Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
88(4) 387–407
© The Author(s) 2012
Reprints and permissions: sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0037549711403645
sim.sagepub.com

Quantization-based new integration methods for stiff ordinary differential equations

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

Abstract
In this paper we introduce new classes of numerical ordinary differential equation (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 primary result of the research described in this article is a first-order accurate QSS-based stiff system solver, called the backward QSS (BQSS). The numerical properties of this new algorithm are 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 collateral result, a first-order accurate QSS-based solver designed for solving marginally stable systems is briefly outlined as well. This new method, called the centered QSS (CQSS), is successfully applied to a challenging benchmark problem describing a high-order system that is simultaneously stiff and marginally stable. However, the primary emphasis of this article is on the BQSS method, that is, on a stiff system solver based on state quantization.

Keywords
discrete event system, quantized state systems, stiff systems

1. Introduction

Practically all of today's commonly used numerical ordinary differential equation (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, $x_{k+1} = x(t_{k+1})$.

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

Yet, all of these algorithms, be they linear or non-linear 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?

[1]Laboratorio de Sistemas Dinámicos, FCEIA - UNR - CIFASIS- CONICET, Argentina.
[2]Department of Computer Science, ETH Zurich, Switzerland.

Corresponding author:
Ernesto Kofman, Laboratorio de Sistemas Dinámicos, FCEIA-UNR-CIFASIS-CONICET, Rosario, Argentina
Email: ekofman@eie.fceia.unr.edu.ar

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 and Kim[1] and Zeigler and Lee,[2] constitutes the basis of a new class of numerical ODE solvers, namely the *quantized state system (QSS)* solvers.[3] 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, that is, 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.[4] The quantization process can be treated as a bounded perturbation on the original ODE. Thus, non-linear stability can be easily studied making use of Lyapunov functions.[3] The approach is related to that established by Dahlquist for analyzing the non-linear stability of conventional numerical solvers.[5,6]
4. QSS solvers offer a global rather than local error bound, that is, 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.[4]
5. The QSS approach allows the definition of explicit solvers that are nevertheless stiffly stable,[7] 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.[8]
7. QSS solvers are excellent at root solving, that is, 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 does not require any iteration.[9] 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),[10] because they can detect and locate discontinuities using explicit methods, and finally, because they allow one to minimize the communication bandwidth between separate agents dealing with different subsystems,[11] 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; 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, that is, 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.[8]

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

The open literature on time slicing methods for stiff and marginally stable systems contains hundreds of algorithms that provide different features and advantages.[8,12,13] 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 interval of real time. 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 the orders of one to four have been developed and have been reported in other publications[3,4,8,14–16]), this paper is the first to deal with the issues of stiffness and marginal stability. It introduces two new QSS algorithms: the backward QSS (BQSS), a stiff first-order ODE solver based on QSS technology, and the centered QSS (CQSS), another first-order QSS solver designed for solving marginally stable systems.

Second- and third-order accurate stiff solvers based on state quantization have been recently completed,[25] and higher-order geometric QSS solvers are currently under development.

2. Quantization-based integration

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

2.1. QSS method

Given the system

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

where $\mathbf{x} \in R^n$ 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:

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

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.[8] 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.[3] Consequently, the numerical solution of Equation (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, the QSS can efficiently handle discontinuities on the right-hand side of Equation (1).[9]

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. QSSs 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.

Figures 1 and 2 show the evolution of $q_1(t)$ and $q_2(t)$ across 500 units of simulated time.

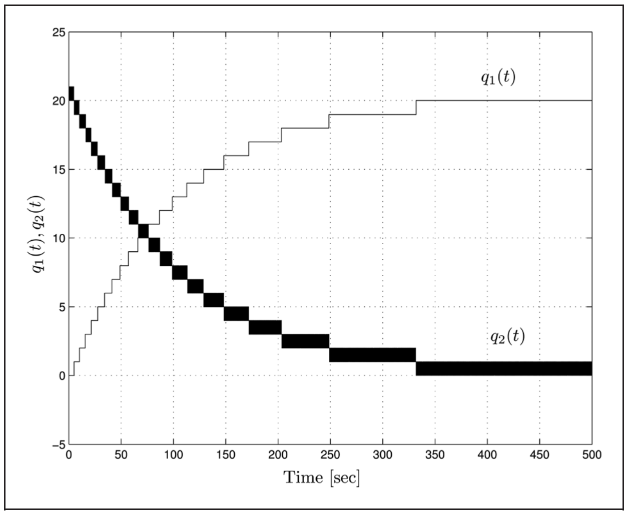

Figure 1. Quantized state system simulation.

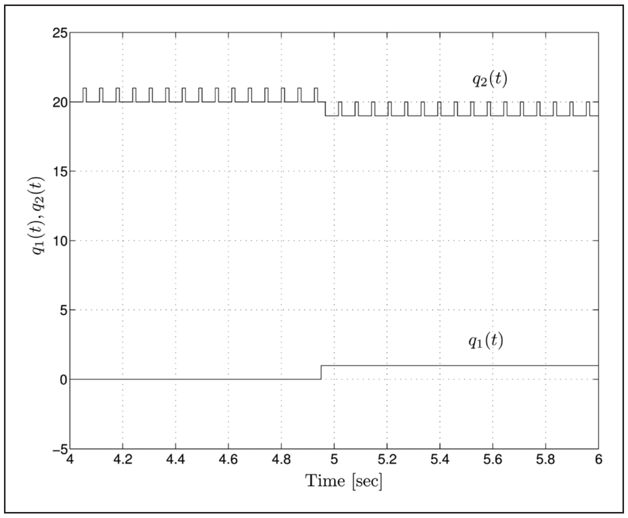

Figure 2. Quantized state system simulation (detail).

The fast oscillations of q_2 provoke a total of 15,995 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 16,000, that is, the number of simulation steps is of the same order of magnitude as the 25,000 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).

The 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 to 2000 in Equation (3), the number of steps is reduced to only 42. Another explicit method similar to the QSS was reported to exhibit good performance in some stiff combustion models.[18]

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 quantized state system

3.1. Basic Idea

The 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 the 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 $t = \infty$.

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,

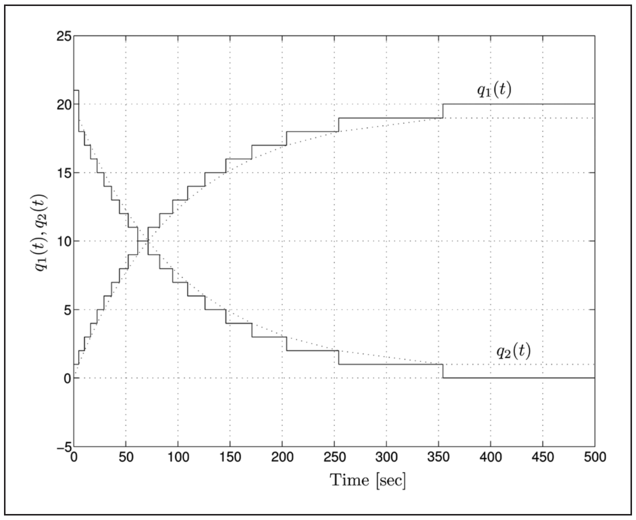

Figure 3. Backward quantized state system simulation.

which equals the performance of any implicit method (although the error might be quite 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.[3,8] 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

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

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

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

with

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

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

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 $f_j(\hat{\mathbf{q}}^{(j)}(t), \mathbf{u}(t)) = 0$:

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

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:

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

that is, 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 the QSS, ΔQ_j is called the quantum, and ε_j is the hysteresis width. Contrary to the 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 the QSS, the oscillation frequency of the BQSS does not grow linearly with decreasing values of ε_j. In the current implementation, the 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 the BQSS and the QSS is the way in which $\mathbf{q}$ is obtained from $\mathbf{x}$, as Equations (10) and (11) imply that the values of the different components are interrelated.

In the QSS, changes in q_j are produced when x_j differs from q_j by ΔQ_j. In the 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 Equations (10) and (11). In addition, 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 Equations (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, $D = \{\ldots, i, \ldots\}$ is the set of sub-indices of the functions f_i already evaluated. Similarly, 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 $D$.
>For each $i \in D$:
- define $\tilde{\mathbf{q}}^{(i)} \hat{=} \mathbf{q}(t^-)$;
- if $f_i(\tilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) \cdot (\tilde{q}^{(i)} - x_i) < 0$;
. include $i$ in $A$;
. Define
```

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

```
- Otherwise, $q_i(t) \hat{=} q_i(t^-)$.
1.b. If $x_i$ reaches $q_i$:
>include $i$ in $A$ and $D$;
>define $\tilde{\mathbf{q}}^{(i)} \hat{=} \mathbf{q}(t^-)$;
>calculate $q_i(t)$ according to Equation (14).
2. While $A \neq \phi$:
>let $j$ be the smallest element of $A$;
>define $B$ as the set of sub-indices $i \notin D$ so that $f_i$ depends on $q_j$.
>For each $i \in B$:
- define $\tilde{\mathbf{q}}^{(i)}$ according to
```

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

```
- if $f_i(\tilde{\mathbf{q}}^{(i)}, \mathbf{u}(t)) \cdot (\tilde{q}_i^{(i)} - x_i) < 0$;
. include $i$ in $A$;
. calculate $q_i(t)$ according to Equation (14);
- otherwise, $q_i(t) \hat{=} q_i(t^-)$;
>add the elements of $B$ to the set $D$ and remove $j$ from $A$.
3. For every $i \notin D$, 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 D only once. Thus, every component of function $\mathbf{f}$ is being evaluated at most once.

Theorem 1.

Consider the BQSS approximation of Equation (6) and suppose that $\mathbf{x}(t)$ and $\mathbf{q}(t^-)$ are known, and they satisfy

Equations (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 Equations (7)–(12).

Proof. For every $i \in D$, we define $\tilde{\mathbf{q}}^{(i)} \hat{=} \mathbf{q}(t^-)$.
We shall prove that an arbitrary component $q_j(t)$ satisfies Equations (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, Equations (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 Equations (7) is always satisfied.
To prove that Equations (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 Equation (14). Thus taking into account Equations (8) and (9), it will be true that

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

On the other hand if we set $q_j(t) = q_j(t^-)$, it is because Equations (16) is satisfied. Consequently, Equations (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
$f_j(\tilde{\mathbf{q}}^{(j)}(t), \mathbf{u}(t)) \cdot f_j(\mathbf{q}(t), \mathbf{u}(t)) \leq 0$
and from the mean value theorem, there exists $\hat{\mathbf{q}}^{(j)}$ between $\tilde{\mathbf{q}}^{(j)}(t)$ and $\mathbf{q}(t)$ such that $f_j(\hat{\mathbf{q}}^{(j)}, \mathbf{u}) = 0$.
Equations (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 $\tilde{\mathbf{q}}_i^{(j)}$ can only adopt the values $\overline{q}_i(t)$, $\underline{q}_i(t)$, or $x_i(t)$ (item 1.b). Then

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

From this equation and since $\hat{q}_i^{(j)}$ is between $q_i(t)$ and $\tilde{\mathbf{q}}_i^{(j)}(t)$, 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 the backward quantized state system

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 the BQSS

The following theorem ensures that the BQSS method cannot exhibit oscillatory behavior with a frequency approaching infinity, that is, 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 Equation (1) is bounded in a domain $D \times D_u$, where $D \subset R^n$, $D_u \subset R^m$ and assume that the trajectory $\mathbf{u}(t) \in D_u$ is piecewise constant. Then,
1. Any solution $\mathbf{x}(t)$ of Equation (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 Equation (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 Equations (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 Equation (9), this situation cannot be repeated until $|x_j(t) - x_j(t_c)| \geq \varepsilon_j$. Thus, the minimum time interval between two discontinuities in $\overline{q}_j(t)$ is

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

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

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

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 Equations (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, that is,

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

which is a finite number.

4.2. Perturbed representation

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

Each component of Equation (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 Equations (10)–(13), Equation (20) can be rewritten as

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

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

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

where

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

because from Equations (7)–(9) it results that

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

and from Equation (12) we have that

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

4.3. Stability and global error bound

In the linear time-invariant case, Equation (1) can be written as

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

and the BQSS approximation is

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

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

Theorem 3.

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

$$|\mathbf{e}(t)| \le |V| \cdot |\mathrm{Re}(\Lambda)^{-1} V^{-1}| \cdot |A| \cdot (\mathbf{\Delta Q} + \varepsilon) \tag{26}$$

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 Equation (25).[***]

Proof. According to Equation (22), the ith component of Equation (25) can be rewritten as

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

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

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

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

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

[*]The definitions and main theoretical properties of the QSS methods hold for both the linear and non-linear cases. However, the unconditional practical stability and error bound only apply to linear time-invariant systems.[4,8] In the BQSS, the situation is the same. All definitions and properties hold for both linear and non-linear systems, except for the unconditional practical stability and error bound.

[**]A square matrix is called Hurwitz when all its eigenvalues have a negative real part. A system such as that of Equation (24) is analytically stable if matrix A is Hurwitz.

[***]The symbol '$\le$' denotes here a componentwise relationship between two vectors. Similarly, '$|\cdot|$' denotes the matrix or vector of componentwise computed absolute values of the matrix or vector elements.

Resuming vector notation, Equation (27) can be written as

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

with

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

Replacing $\mathbf{x}(t)$ in Equation (24) by $\phi(t)$ and in Equation (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 Kofman[19] establishes the validity of Equation (26) from Equations (30) and (31). The same result can be derived for the non-diagonalizable case from Theorem 3.3 of Kofman et al.[20]

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

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

5. Towards F-stability[****]

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 using the QSS1 and BQSS with the parameter values $\omega = 1$, $\alpha = 0$ (a marginally stable case) and initial states $x_1 = 4$, $x_2 = 0$.

Both methods give qualitatively wrong results, as they are not F-stable.

Both the QSS1 and BQSS 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

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

[****]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 [Cellier and Kofman]. F-stability has also been referred to as precise A-stability in some references.[17]

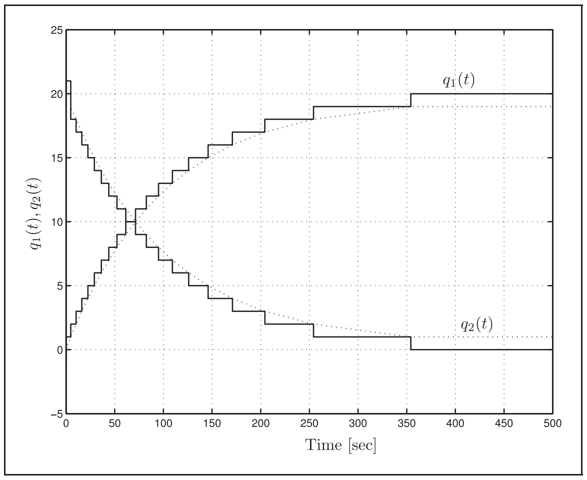

Figure 4. First order QSS (QSS1) and backward quantized state system (BQSS) simulations of a marginally stable system.

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

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

with

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

In this case

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

From the last equation and Equation (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 Equations (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 function

$$V(x) \hat{=} \frac{1}{2}\left(x_1^2 + x_2^2\right). \tag{39}$$

with the time derivative

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

Thus,

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

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 Equation (40) holds. Then, both methods, the QSS1 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 $\alpha = 0$, that is, when the eigenvalues are located on the imaginary axis. In this case, it can be seen that $\Delta f_i = 0$ in the BQSS whenever $PxP > 2\Delta Q_1 + 2\Delta Q_2$. 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 the 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 of marginally stable systems using the BQSS will produce practically stable results as V (and hence $\|x\|$) decreases with time.

For the QSS1, we need to evaluate V between two instants of time:

$$\begin{aligned}V(t_b)-V(t_a) &= \int_{t_a}^{t_b}\dot{V}(x)dt = \int_{t_a}^{t_b}(-\dot{x}_1\Delta x_1 - \dot{x}_2\Delta x_2)dt\\ &\hat{=} -\Delta V_1 - \Delta V_2\end{aligned}$$

Let us call $t_1, t_2, \ldots, t_m$ 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

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

In the QSS1, we have $q_1(t_k) = x_1(t_k)$, and the integral evaluates to $-\Delta Q_1^2/2$. Hence the QSS1 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 $q_i(t_k) = 0.5(x_i(t_{k+1}) + x_i(t_k))$, that is, by taking the mean value between the QSS1 and BQSS values for q.

This very simple idea defines a new method that shall be called the CQSS method. Figure 5 shows the simulation of the marginally stable system of Equation (32) using CQSS.

6. Examples

The new BQSS and CQSS methods were programmed in PowerDEVS, a discrete event system (DEVS) simulation engine that has already implemented the QSS$_i$ family of numerical integration methods.[21] 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 Equation (3) with initial conditions $x_1(0) = 0$, $x_2(0) = 20$. This time, we wish to solve the system until $t = t_f = 1000$.

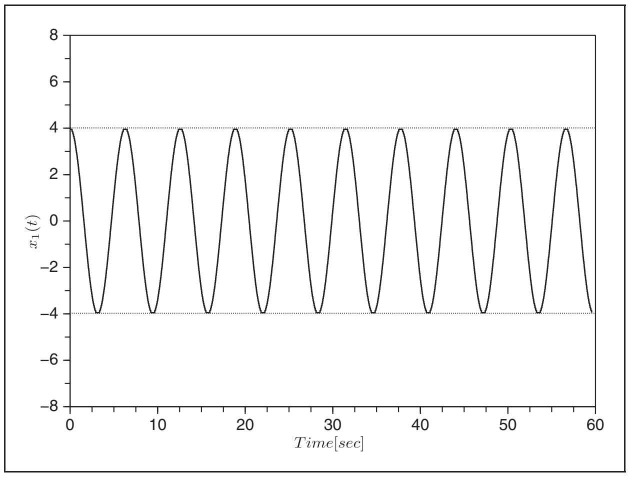

Figure 5. Centered quantized state system simulation of a marginally stable system.

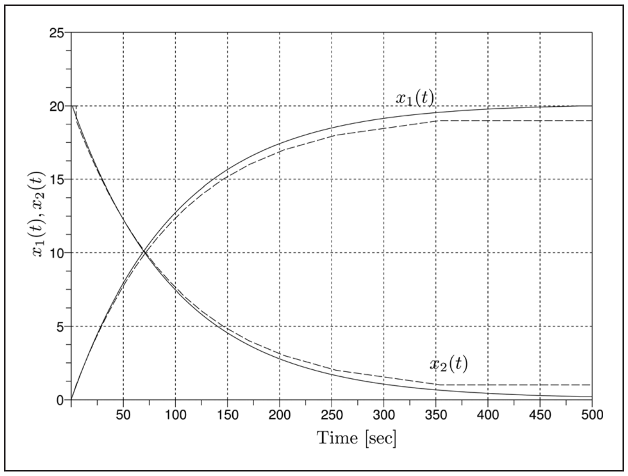

Figure 6. Backward quantized state system simulation of Equation (3).

Table 1. Results for the linear second-order system

Integration method	Max, mean error	Number of steps	Function f_i evaluations
PowerDEVS 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

BQSS: backward quantized state system.

We first simulated the introductory experiment using the BQSS with a quantum of $\Delta Q_1 = \Delta Q_2 = 1$. Then, we repeated the simulation 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, respectively.

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

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 simulation trajectories obtained with $\Delta Q_i = 1$ (dashed lines) and $\Delta Q_i = 0.1$ (solid lines). We did not include the simulation results corresponding to $\Delta Q_i = 0.01$, because they cannot be distinguished from the latter by the naked eye.

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.

Table 1 summarizes the results. As can be seen by comparing the results reported in the error column (column #2) with the theoretical error bounds established earlier, the error bounds 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 by adding the number of steps during each monotonic segment. This is equivalent to dividing the signal activity[22] by the quantum.

6.2. Non-linear third-order stiff system

The following system is a stiff standard benchmark problem for ODE solvers:[23]

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

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

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

Figure 7 shows the simulation trajectories. The trajectory of $x_3(t)$ exhibits initially a very fast negative gradient that cannot be discerned at the resolution of the figure. The figure leaves the impression that $x_3(t)$ is experiencing a discontinuity at time zero, which is, of course, not the case.

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 $t = 419.66$. After that time, no further events take place.

PowerDEVS finished the calculations after 24ms on a 2GHz personal computer (PC) running Linux.

When the quantum is reduced tenfold, the number of function evaluations grows about 10 times. However, the integration time increases by less than a factor of three. This suggests that, in the original simulation, PowerDEVS spends most of its time in the initialization procedure.

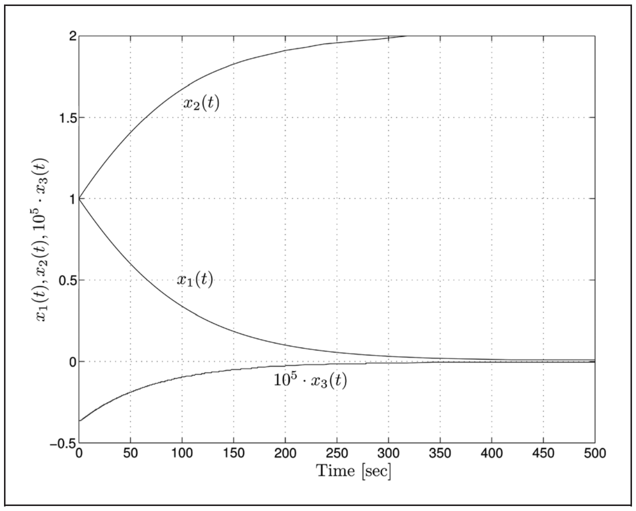

Figure 7. Backward quantized state system simulation of the system of Equation (43).

We then repeated the experiment with the F-stable CQSS method and with the same quantum as used in the original simulation. The results obtained were quite similar to those obtained by the BQSS (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 this simulation to a longer final time, the CQSS would have produced additional events due to the steady-state oscillations, whereas the BQSS would not. Both solvers are useful for simulation, but their strengths lie in different types of applications. The BQSS performs better than the CQSS in stiff system simulations, whereas the CQSS performs better than the BQSS in marginally stable system simulations.

For comparison purposes, we then performed the same simulations using the stiff ode15s and ode23s solvers of Matlab. For these simulations, we requested a relative error tolerance of 10^{-3}, that is, the same tolerance as in the more accurate second simulation run using the BQSS.

Table 2 summarizes the results. The errors shown (column #2) in each case are the largest of the three mean absolute errors for the three state variables.

We notice that the BQSS generated results that were about as accurate as requested, that is, with a quantum of $\Delta Q_1 = \Delta Q_2 = 0.01$, we obtained results with an absolute global error of approximately 0.01, whereas with a quantum of $\Delta Q_1 = \Delta Q_2 = 0.001$, we obtained results with an absolute global error of approximately 0.001. The errors were calculated by comparing the simulation results with a reference solution obtained by simulating the model once more using ode15s with a relative error tolerance of 10^{-9}.

If we compare the number of steps used by the second BQSS simulation (with a quantum of $\Delta Q_1 = \Delta Q_2 = 0.001$) with those used by ode15s and ode23s, we notice that the BQSS requires between 100 and 200 times as many steps. This much larger number

Table 2. Results for the third-order system

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	12,860	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 (Tol. $= 10^{-3}$)	$7.11 \cdot 10^{-4}$	38	> 342	0.087
Matlab – ode23s (Tol. $= 10^{-3}$)	$8.42 \cdot 10^{-4}$	22	> 198	0.049

CPU: central processing unit, BQSS: backward quantized state system, CQSS: centered quantized state system.

of steps is partly caused by the fact that the BQSS is an asynchronous solver, that is, each integration step updates only a single state variable, whereas the Matlab solvers are synchronous solvers that update all three state variables simultaneously. Thus, the BQSS uses in reality only about 50 times as many steps as the two Matlab solvers. The main reason for the much larger number of steps is, however, caused by the fact that the BQSS is a first-order algorithm, whereas the Matlab solvers are higher-order algorithms.

We already noticed when developing and testing the non-stiff QSS solvers that the number of steps required by these solvers is comparable to the number of steps required by classical non-stiff ODE solvers of the same order.[9] We expect that the same will hold also for the stiff solvers, that is, once the higher-order stiff QSS solvers have been fully developed and debugged, they also will use a similar number of steps as the corresponding Matlab solvers.

The number of individual function evaluations can be directly measured in PowerDEVS. In the case of Matlab, we can only obtain an estimate of a lower bound by multiplying the number of steps with the order of the system (three in this case) and with the minimum number of function evaluations per step of the method. Both methods need at least three function evaluations per step, assuming that the Newton iteration converges after only one iteration (i.e. involving two function evaluations), and an extra function evaluation for step-size control.

Yet in spite of the fact that the BQSS is a first-order stiff ODE solver only and in spite of the fact that this solver requires many more steps to complete the simulation, the BQSS simulation runs almost as fast (0.057 seconds) as the more efficient among the two Matlab simulations (0.049 seconds), and the results have a similar accuracy.

The execution times of the two codes are meaningfully comparable, because both Matlab and PowerDEVS partly compile their solvers and/or model equations, but neither of the two codes compiles the entire simulation into a flat simulation code that can then be executed, that is, both codes are partly compiled and partly interpreted.

Of course, the BQSS will become hopelessly inefficient when the accuracy requirements are tightened. For example, if we request a relative accuracy of 10^{-6}, the Matlab solvers will still simulate the problem adequately, whereas the BQSS will take forever to complete the simulation, since the number of steps in the BQSS grows inversely proportional to the quantum.

In order to simulate stiff systems with stringent accuracy requirements using a QSS-based stiff system solver, we shall need to wait for the higher-order codes that are currently under development.

6.3. 80th-order marginally stable stiff non-linear system

The following system of equations represents a lumped model of a lossless transmission line, where $L = C = 1$, with a non-linear 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:

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

and a non-linear 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 80th-order system. Linearization around the origin ($u_i = \phi_i = 0$) shows that the system is marginally stable, that is, the linearized model does not have any damping term. However, a more careful analysis concludes that the system is also stiff, as the non-linear load adds a fast mode when u_n grows.

We decided to simulate the system of Equations (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 finer resolution, since its value will be very close to zero.

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

Figure 9 shows the voltage at the 35th section of the transmission line, that is, towards the end with the load.

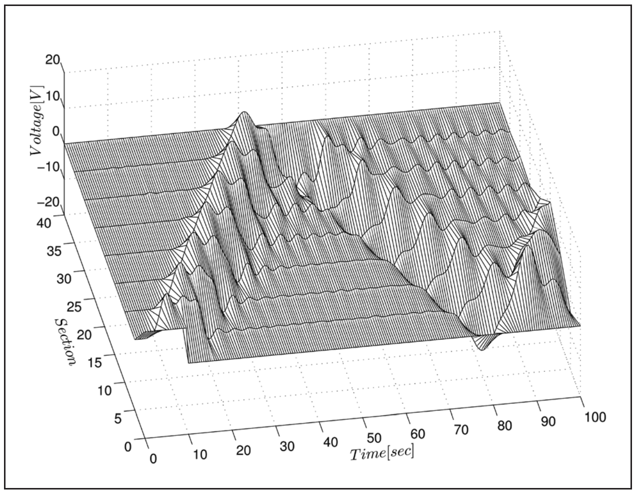

Figure 8. Centered quantized state system simulation of the system of Equations (44)–(46).

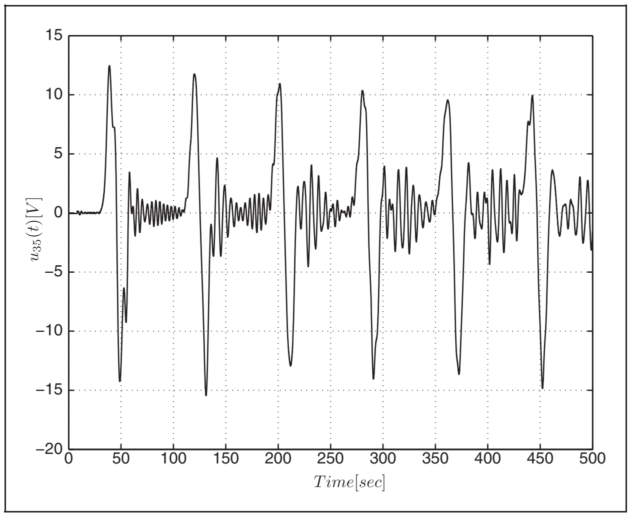

Figure 9. Centered quantized state system simulation of the system of Equations (44)–(46).

In order to be able to analyze and discuss the accuracy and efficiency of the CQSS solution, we repeated the simulation twice with larger values of the quanta, and we also performed simulations of the same model in Matlab using the two ODE solvers that worked best on this example: ode23t, an implementation of the F-stable trapezoidal rule, and ode15s, a variable-step, variable-order stiff backward differentiation formula (BDF) solver.

Table 3 summarizes the results. The error given in the table (column #2) refers to the average absolute error of the voltage of the 35th transmission line section, that is, the same variable that is shown in Figure 9.

The absolute errors obtained by the CQSS simulations were approximately those requested through the selection of the quanta: for $\Delta Q_i = 0.1$, we obtained an average absolute error of 0.133; with $\Delta Q_i = 0.2$, we achieved an average absolute error of 0.29; and setting $\Delta Q_i = 0.5$, the average absolute error turned out to be 0.833. The Matlab solutions were a little less accurate, but much more importantly, the CQSS, by controlling the global absolute error, provides the user with a more finely tuned error control than either ode23t or ode15s, which only control the local relative error. The errors were calculated by comparing the simulation results with a reference solution obtained by simulating the model once more using ode23t with a relative error tolerance of 10^{-9}.

We notice further that the CQSS simulations were, for this example, significantly more efficient than the more efficient among the two Matlab solutions.

In order to gain a yet better understanding of the errors in these simulations, we plotted the absolute errors of the voltage of the 35th transmission line section, simulated using the most accurate of the CQSS and Matlab solutions, as functions of time. These results are depicted in Figure 10.

Notice that the largest absolute errors are about 10 times larger for ode23t than for the CQSS. In addition, the average errors for both ode23t and ode15s are slowly growing over time, whereas the average errors of the CQSS simulation remain constant over time.

Table 3. Accuracy and efficiency analysis for the transmission line

Integration method	Mean error	Number of steps	Function f_i evaluations	CPU time [sec]
PowerDEVS CQSS ($\Delta Q_i = 0.1$)	0.133	497,526	995,052	3.94
PowerDEVS CQSS ($\Delta Q_i = 0.2$)	0.290	263,506	527,012	2.15
PowerDEVS CQSS ($\Delta Q_i = 0.5$)	0.833	144,171	288,342	1.23
Matlab ode23t (Tol. $= 10^{-3}$)	0.706	5551	$>$ 1,332,240	8.80
Matlab ode15s (Tol. $= 10^{-3}$)	1.1498	5192	$>$ 1,246,080	9.52

CPU: central processing unit, CQSS: centered quantized state system.

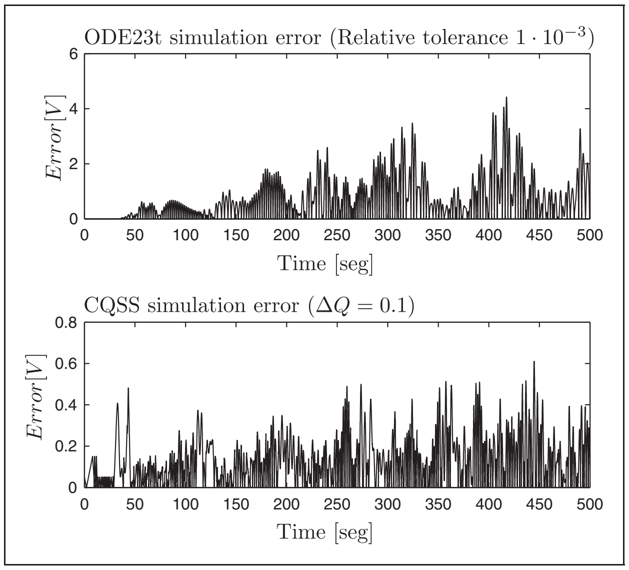

Figure 10. Absolute simulation error in $u_{35}(t)$.

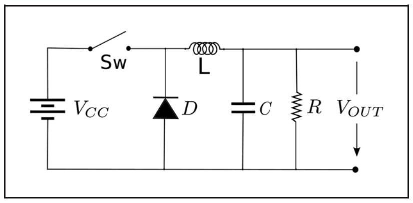

Figure 11. Buck converter diagram.

6.4. Buck converter

The buck converter is a highly efficient step-down direct current/direct current (DC/DC) power-switching converter. Buck converters form an integral part of many electronic circuits. The reason why they are called step-down converters is because their output voltage can never exceed their input voltage. Given an unregulated continuous voltage, they produce a regulated output voltage of smaller magnitude. Figure 11 shows a circuit diagram of a buck converter containing two non-linear elements: a switch, Sw, and a diode, D.

Models of ideal switching power converters are non-linear, discontinuous, variable structure models. A variable structure model is a model, in which the state variables change as a function of a switch parameter. It can be seen easily that the ideal buck converter circuit is represented by a variable structure model. To this end, consider the case where the switch is open. Then, when the diode is conducting, current flows through the inductor, whereas when the diode is blocking, no current flows through the inductor. Consequently, the inductor current can be used as a state variable of the model while the diode is conducting, but it cannot be used as a state variable while the diode is blocking.

To avoid the problem of having to deal with a variable structure model, many modelers make use of leaky diodes and non-ideal switches, that is, they represent closed switches and conducting diodes by a very small resistance, and they represent open switches and blocking diodes by a very large resistance. Consequently, both elements can be modeled as resistors with resistance values that vary over time. This is also the approach that we shall be using in this example.

In this way, we avoid the variable structure problem, and we can obtain, for example, the following differential algebraic equation (DAE) model describing this circuit:

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

where

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

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

The form of the DAE model obtained is not unique. It defines the algebraic variable, V_D, implicitly, that is, the model contains an algebraic loop, and consequently, that equation can be formulated in many different ways.

Figure 12 shows how this model has been implemented using the graphical block diagram editor of PowerDEVS. The hysteretic quantized integrator block, QSS Integrator1, computes the inductive current, I_L. Its derivative is a linear-weighted sum of V_D and V_C and is computed by the block WSum1. The second integrator block, QSS Integrator2, computes the capacitive voltage, V_C. Its derivative is a linear-weighted sum of I_L and V_C and is computed by the block WSum2. The implicitly formulated algebraic variable, V_D, is computed by the block ImpFunction1. It is a non-linear function of the switch resistances, R_{LL} and R_D, and of the inductive current, I_L. The top switch block, Switch1, computes the switch resistance, R_{LL}, whereas the bottom switch block, Switch2, computes the diode resistance, R_D. R_D is a function of V_D. In order to avoid an illegitimate model, that is, oscillations occurring with infinite frequency, a small hysteresis was introduced between the block that computes V_D and its use by the switch block, Switch2. Table 4 lists the parameter values used by the model.

It must be noted that modeling the switch and the diode by non-ideal switch elements to avoid the variable structure problem comes at a price. The resulting model is very stiff in some of the switch positions, and the smaller the value of R_{on} and the larger the value of R_{off} are chosen, the more pronounced the (artificial) stiffness of the model will turn out to be.

Consequently, we need a stiff system solver to simulate this discontinuous model, and therefore, we chose the BQSS for simulating this circuit. The integration algorithm to be used can be set in PowerDEVS as a parameter of the hysteretic quantized integrator block, and it is set for each integrator independently, that is, PowerDEVS offers a convenient way to specify mixed-mode integration.

Figure 13 shows the output voltage, V_{out}, of model (47) starting from zero initial conditions, $V_C(0) = 0$

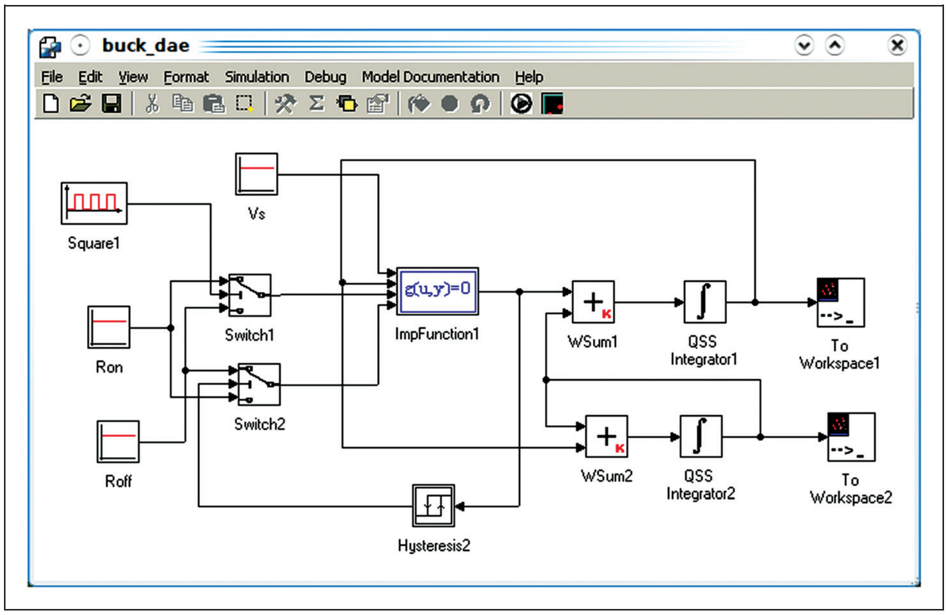

Figure 12. PowerDEVS model of buck converter.

Table 4. Parameters of the buck converter circuit

Parameter	Value
R	10 Ω
C	100 μF
L	0.1 mH
R_{on} Diode	10^{-6}
R_{off} Diode	10^{6}
R_{on} Switch	10^{-6}
R_{off} Switch	10^{6}
Switch commutation Frequency	10 kHz
V_{cc}	24 V

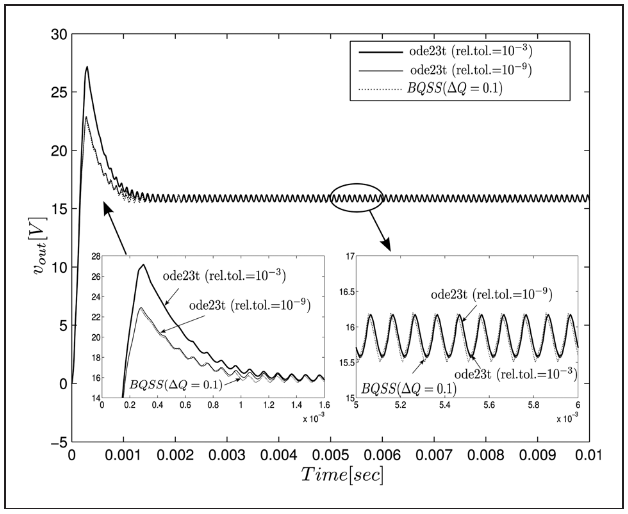

Figure 13. Buck converter simulation results. BQSS: backward quantized state system.

and $I_L(0) = 0$, and using the parameters of Table 4. The position of the externally controlled switch is toggled with a frequency of 10 kHz.

The simulation was run thrice. The first simulation used the BQSS with $\Delta Q_1 = \Delta Q_2 = 0.1$ for the two integrators. In PowerDEVS, ΔQ is specified as a parameter of the hysteretic quantized integrator block. The second simulation was run in Matlab using ode23t with a relative error tolerance of 10^{-3}. The third simulation run generated the reference solution. It was run also in Matlab using ode23t with a relative error tolerance of 10^{-9}.

We notice that the solution produced by ode23t deviates significantly from the reference solution during the initial phase of the simulation, whereas the solution produced by the BQSS is quite accurate. On the other hand, once the (oscillatory) steady state has been reached, the ode23t solution is more accurate than that produced by the BQSS.

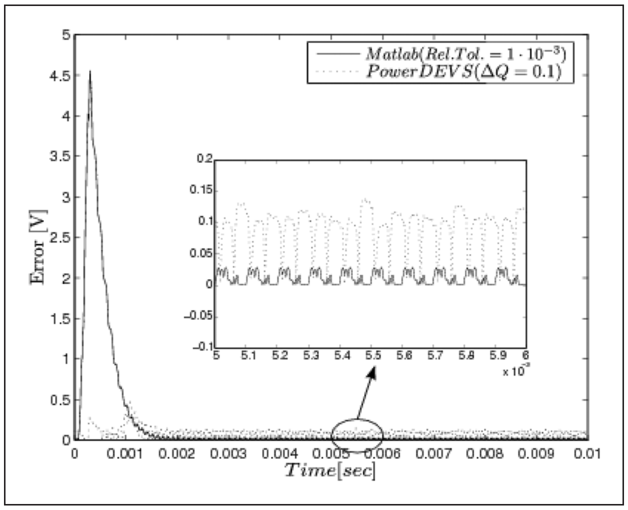

Figure 14. Absolute errors of buck converter simulations.

Figure 14 depicts the absolute errors committed by the two simulations, that is, by the BQSS and ode23t. They were computed by comparing the simulation results with the reference solution.

The error committed by ode23t during the initial simulation phase is huge. The simulated output voltage, V_{out}, deviates from the reference solution by roughly 20%, although a relative error tolerance of 0.001 had been requested. Thus, the actual simulation error is too large by a factor of 200. During the (oscillatory) steady state, the absolute error committed by ode23t is of the order of 0.015, whereas the output itself assumes a value of roughly 15. Hence ode23t performs as requested during steady-state operation.

In contrast, the absolute error committed by the BQSS remains of the order of 0.1 throughout the entire simulation, as requested by the setting of the ΔQ_i values.

We notice once again that the BQSS does a much better job than ode23t in controlling the error accurately. Neither does the BQSS waste precious computing resources by calculating a solution that is more accurate than requested, nor does it deceive the user by producing simulation results that are less accurate than demanded.

In Table 5, the mean errors of the output voltage, V_{out}, from a number of simulations are tabulated together with the execution times that these simulations required.

In addition to the two simulations mentioned earlier, we performed an additional simulation using the BQSS with larger quanta of $\Delta Q_1 = \Delta Q_2 = 0.2$, two simulations using the CQSS with different quanta, a series of simulations using Matlab's ode23t simulator with different relative error tolerances, a comparative simulation using LTSpice, a version of the Spice electronic

Table 5. Accuracy and efficiency of buck converter circuit simulations

Integration method	Mean error	Number of steps	Function f_i evaluations	CPU time [sec]
PowerDEVS BQSS ($\Delta Q_1 = \Delta Q_2 = 0.1$)	0.067	10,431	20,862	0.09
PowerDEVS BQSS ($\Delta Q_1 = \Delta Q_2 = 0.2$)	0.139	5386	10,772	0.06
PowerDEVS CQSS ($\Delta Q_1 = \Delta Q_2 = 0.1$)	0.044	10,027	20,054	0.09
PowerDEVS CQSS ($\Delta Q_1 = \Delta Q_2 = 0.2$)	0.198	4857	9714	0.04
Matlab ode23t (Tol. $= 10^{-3}$)	0.199	> 2003	$> 24,036$	0.65
Matlab ode23t (Tol. $= 10^{-4}$)	0.213	> 2327	$> 27,924$	0.71
Matlab ode23t (Tol. $= 10^{-5}$)	0.033	> 3561	$> 42,732$	0.81
LTSpice ModTrap (Tol. $= 10^{-3}$)	0.209	11,714	$> 67,000$	0.53
Dymola esdirk23a (Tol. $= 10^{-2}$)	0.019	734	8766	1.041

CPU: central processing unit, BQSS: backward quantized state system, CQSS: centered quantized state system.

circuit simulator specifically designed for the efficient simulation of digital circuits, such as switching power converter circuits, and finally, we also simulated this same circuit in Dymola, an object-oriented modeling and simulation environment designed for modeling and simulating physical systems.

We notice that, for this example, the performance of the BQSS and the CQSS is very similar, both in terms of accuracy and efficiency. We notice further that the execution times of ode23t grow insignificantly when tightening the relative error tolerance from 10^{-3} to 10^{-5}, but we also notice that the solver does not deliver either, that is, the mean errors should get reduced by a factor of 100, which is not at all the case. Another observation is that LTSpice, although specialized for the simulation of precisely this type of electronic circuit, is about as equally efficient as Matlab with ode23t. We finally observe that Dymola, when using esdirk23a, simulates the circuit very accurately, but is also the least efficient of the tools. Dymola is slower than Matlab and/or LTSpice by almost a factor of two. We tried all solvers that Dymola offers and reported the results for the solver that simulated this circuit most effectively.

Although the BQSS and the CQSS are only first-order accurate solvers, both turned out to be highly competitive in the simulation of the buck converter circuit. Both solvers are faster than Matlab and LTSpice by at least a factor of five, and they beat Dymola in efficiency by at least a factor of 10.

Of course, our accuracy requirements were set relatively low. We had to compute the reference solution in all four examples using Matlab, because neither the BQSS nor the CQSS would be able to calculate a highly accurate solution efficiently. When the quanta are reduced by a factor of 10, the number of events for both the BQSS and the CQSS will increase by roughly a factor of 10, and consequently, the simulation will also take 10 times longer to execute.

We shall have to wait for higher-order stiff and marginally stable QSS-based solvers to remedy this shortcoming. In a second-order QSS solver, the number of events grows inversely proportional to the square root of the quantum, and in a third-order accurate QSS solver, the number of events grows inversely proportional to the cubic root of the quantum. Hence by using a third-order stiff QSS solver, we shall be able to reduce the quantization by a factor of 1000, and yet will only have to wait about 10 times as long for the simulation to end.

7. Conclusions

In this paper, a new stiff numerical ODE solver has been presented that bases its discretization on state quantization instead of time slicing. The BQSS method, as described in this article, is only first-order accurate, and therefore, the accuracy obtainable by BQSS simulations is evidently quite limited, but an extension to higher orders of approximation accuracy has already been proposed in a PhD dissertation,[17] and solvers based on this idea are currently under development.

The numerical properties of the BQSS, that is, its stability and convergence properties, were rigorously analyzed. Although state quantization leads invariably to non-linear solutions, even when applied to the simulation of linear time-invariant systems, it has been possible to develop theorems, using analysis of perturbations,[20,24] about the 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 do not require any iteration.

The reason for this surprising discovery is that there are only two possible next state values that need to be investigated (one level up and one level down), that is, in the worst of all cases, the computation of the next step would need to be repeated only once. Yet, since QSS-based algorithms offer a naturally dense output, a decision, which of the two possible solutions must be taken, can be made beforehand, that is, without actually performing a simulation step. The additional computational cost for making this decision is quite low. Comparing the computational effort of calculating one step of the stiff 'implicit' BQSS solver with computing one step of the non-stiff 'explicit' QSS1 solver, we find that simulating a step of the BQSS is only about 10% more expensive than computing a step of QSS1. In contrast, classical implicit ODE solvers require on average three Newton iteration steps for each integration step.

In classical ODE solvers, step-size control is performed by comparing the simulation results of two integration algorithms that are simulated in parallel with each other. These algorithms control the local integration error during a single step, and if it turns out that a step was calculated too inaccurately, the step needs to be repeated with a smaller step size. This is true for both explicit and implicit ODE solvers.

QSS-based algorithms operate differently. Here, step-size control is a natural feature of the algorithms themselves. It was shown that the global integration error is a function of the quantum chosen, and the step size of these methods is adapted for each state variable separately in accordance with its current gradient and the selected quantization. Thus, QSS-based algorithms never need to repeat an integration step.

As an almost accidental by-product, a second algorithm, the CQSS, was also introduced in this article. The CQSS solver is a mixture between the non-stiff QSS1 solver and the stiff BQSS solver. It turns out that the CQSS is a true geometric solver.[13] It shares all of the properties of geometric solvers. The algorithm is symmetric, symplectic, and preserves ρ-reversibility.[13] In addition, the CQSS is only first-order accurate. Hence, the CQSS is ideally suited for the simulation of conservative (frictionless) mechanical systems at low-to-moderate accuracy.

The properties of the CQSS were, however, not presented as rigorously as those of the BQSS algorithm. The reason is that the geometric nature of the CQSS seems to be a bit of an accident. At least until now, we do not see any way to extend the geometric properties of the CQSS to higher orders of approximation accuracy. Of course, it is possible to design higher-order QSS-based algorithms that will be decently well suited for the simulation of marginally stable systems, for example, by toggling steps of the already developed higher-order non-stiff algorithms, QSS_i with steps of the higher-order stiff algorithms that are currently under development, but these cyclic algorithms do unfortunately not preserve the geometric properties of the CQSS in a rigorous way.

The more theoretical discussions of the properties of the BQSS (and to a lesser extent, the CQSS) are accompanied in this paper by a more practical section, in which the performance of the BQSS and the CQSS is compared against the performance of classical stiff and marginally stable system solvers. Comparisons were made with ODE solvers available as part of Matlab, in particular ode15s, ode23s, and ode23t, with LTSpice, a dialect of the widely used Spice electronic circuit simulator, designed specifically for the simulation of power electronic switching circuits, and also with Dymola, a state-of-the-art environment for modeling and simulating physical systems.

It turned out that, in spite of their limitations of being first-order algorithms only, both the BQSS and the CQSS are quite competitive when simulating systems with low-to-moderate accuracy requirements. In one example, simulating a switching electronic power converter circuit, the BQSS and CQSS simulations were about five times faster than Matlab and LTSpice, and about 10 times faster than Dymola.

In addition, all of the examples demonstrated that QSS-based solvers do a much better job than classical ODE solvers in terms of controlling the integration error. They do not waste precious computational resources on computing the solution more accurately than requested, but they do not 'cheat' either by computing a solution that does not satisfy the accuracy requirements imposed on the solution.

The BQSS was designed to solve general stiff systems. However, taking into account that QSS methods are particularly efficient for the simulation of systems exhibiting many discontinuities, we found that the BQSS shows important advantages when simulating discontinuous stiff ODEs, such as those present in power electronic circuit models. In those cases, the BQSS significantly outperforms all of the stiff solvers available in Dymola, Matlab, and LTSpice, at least for low-to-moderate accuracy requirements.

In the following list, we summarize the advantages and disadvantages of the BQSS and CQSS methods.

Advantages.

- The BQSS and the CQSS are semi-explicit methods. They do not require Newton iteration. In spite of this, they are suitable for simulating stiff systems.
- The CQSS is also suitable for simulating marginally stable systems.
- The BQSS and the CQSS, like all QSS methods, are highly efficient in handling discontinuities.
- In systems that are both stiff and discontinuous – very typical in power electronics – the BQSS and the CQSS offer very efficient solutions, at least for low-to-moderate accuracy requirements. As shown in the last example, these solvers can improve the speed of the simulations by more than one order of magnitude in comparison with state-of-the-art discrete-time methods.
- The BQSS and the CQSS exhibit strong theoretical properties regarding numerical stability and the global error bounds.

Disadvantages.

- The BQSS and the CQSS are only first-order accurate. They cannot perform simulations with stringent accuracy requirements.
- The implementation of BQSS and CQSS methods is more involved than that of classic algorithms. Firstly, it requires some sort of discrete-event simulation engine. In addition, the state equations need to be evaluated componentwise. Thus, it is not straightforward to include a BQSS/CQSS solver in conventional simulation packages.
- In large non-sparse systems, the BQSS and the CQSS, just like other QSS methods, become inefficient as every step in each state variable results in a large number of individual function evaluations. In this case, discrete-time algorithms are more efficient as they update all state variables together, thereby

performing a much smaller number of full function evaluations.

- In some cases, when integrating non-linear systems, the BQSS may find spurious equilibrium points and terminate the simulation prematurely. Due to this shortcoming, the current implementation of the BQSS may be less robust than a classical stiff system solver that is based on time slicing.

Funding

We wish to acknowledge support by CONICET under grant PIP–2009/2011–00183.

Conflict of interest statement

None declared.

References

1. Zeigler B, Kim TG and Praehofer H. *Theory of modeling and simulation*, 2nd ed. New York: Academic Press, 2000.
2. Bernard P. Zeigler and J. S. Lee, "Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment", In Proceeding of SPIE Vol.3369, pp.49–58, Orlando, Fl., 1998.
3. Kofman E and Junco S. Quantized state systems. A DEVS approach for continuous system simulation. *Trans SCS* 2001; 18: 123–132.
4. Kofman E. A second order approximation for DEVS simulation of continuous systems. *Simulation* 2002; 78: 76–89.
5. Butcher JC. Thirty years of G-stability. *BIT Numer Math* 2006; 46: 479–489.
6. Dahlquist G. Error analysis for a class of methods for stiff nonlinear initial value problems. In: *Proceedings of the Numerical Analysis Conference*, in Lecture Notes Math, Dundee, Scotland, 1975. New York: Springer, 1976, pp.60–74.
7. Gear CW. The automatic integration of ordinary differential equations. *Commun ACM* 1971; 14: 176–179.
8. Cellier FE and Kofman E. *Continuous system simulation*. New York: Springer, 2006.
9. Kofman E. Discrete event simulation of hybrid systems. *SIAM J Sci Comput* 2004; 25: 1771–1797.
10. Nutaro J. Parallel discrete event simulation with application to continuous systems. *PhD thesis*, The University of Arizona, 2003.
11. Kofman E and Braslavsky J. Level Crossing sampling in feedback stabilization under data-rate constraints. In: *Proceedings of CDC'06, IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp.4423–4428.
12. Hairer E and Wanner G. *Solving ordinary differential equations II. Stiff and differential-algebraic problems*. Berlin: Springer, 1991.
13. Hairer E, Lubich C and Wanner G. *Geometric numerical integration structure-preserving algorithms for ordinary differential equations*. Berlin, Germany: Springer, 2002.
14. Kofman E. A third order discrete event simulation method for continuous system simulation. *Lat Am Appl Res* 2006; 36: 101–108.
15. Nutaro J. 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*, 2007, pp.25–31.
16. Nutaro J and Zeigler B. On the stability and performance of discrete event methods for simulating continuous systems. *J Comput Phys* 2007; 227: 797–819.
17. Migoni G. *Simulación por Cuantificación de Sistemas Stiff*. *PhD thesis*, Facultad de Ciencias Exactas, Ingeniera y Agrimensura. Rosario, Argentina: Universidad Nacional de Rosario, Rosario, Argentina, 2010.
18. Lambert JD. *Numerical methods for ordinary differential systems: The initial value problem*. John Wiley & Sons, 1991.
19. Mosbach S and Kraft M. An explicit numerical scheme for homogeneous gas-phase high-temperature combustion systems. *Combust Theor Model* 2006; 10: 171–182.
20. Kofman E. Non conservative ultimate bound estimation in LTI perturbed systems. *Automatica* 2005; 41: 1835–1838.
21. Kofman E, Haimovich H and Seron M. A systematic method to obtain ultimate bounds for perturbed systems. *Int J Control* 2007; 80: 167–178.
22. Bergero F and Kofman E. PowerDEVS. A tool for hybrid system modeling and real time simulation. *Simulation* 2011; 87: 113–132.
23. Jammalamadaka R. Activity characterization of spatial models: application to discrete event solution of partial differential equations. *Master's thesis*, The University of Arizona, 2003.
24. Enright WH and Pryce JD. Two (FORTRAN) packages for assessing initial value methods. *ACM Trans Math Software* 1987; 13(1): 1–27.
25. Khalil H. *Nonlinear systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

Gustavo Migoni received his BS degree in electronic engineering in 2004 and his PhD degree in Automatic Control in 2010, all from the National University of Rosario. He currently holds a Postdoc Fellowship at the French Argentine International Center for Information and Systems Sciences (CIFASIS-CONICET), Rosario, Argentina. He is also a teaching assistant at the National University of Rosario, Faculty of Engineering and Exact Sciences, Department of Control, Rosario, Argentina.

Ernesto Kofman received his BS degree in electronic engineering in 1999 and his PhD degree in Automatic Control in 2003, all from the National University of Rosario. He is an Adjunct Professor at the Department of Control of the National University of Rosario (FCEIA - UNR). He also holds a research position at the National Research Council of

Argentina (CIFASIS-CONICET). His research interests include automatic control theory and numerical simulation of hybrid systems. He coauthored a textbook on continuous system simulation in 2006 with Springer, New York.

François E Cellier received his BS degree in electrical engineering in 1972, his MS degree in automatic control in 1973, and his PhD degree in technical sciences in 1979, all from the Swiss Federal Institute of Technology (ETH) Zurich. He worked at the University of Arizona as professor of Electrical and Computer Engineering from 1984 until 2005. He then returned to his home country of Switzerland, where he is now once again working at his alma mater of ETH Zurich. His main scientific interests concern modeling and simulation methodologies, and the design of advanced software systems for simulation, computer-aided modeling, and computer-aided design. He has authored or co-authored more than 250 technical publications, and he has edited several books. He published a textbook on CONTINUOUS SYSTEM MODELING in 1991 and a second textbook on continuous system simulation in 2006, both with Springer, New York.