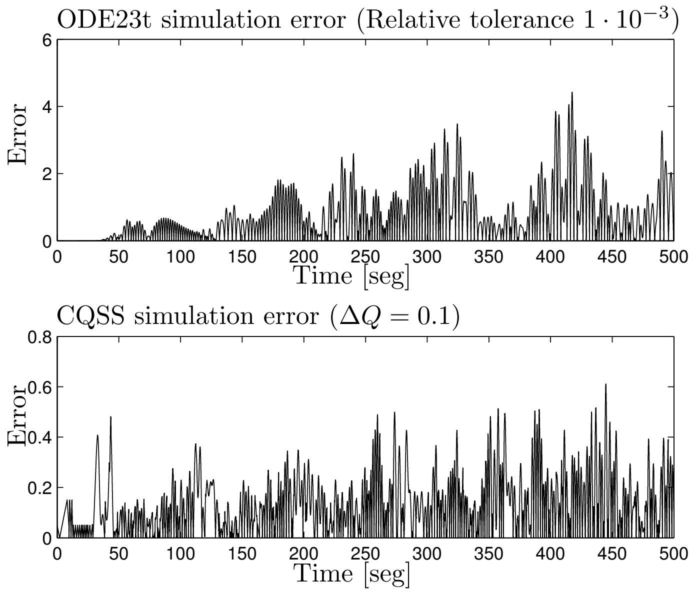

Figure 10: Simulation error in $u_{35}(t)$ result comparison

simulation took about 2.33 seconds.

We also simulated the system using different solvers offered by Matlab®. The best results were obtained with ode15s. For a relative tolerance of 1×10^{-3} and an absolute tolerance of 1×10^{-7}, the simulation took 9683 steps and was completed in 8.37 seconds.

The results obtained with ode15s are clearly more accurate than those obtained by CQSS. After all, ode15s is a variable-order algorithm that can increase its order of approximation accuracy up to fifth order, whereas CQSS is a first-order accurate method only.

Higher-order BQSS algorithms are currently under development. They are based upon the idea of back-interpolation [3], i.e., they integrate a higher-order non-stiff QSS algorithm backward through time using a leap-frogging approach.

Higher-order CQSS methods are also under development. They are based upon the theories of geometric ODE solvers introduced by Hairer et al. [8].

6.4 Buck Converter

The buck converter is a high efficiency step-down DC/DC switching converter. They are an integral part of many electronical circuits. The reason which are called step-down converters is because its output voltage may never be greater than the input. Given a continuous voltage unregulated, they produce an another one of lesser magnitude and regulated. Figure 11 shows the topology of this converter where the presence of a conmutation component Sw, a diode D (discontinuous component), an inductance L, a capacitor C and a resistance R

can be seen.

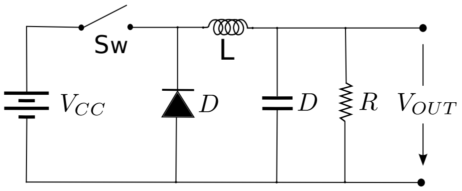

Figure 11: Buck converter scheme

Switching power converters are circuits discontinuous and nonlinear.

Assuming that the components are ideal, different models are obtained depending on the state of the switching components. Consider for example the case of changes in the state of the diode D while keeping open the state of the switch Sw. The corresponding models for both conditions in the state of the diode have different topologies. The model order corresponding to the situation with the diode state-on is different to the corresponding one with the diode state-off. As can be seen through this simple example, considering the switching components as real, result in models whith different structures.

In a simulation environment object-oriented (such as Dymola and other tools based on Modelica), such changes in structure presents a major problem. For this reason, this type of software add realism to its components thus achieving that there is no structural changes. For the example shown, if the switching components are modeled as a very low resistance when the state component is on and, a very high resistance if the state component is off, the following DAE (Differencial Algebraic Equation) model can be get:

$$\begin{array}{lcl} \dot{V}_C & = & \frac{I_L}{C} - \frac{V_C}{RC} \\ \dot{I}_L & = & \frac{V_D - V_C}{L} \\ V_D & = & R_D\left(\frac{V_{CC}-V_D}{R_{LL}} - I_L\right) \\ v_{out} & = & V_C \end{array} \tag{47}$$

where

$$R_{LL} = \begin{cases} R_{LL-on} & \text{if } Sw\text{–ON} \\ R_{LL-off} & \text{if } Sw\text{–OFF} \end{cases} \tag{48}$$

$$R_D = \begin{cases} R_{D-off} & \text{if } V_D > 0 (I_D \cdot R_D > 0) \\ R_{D-on} & \text{if } V_D \leq 0 (I_D \cdot R_D \leq 0) \end{cases} \tag{49}$$

This new system, but no changes in structure (only change parameters) is very stiff due to the presence of resistance very large or very small depending on the conmutation components states.

Figure 12 shows the simulation results of model (47) from the initial conditions $V_C(0) = 0$ and $I_C(0) = 0$, using the parameters of Table 6.4, and changing the state of the switch Sw with a frequency of 10kHz. The results shown in

Parameter	**Value**
R	10Ω
C	$1 \cdot 10^{-4}$ F
L	$1 \cdot 10^{-4}$ Hr
Ron Diode Roff Diode	$1 \cdot 10^{-6}$ $1 \cdot 10^{6}$
Ron Sw Roff Sw	$1 \cdot 10^{-6}$ $1 \cdot 10^{6}$
Switch Commutation Frequency	10 kHz
Vcc	24V

Table 4: Parameters of the Buck Circuit

this figure where simulated using de method BQSS (dQ $= \Delta Q_I = \Delta Q_V = 0.1$) on the PowerDEVS environment and in Simulink/Matlab(ODE23/Trapezoidal - Relative Tolerance $1 \cdot 10^{-3}$) using the PowerSIM library.

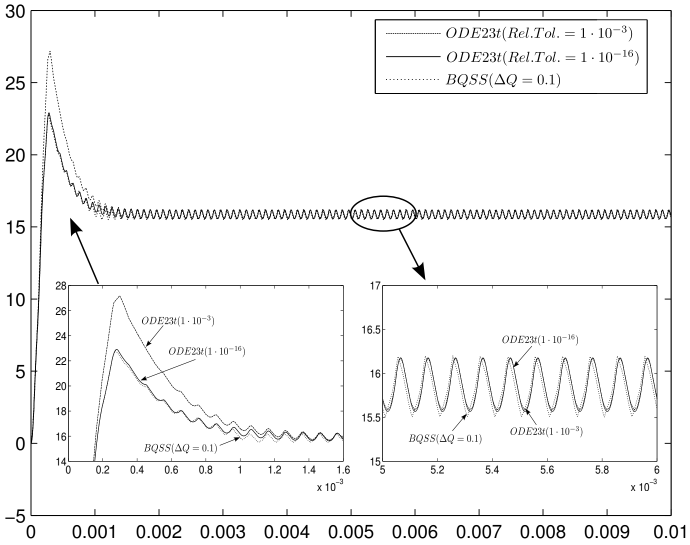

Figure 12: Simulation result comparison

In Table 6.4 the mean error together with the amount of work and CPU time (in seconds) are presented for several methods and tolerances and Fig.13 shows a comparation betwen error using diferent methods (using as a reference signal a simulation with low tolerance error)

Integration Method	**Mean Error**	**Number of Steps**	**Function f_i evaluations**	**CPU Time [sec]**
PowerDEVS BQSS ($\Delta Q_i = \Delta Q_V = 0.1$)	0.067	10431	20862	0.09
PowerDEVS BQSS ($\Delta Q_i = \Delta Q_V = 0.2$)	0.139	5386	10772	0.06
PowerDEVS BQSS ($\Delta Q_i = \Delta Q_V = 0.4$)	2.467	2935	5870	0.03
PowerDEVS CQSS ($\Delta Q_i = \Delta Q_V = 0.1$)	0.044	10027	20054	0.09
PowerDEVS CQSS ($\Delta Q_i = \Delta Q_V = 0.2$)	0.198	4857	9714	0.04
PowerDEVS CQSS ($\Delta Q_i = \Delta Q_V = 0.4$)	2.078	1968	3936	0.02
Matlab ODE23t (To=10^{-3})	0.199	> 2003	> 24036	0.65
Matlab ODE23t (To=10^{-4})	0.213	> 2327	> 27924	0.71
Matlab ODE23t (To=10^{-5})	0.033	> 3561	> 42732	0.81
LTSpice ModTrap (Tol=10^{-3})	0.209	11714	> 67000	0.53
Dymola esdirk23a (To=10^{-2})	0.019	734	8766	1.041

Table 5: Errors and work amount comparation for problem 47

7 Conclusions.

In this paper, new classes of stiff and geometric numerical ODE solvers were introduced that use state quantization instead of time-slicing as their discretization method.

Although state quantization leads invariably to non-linear solvers, it has been possible to develop theorems, using analysis of perturbations [10, 16], about accuracy, consistency, and numerical stability of these methods that are analogous to those developed for classical ODE solvers.

It was shown that QSS-based solvers share a number of striking properties that make it well worthwhile studying them as potentially interesting alternatives to classical ODE solvers.

In particular, it was shown that QSS-based “implicit” solvers can be implemented by explicit algorithms that don’t require any iteration.

The reason is that there are only two possible next state values that need to be investigated (one level up and one level down), i.e., in the worst of all cases, the computation of the next step needs to be repeated once, but this happens

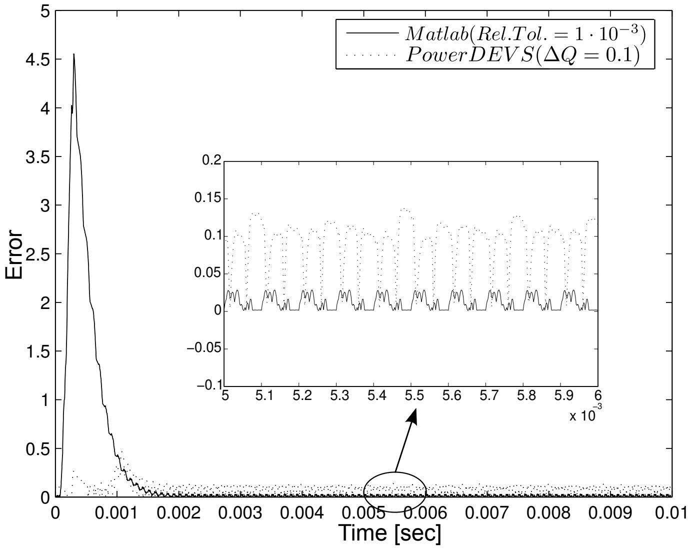

Figure 13: Absolute error

only, when the gradient changes its sign, which occurs rarely. At all other times, the additional cost of using a BQSS or CQSS algorithm over using the non-stiff QSS algorithm of first order is almost negligible (less than 10%). In contrast, classical implicit ODE solvers require on average three Newton iteration steps for each integration step.

BQSS was designed to solve general stiff systems. However, taking into account that QSS methods are intrinsically efficient to simulate discontinuous systems, we found that BQSS shows important advantages in discontinuous stiff ODEs, as those present in power electronic circuits. In these cases, for low accuracy settings, BQSS overperforms to all the stiff solvers available in Dymola, Matlab and PSpice.

The paper also proposed an interesting new test case simulating a linear transmission line without damping together with a non-linear load at one end. Compartmentalization of the transmission line led to an 80^{th} order system that is simultaneously marginally stable (due to the transmission line) and stiff (due to the non-linear load), a type of model that most classical numerical ODE solvers have difficulties dealing with.

The model was coded in Simulink®, and simulated in Matlab®. All available numerical ODE solvers were tried. All of the explicit solvers died because of the stiffness of the problem, and most of the stiff system solvers produced highly inaccurate results due to the marginal stability of the problem. They exhibited far too much numerical dissipation, and the oscillations died out rapidly. Only one of the ODE solvers, ode15s, was capable of producing a simulation result that comes acceptably close to reality.

The model was then recoded in PowerDEVS (the user interface of PowerDEVS is almost identical to that of Simulink®), and simulated trying all of the (QSS-based) numerical ODE solvers offered by PowerDEVS [1]. Whereas the three non-stiff algorithms (QSS, QSS2, and QSS3) failed to simulate this model (not surprisingly), both BQSS and CQSS completed the simulation quite rapidly. BQSS also exhibited too much numerical dissipation (the oscillations died out rapidly); CQSS solved the problem well.

In spite of the fact that CQSS is a first-order algorithm only, the code solved the test problem as rapidly as ode15s and with results that are comparable in accuracy for relaxed (relatively large) tolerance values. When more accurate results are desired, ode15s beats CQSS by leaps and bounds due to its higher order of approximation accuracy. Since CQSS is a first-order accurate method only, the number of state changes grows linearly in the quantization, ΔQ. In CQSS2, once available, the computational load will grow with the square root of the quantization, and in CQSS3, it will grow with the cubic root of the quantization.

Due to the non-linear and marginally stable nature of the problem, the BQSS algorithm was unable to produce simulation results that remained within the global error bound, but that error bound has been proven to hold for linear time-invariant analytically stable systems only.

References

[1] F. Bergero and E. Kofman. PowerDEVS. A Tool for Hybrid System Modeling and Real Time Simulation. *Simulation: Transactions of the Society for Modeling and Simulation International*, 2010. in Press.

[2] J. C. Butcher. Thirty Years of G–Stability. *BIT Numerical Mathematics*, 46(3):479–489, 2006.

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

[4] G. Dahlquist. Error Analysis for a Class of Methods for Stiff Nonlinear Initial Value Problems. In *Proc. Numer. Anal. Conf.*, volume 506 of *Lecture Notes Math.*, pages 60–74, Dundee, Scotland, 1975. Springer, New York, 1976.

[5] 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.

[6] C.W. Gear. The Automatic Integration of Ordinary Differential Equations. *Communications of the ACM*, 14(3):176–179, 1971.

[7] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer, Berlin, 1991.

[8] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration Structure-Preserving Algorithms for Ordinary Differential Equations.* Springer, 2002.

[9] Rajanikanth Jammalamadaka. Activity Characterization of Spatial Models: Application to Discrete Event Solution of Partial Differential Equations. Master's thesis, The University of Arizona, 2003.

[10] Hassan Khalil. *Nonlinear Systems.* Prentice-Hall, New Jersey, 2nd edition, 1996.

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

[12] E. Kofman. Discrete Event Simulation of Hybrid Systems. *SIAM Journal on Scientific Computing*, 25(5):1771–1797, 2004.

[13] E. Kofman. Non conservative ultimate bound estimation in LTI perturbed systems. *Automatica*, 41(10):1835–1838, 2005.

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

[15] E. Kofman and J. Braslavsky. Level Crossing Sampling in Feedback Stabilization under Data-Rate Constraints. In *Proceedings of CDC'06, IEEE Conference on Decision and Control*, pages 4423–4428, San Diego, 2006.

[16] E. Kofman, H. Haimovich, and M. Seron. A systematic method to obtain ultimate bounds for perturbed systems. *International Journal of Control*, 80(2):167–178, 2007.

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

[18] J.D. Lambert. *Numerical methods for ordinary differential systems: the initial value problem.* John Wiley & Sons, 1991.

[19] S. Mosbach and M. Kraft. An explicit numerical scheme for homogeneous gas-phase high-temperature combustion systems. *Combustion Theory and Modelling*, 10(1):171–182, 2006.

[20] J. Nutaro. A Second Order Accurate Adams-Bashforth Type Discrete Event Integration Scheme. In *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation*, pages 25–31, 2007.

[21] J. Nutaro and B. Zeigler. On the Stability and Performance of Discrete Event Methods for Simulating Continuous Systems. *Journal of Computational Physics*, 227(1):797–819, 2007.

[22] James Nutaro. *Parallel Discrete Event Simulation with Application to Continuous Systems*. PhD thesis, The University of Arizona, 2003.

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

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