Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, and E. Zhou, eds.

IMPROVING A LINEARLY IMPLICIT QUANTIZED STATE SYSTEM METHOD

Franco Di Pietro
Gustavo Migoni
Ernesto Kofman
CIFASIS – CONICET; FCEIA –UNR

27 de Febrero 210 bis
Rosario, 2000, ARGENTINA

ABSTRACT

In this article we propose a modification to the first order Linearly Implicit Quantized State System Method (LIQSS1), an algorithm for continuous system simulation that replaces the classic time discretization by the quantization of the state variables. LIQSS was designed to efficiently simulate stiff systems but it only works when the system has a particular structure. The proposed modification overcomes this limitation allowing the algorithm to efficiently simulate stiff systems with more general structures. Besides describing the new method and its software implementation, the article analyzes the algorithm performance in the simulation of a complex power electronic converter.

1 INTRODUCTION

The simulation of continuous time models requires the numerical integration of the Ordinary Differential Equations (ODEs) that represent them. The literature on numerical methods for ODEs (Hairer, Nørsett, and Wanner 1993, Hairer and Wanner 1991, Cellier and Kofman 2006) contains hundreds of algorithms with different features that make them suitable for solving different type of problems.

Some ODE systems exhibit certain characteristics that pose difficulties to numerical ODE solvers. The presence of simultaneous fast and slow dynamics, known as stiffness, is one of these cases. Due to stability reasons, these systems enforce the usage of implicit ODE solvers that must perform expensive iterations over sets of nonlinear equations at each time step. The presence of discontinuities is another difficult case, where the ODE solvers must detect their occurrence using iterative procedures, restarting the simulation after each event.

ODE models coming from power electronics, spiking neural networks, multi-particle collision dynamics, and several other technical areas, exhibit very frequent discontinuities and, sometimes, stiffness. Consequently, the simulation of these systems becomes very expensive.

In the last years, a new family of numerical ODE solvers that can efficiently handle discontinuities was developed. These algorithms, called Quantized State System (QSS) (Kofman and Junco 2001, Cellier and Kofman 2006), replace the time discretization performed by classic ODE solvers by the quantization of the state variables. Regarding stiffness, a family of Linearly Implicit QSS (LIQSS) solvers was recently developed (Migoni, Bortolotto, Kofman, and Cellier 2013), that can efficiently simulate some of these systems.

A limitation of LIQSS algorithms is that they require that the stiffness is due to the presence of large entries on the main diagonal of the Jacobian matrix of the system. Otherwise, spurious oscillations may appear on the simulated trajectories impoverishing the performance.

In this article, we propose a modification of the first order LIQSS algorithm that overcomes that limitation, extending the cases in which stiffness is efficiently handled.

Besides introducing the new algorithm, we analyze its properties, we describe its implementation in the stand-alone QSS solver (Fernández and Kofman 2014) and we present simulation results, comparing the performance of the new method with that of the original LIQSS1 and the classic DASSL solver.

The paper is organized as follows: Section 2 introduces the previous concepts and definitions used along the rest of the work. Then, Section 3 describes the new algorithm and describes its implementation. Finally, Section 4 presents the simulation results and Section 5 concludes the article.

2 BACKGROUND

This section provides the background required to tackle the rest of the article. Starting with a brief description of the problems suffered by classical ODE solvers when dealing with discontinuous and stiff systems.Then, the family of QSS solvers is presented.

2.1 Numerical Integration of Stiff and Discontinuous ODEs

Many dynamical systems of practical relevance, both in science and engineering, are stiff. Integration of these systems using traditional numerical methods based on time discretization requires the use of implicit algorithms, because all explicit methods must necessarily restrict the integration step to ensure numerical stability. In return, implicit methods have higher computational cost than explicit ones, because they call for iterative algorithms in each step to calculate the next value.

Regarding discontinuities, it must be taken into account that classic algorithms are based, either explicitly or implicitly, on *Taylor series expansions* that express the solution at the next time t_{k+1} as polynomials in the step size h around the current time t_k. As discontinuous trajectories cannot be represented by polynomials, the numerical algorithms usually introduce unacceptable errors when a discontinuity occurs between time t_k and t_{k+1}.

To avoid this problem, ODE solvers must detect the exact instant at which the discontinuity occurs, advance the simulation until that time, and restart the simulation from the new conditions. This strategy, known as zero crossing detection and and event handling, is expensive in terms of computational costs as the zero crossing location usually involves iterations.

2.2 Quantized State System Methods

QSS methods replace the time discretization of classic numerical integration algorithms by the quantization of the state variables.

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

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

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector, the first order Quantized State System (QSS1) method (Kofman and Junco 2001) analytically solves an approximate ODE called Quantized State System:

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

Here, $\mathbf{q}(t)$ is the *quantized state* vector that follows piecewise constant trajectories. Each quantized state $q_i(t)$ is related to the corresponding state $x_i(t)$ by a *hysteresis quantization function*:

$$q_i(t) = \begin{cases} x_i(t) & \text{if } |q_i(t^-) - x_i(t)| = \Delta Q_i \\ q_i(t^-) & \text{otherwise} \end{cases}$$

This is, $q_i(t)$ only changes when it differs from $x_i(t)$ by a magnitude ΔQ_i called *quantum.* After each change in the quantized variable, it results that $q_i(t) = x_i(t)$.

Since the quantized state trajectories $q_i(t)$ are piecewise constant, then, the state derivatives $\dot{x}_i(t)$ also follow piecewise constant trajectories and, consequently, the states $x_i(t)$ follow piecewise linear trajectories.

Due to the particular form of the trajectories, the numerical solution of Eq. (2) is straightforward and can be easily translated into a simple simulation algorithm.

For $j = 1, \dots, n$, let t_j denote the next time at which $\left|q_j(t) - x_j(t)\right| = \Delta Q_j$. Then, the QSS1 simulation algorithm works as follows:

Algorithm 1: QSS1.

```
while($t < t_f$) // simulate until final time tf
  $t$ = min($t_j$) // advance simulation time
  $i$ = argmin($t_j$) // the i-th quantized state changes first
  $e_{xi}$ = $t - t_i^x$ // elapsed time since last xi update
  $x_i$ = $x_i + \dot{x}_i \cdot e_{xi}$ // update i-th state value
  $q_i$ = $x_i$ // update i-th quantized state
  $t_i$ = min($\tau > t$) subject to $|q_i - x_i(\tau)| = \Delta Q_i$ // compute next i-th quantized state change
  for each $j \in [1,n]$ such that $\dot{x}_j$ depends on $q_i$
    $e_{xj}$ = $t - t_j^x$ // elapsed time since last xj update
    $x_j$ = $x_j + \dot{x}_j \cdot e_{xj}$ // update j-th state value
    $t_j^x$ = $t$ // last xj update
    $\dot{x}_j$ = $f_j(\mathbf{q},t)$ // recompute j-th state derivative
    $t_j$ = min($\tau > t$) subject to $\left|q_j - x_j(\tau)\right| = \Delta Q_j$ // recompute j-th quantized state changing
        time
  end for
  $t_i^x$ = $t$ // last xi update
end while
```

QSS1 has very nice stability and error bound properties: the simulation of a stable system provides stable results (Kofman and Junco 2001) and the maximum simulation error (in the simulation of a linear time invariant system) is bounded by a linear function of the quantum size ΔQ.

Since the states follow piecewise linear trajectories, the instant of times at which they cross a given threshold can be computed without iterations, allowing the straightforward detection of discontinuities. Moreover, when a discontinuity occurs, it will eventually change some state derivatives in the same way a change in a quantized variable does during a normal step. That way, the simulation does not need to be restarted. In conclusion, the detection and handling of a discontinuity does not take more computational effort than that of a single step. Thus, the QSS1 method is very efficient to simulate discontinuous systems (Kofman 2004).

In spite of this advantage and the fact that it has some nice stability and error bound properties (Cellier and Kofman 2006), QSS1 performs only a first order approximation and it cannot obtain accurate results without significantly increasing the number of steps. This accuracy limitation was improved with the definition of the second and third-order accurate QSS methods (Cellier and Kofman 2006, Kofman 2006).

2.3 Linearly Implicit QSS Methods

In spite of their advantages, QSS1, QSS2 and QSS3 methods are inefficient to simulate stiff systems. In presence of simultaneous slow and fast dynamics these methods introduce spurious high frequency oscillations that provoke a large number of steps with its consequent computational cost (Cellier and Kofman 2006).

To overcome this problem, the family of QSS methods was extended with a set of algorithms called Linearly Implicit QSS (LIQSS) which are appropriate to simulate some stiff systems (Migoni, Bortolotto, Kofman, and Cellier 2013). LIQSS methods combine the principles of QSS methods with those of classic linearly implicit solvers. There are LIQSS algorithms that perform first, second and third-order accurate approximations: LIQSS1, LIQSS2, and LIQSS3, respectively.

The main idea behind LIQSS methods is inspired in classic implicit methods that evaluate the state derivatives at future instants of time. In classic methods, these evaluations require iterations and/or matrix inversions to solve the resulting implicit equations. However, taking into account that QSS methods know the future value of the quantized state (it is $q_i(t) \pm \Delta Q_i$), the implementation of LIQSS algorithms is explicit and does not require iterations or matrix inversions.

LIQSS methods share with QSS methods the definition of Eq. (2), but the quantized states are computed in a more involved way, taking into account the sign of the state derivatives.

In LIQSS1 the idea is that $q_i(t)$ is set equal to $x_i(t) + \Delta Q_i(t)$ when the future state derivative $\dot{x}_i(t^+)$ is positive. Otherwise, when the future state derivative is negative $q_i(t)$ is set equal to $x_i(t) - \Delta Q_i(t)$. Then, when x_i reaches q_i, a new step is taken. That way, the quantized state is a future value of the state and the derivatives in Eq.(2) are computed using a future state value, as in classic implicit algorithms.

In order to predict the sign of the future state derivative the following linear approximation for the i–th state dynamics is used:

$$\dot{x}_i(t) = A_{i,i} \cdot q_i(t) + u_{i,i}(t) \tag{3}$$

where $A_{i,i} = \frac{\partial f_i}{\partial x_i}$ is the i-th main diagonal entry of the Jacobian matrix and $u_{i,i}(t) = f_i(\mathbf{q}(t),t) - A_{i,i} \cdot q_i(t)$ is an affine coefficient.

It could happen that $A_{i,i} \cdot (x_i(t) + \Delta Q_i) + u_{i,i}(t) < 0$, i.e., when we propose to use $q_i(t) = x_i(t) + \Delta Q_i$ the derivative $\dot{x}_i(t^+)$ becomes negative. It can also happen that $A_{i,i} \cdot (x_i(t) - \Delta Q_i) + u_{i,i}(t) > 0$. Thus, $q_i(t)$ cannot be chosen as a future value for $x_i(t)$. However, in that case, q_i can be chosen such that $\dot{x}_i(t) = 0$. That equilibrium value for q_i can be calculated from Eq.(3) as

$$q_i = -\frac{u_{i,i}}{A_{i,i}} \tag{4}$$

Then, the LIQSS1 simulation algorithm works as follows:

Algorithm 2: LIQSS1.

```
while($t < t_f$) // simulate until final time tf
  $t$ = min($t_j$) // advance simulation time
  $i$ = argmin($t_j$) // the i-th quantized state changes first
  $e_{xi}$ = $t - t_i^x$ // elapsed time since last xi update
  $x_i$ = $x_i + \dot{x}_i \cdot e_{xi}$ // update i-th state value
  $q_i^-$ = $q_i$ //store previous value of qi
  $\dot{x}_i^-$ = $\dot{x}_i$ //store previous value of dxi/dt
  $\dot{x}_i^+$ = $A_{i,i} \cdot (x_i + sign(\dot{x}_i) \cdot \Delta Q_i) + u_{i,i}$ // future state derivative estimation
  if ($\dot{x}_i \cdot \dot{x}_i^+ > 0$) //the state derivative keeps its sign
    $q_i$ = $x_i + sign(\dot{x}_i) \cdot \Delta Q_i$
  else //the state changes its direction
    $q_i$ = $-u_{i,i}/A_{i,i}$ // choose qi such that dxi/dt = 0
  end if
  $t_i$ = min($\tau > t$) subject to $x_i(\tau) = q_i$ // compute next i-th quantized state change
  for each $j \in [1,n]$ such that $\dot{x}_j$ depends on $q_i$
    $e_{xj}$ = $t - t_j^x$ // elapsed time since last xj update
    $x_j$ = $x_j + \dot{x}_j \cdot e_{xj}$ // update j-th state value
    $t_j^x$ = $t$// last xj update
    $\dot{x}_j$ = $f_j(\mathbf{q},t)$ // recompute j-th state derivative
    $t_j$ = min($\tau > t$) subject to $x_j(\tau) = q_j$ or $|q_j - x_j(\tau)| = 2\Delta Q_j$ // recompute next j-th quantized
        state change
  end for
  // update linear approximation coefficients
  $A_{i,i}$ = ($\dot{x}_i - \dot{x}_i^-$)/($q_i - q_i^-$) // Jacobian diagonal entry
  $u_{i,i}$ = $\dot{x}_i - A_{i,i} \cdot q_i$ // affine coefficient
  $t_i^x$ = $t$ // last xi update
end while
```

It can be seen that LIQSS1 steps only add a few calculations to those of QSS1. In particular, LIQSS1 estimates the future state derivative using a linear model (line 8) and it estimates the Jacobian main diagonal entry $A_{i,i}$ and the affine coefficient (lines 23–24).

Notice also that in line 20 the algorithm checks the additional condition $\left|q_j - x_j(\tau)\right| = 2\Delta Q_j$, as a change in variable q_i can change the sign of the state derivative $\dot{x}_j(t)$ so that x_j does no longer approach q_j. In this case, we still ensure that the difference between x_j and q_j is bounded (by $2\Delta Q_j$). However, we shall see then that the fact that x_j does not always approach q_j may result into non efficient simulation of some stiff systems.

LIQSS1 shares the main advantages of QSS1 and it can efficiently integrate stiff systems provided that the stiffness is due to the presence of large entries in the main diagonal of the Jacobian matrix. Like QSS1, it cannot achieve good accuracy and higher order LIQSS methods were proposed.

The second and third order accurate LIQSS2 and LIQSS3 combine the ideas of QSS2 and QSS3 with the principles of LIQSS1.

2.4 Implementation of QSS Methods

The easiest way of implementing QSS methods is by building an equivalent DEVS model, where the events represent changes in the quantized variables. Based on this idea, the whole family of QSS methods were implemented in PowerDEVS (Bergero and Kofman 2011), a DEVS–based simulation platform specially designed for and adapted to simulating hybrid systems based on QSS methods. In addition, the explicit QSS methods of orders 1 to 3 were also implemented in a DEVS library of Modelica (Beltrame and Cellier 2006) and implementations of the first–order QSS1 method can also be found in CD++ (D'Abreu and Wainer 2005) and VLE (Quesnel, Duboz, Ramat, and Traoré 2007).

Recently, the complete family of QSS methods was implemented in a *stand–alone QSS solver* (Fernández and Kofman 2014) that improves DEVS–based simulation times in more than one order the magnitude.

The stand–alone QSS solver requires that the models are described in a subset of the Modelica modeling language (Tiller 2012), called μ-Modelica (Fernández and Kofman 2014).

3 MODIFIED LIQSS1 ALGORITHM

In this section, we first analyze the main limitation of LIQSS1 concerning the appearance of fast oscillations in systems where the stiffness is not due to large entries on the main diagonal of the Jacobian matrix. Then, we propose an idea to overcome this problem, and, using this approach we propose a first order accurate modified LIQSS method.

3.1 LIQSS1 limitations

The simulation of a stable first order system with QSS1 algorithm produces a result that usually finishes with the state trajectory oscillating around the equilibrium point (Cellier and Kofman 2006). These oscillations are the reason why QSS1 is not efficient to simulate stiff systems.

That problem is solved by LIQSS1, that prevents the oscillations by taking the quantized state as a future value of the state. When it is not possible, LIQSS1 finds the equilibrium point using a linear approximation.

However, LIQSS1 cannot ensure that q_i is always the future value of x_i because, after computing q_i, $\dot{x}_i$ can change its sign due to a change in some other quantized variable q_j. In such case, then it can also happen that the next change in q_i triggers a change in the sign of $\dot{x}_j$. This situation may lead to oscillations involving states x_i and x_j.

3.2 Basic Idea

In order to avoid oscillations between pairs of variables, we propose to check whether a quantized state update changes the sign of some other state derivative. If so, we additionally check whether an eventual update of the second quantized state would change back the sign of the first state derivative. Under this situation, we expect that both variables experience spurious oscillations, and, in order to prevent them, we apply a simultaneous change in both quantized states using a linearly implicit Backward Euler step.

While this strategy may not solve general stiff structures, it will avoid the appearance of oscillations between pairs of variables, what covers several practical cases.

3.3 Modified LIQSS1

Based on the idea expressed above, the modification introduced to the LIQSS1 algorithm consists in checking an additional condition to verify that after changing a quantized state, the other state derivatives would not change their sign. To check this condition, for each pair of state variables x_i, x_j, such that both influence each other state derivatives, a second order linear approximation model of the form

$$\begin{aligned} \dot{x}_i &= A_{ii} \cdot q_i + A_{ij} \cdot q_j + u_{ij} \\ \dot{x}_j &= A_{ji} \cdot q_j + A_{jj} \cdot q_j + u_{ji} \end{aligned} \tag{5}$$

is used. Here, $A_{i,j} = \frac{\partial f_i}{\partial x_j}(\mathbf{q},t)$ is the i,j entry of the Jacobian matrix, and $u_{ij} = f_i(\mathbf{q},t) - A_{ii} \cdot q_i - A_{ij} \cdot q_j$ is an affine coefficient.

If the new value of q_i does not introduce any change in the sign of the other state derivatives computed with the linear approximation of Eq.(5), the algorithm works identically to LIQSS1. However, when the new value of q_i provokes that the sign of $\dot{x}_j$ changes in Eq.(5), we propose a new value for q_j in the new direction of x_j. Then, we check if that proposed value for q_j changes the sign of $\dot{x}_i$. If it does not, we forget about the change in q_j and the algorithm follows identical steps to those of LIQSS1. Otherwise, we know that an oscillation may appear between states x_i and x_j, so we compute both quantized states q_i and q_j simultaneously using a Backward Euler step on the model of Eq.(5).

Defining

$$\mathbf{q}_{ij} \triangleq \begin{bmatrix} q_i \\ q_j \end{bmatrix} \; \mathbf{x}_{ij} \triangleq \begin{bmatrix} x_i \\ x_j \end{bmatrix} \; \dot{\mathbf{x}}_{ij} \triangleq \begin{bmatrix} \dot{x}_i \\ \dot{x}_j \end{bmatrix} \; \mathbf{A}_{ij} = \begin{bmatrix} A_{ii} \, A_{ij} \\ A_{ji} \, A_{jj} \end{bmatrix} \; \mathbf{u}_{ij} \triangleq \begin{bmatrix} u_{ij} \\ u_{ji} \end{bmatrix} \tag{6}$$

the backward step is given by the equation

$$\mathbf{q}_{ij}(t) = \mathbf{x}_{ij}(t) + h \cdot \dot{\mathbf{x}}_{ij}(t+h) = \mathbf{x}_{ij}(t) + h \cdot (\mathbf{A}_{ij} \cdot \mathbf{q}_{ij}(t) + \mathbf{u}_{ij}) \tag{7}$$

where h is computed as the maximum step size so that the difference between the states and the quantized states is bounded by the quantum. When the states x_i and x_j are near an equilibrium point for Eq.(5), the maximum step size h is infinite and the resulting quantized states are those corresponding to the equilibrium.

Taking into account these considerations, the modified LIQSS1 simulation algorithm is identical to that of LIQSS1 (Algorithm 2) until line 14. Afterwards it continues as follows:

Algorithm 3: Modified LIQSS1.

```
15  for each $j \in [1,n]$ such that (i≠j and $A_{ij} \cdot A_{ji} \neq 0$)
16    $e_{xj}$ = $t - t_j^x$ // elapsed time since last xj update
17    $x_j$ = $x_j + \dot{x}_j \cdot e_{xj}$ // update j-th state value
18    $u_{ji}$ = $u_{jj} - A_{ji} \cdot q_i^-$ // affine coefficient
19    $\dot{x}_j^+$ = $A_{ji} \cdot q_i + A_{jj} \cdot q_j + u_{ji}$ // next j-th state der. est.
20    if($\dot{x}_j \cdot \dot{x}_j^+ < 0$) // update in qi => change of sign in dxj/dt
21      $q_j^+$ = $x_j + sign(\dot{x}_j^+) \cdot \Delta Q_j$ // update qj in future xj's direction
22      $u_{ij}$ = $u_{ii} - A_{ij} \cdot q_j$ // affine coefficient
```

```
23        $\dot{x}_i^{++} = A_{ii} \cdot q_i + A_{ij} \cdot q_j^{+} + u_{ij}$ // next i-th state der. est.
24        if ($\dot{x}_i^{+} \cdot \dot{x}_i^{++} < 0$) // update in qj => change of sign in dxi/dt
25          // presence of oscillations
26          $q_j^{-} = q_j$ //store previous value of qj
27          $\dot{x}_j^{-} = \dot{x}_j$ //store previous value of dxj/dt
28          $h$ = MAX_BE_STEP_SIZE($x_i$,$x_j$) // maximum BE step size such that ( $|x_i(k+1) - x_i(k)| \le \Delta Q_i \wedge |x_j(k+1) - x_j(k)| \le \Delta Q_j$)
29          $[q_i, q_j]$ = BE_step($x_i$,$x_j$,$h$) // qi and qj are computed using a BE step size h from xi and xj
30          $t_j^q = t$ // last qj update
31          $t_j$ = min($\tau > t$) subject to $x_j(\tau) = q_j$ or $|q_j - x_j(\tau)| = 2\Delta Q_j$ // compute next j-th quantized state
32          for each $k \in [1,n]$ such that $\dot{x}_k$ depends on $q_j$
33            $e_{xk} = t - t_k^x$ // elapsed time since last xk update
34            $x_k = x_k + \dot{x}_k \cdot e_{xk}$ // update k-th state value
35            $\dot{x}_k^{-} = \dot{x}_k$ //store previous value of dxk/dt
36            $\dot{x}_k = f_k(\mathbf{q},t)$ // recompute k-th state derivative
37            $t_k$ = min($\tau > t$) subject to $x_k(\tau) = q_k$ or $|q_k - x_k(\tau)| = 2\Delta Q_k$ // compute next k-th quantized state
38            $A_{k,j}$ = ($\dot{x}_k - \dot{x}_k^{-}$) / ($q_j - q_j^{-}$) // Jacobian
39            $t_k^x = t$ // last xk update
40          end for
41          $A_{j,j}$ = ($\dot{x}_j - \dot{x}_j^{-}$) / ($q_j - q_j^{-}$) // Jacobian diagonal entry
42          $u_{j,j} = \dot{x}_j(t) - A_{j,j} \cdot q_j$ // affine coefficient
43        end if
44      end if
45    end for
46    for each $j \in [1,n]$ such that $\dot{x}_j$ depends on $q_i$
47      $e_{xj} = t - t_j^x$ // elapsed time since last xj update
48      $x_j = x_j + \dot{x}_j \cdot e_{xj}$ // update j-th state value
49      $\dot{x}_j^{-} = \dot{x}_j$ //store previous value of dxj/dt
50      $\dot{x}_j = f_j(\mathbf{q},t)$ // recompute j-th state derivative
51      $t_j$ = min($\tau > t$) subject to $x_j(\tau) = q_j$ or $|q_j - x_j(\tau)| = 2\Delta Q_j$ // compute next j-th quantized state
52      $A_{j,i}$ = ($\dot{x}_j - \dot{x}_j^{-}$) / ($q_i - q_i^{-}$) // Jacobian
53      $t_j^x = t$ // last xj update
54    end for
55    // update linear approximation coefficients
56    $A_{i,i}$ = ($\dot{x}_i - \dot{x}_i^{-}$) / ($q_i - q_i^{-}$) // Jacobian diagonal entry
57    $u_{i,i} = \dot{x}_i(t) - A_{i,i} \cdot q_i$ // affine coefficient
58    $t_i^x = t$ // last xi update
59  end while
```

Notice that this new algorithm adds the calculation of a simultaneous step on states x_i and x_j (lines 28–29), but this only takes place under the occurrence of oscillations. In other case, the algorithm only has some additional calculations to detect changes in the sign of the state derivatives, what requires estimating them (lines 19 and 23) and estimating also the complete Jacobian matrix (lines 38, 41, 52 and 56) and different affine coefficients.

3.4 Implementation of Modified LIQSS methods

The modified algorithm was implemented in the Stand Alone QSS Solver. For that purpose, the pseudo code of Algorithm 3 was programmed as plain C functions of the QSS solver. The corresponding codes are available at `https://sourceforge.net/projects/qssengine/`

4 EXAMPLES AND RESULTS

This section shows the simulation results, comparing the performance of the original LIQSS1 method with the modified algorithm mLIQSS1 and with the classic DASSL solver in the simulation of power converter.

In order to perform this comparison, we run a set of experiments according to the conditions described below:

- We simulated all systems under two different error tolerance settings: rel.tol. = abs.tol $= 10^{-1}$ and rel.tol. = abs.tol $= 10^{-2}$.
- The simulations were performed on an AMD A4-3300 APU@2.5GHz PC under Ubuntu OS.
- In all cases, we measured the CPU time, the number of scalar function evaluations and the relative error, computed as:

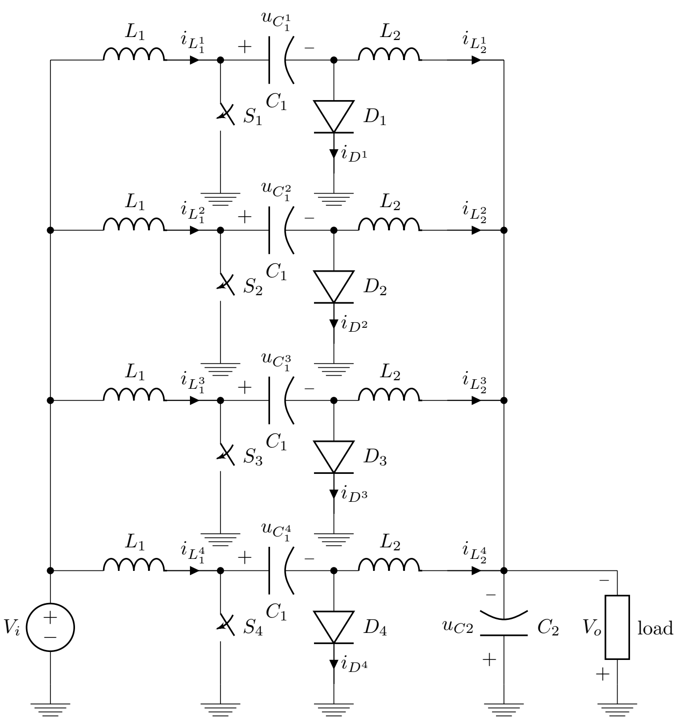
$$e_{rr} = \sqrt{\frac{\sum(u_C[k] - u_{C_{REF}}[k])^2}{\sum u_{C_{REF}}[k]^2}} \tag{8}$$

  where the reference solution $u_{C_{REF}}[k]$ was obtained using DASSL with a very small error tolerance (10^{-9}).

Interleaved Ćuk Converter

As we discussed, the LIQSS algorithms cannot efficiently simulate systems where the stiffness is not due to the presence of large entries on the main diagonal of the Jacobian matrix. An example where LIQSS fails is a power electronic device called *Ćuk* converter, as it is reported in (Migoni, Bergero, Kofman, and Fernández 2015).

Here, we analyze the performance of the modified LIQSS1 in the simulation of this system, considering a four stage–*interleaved* version of the circuit as depicted in Fig. 1.



$$\frac{d\,i_{L_1^j}}{dt} = \frac{U - u_{C_1^j} - i_{D^j}\cdot R_{D^j}}{L_1}$$

$$\frac{d\,i_{L_2^j}}{dt} = \frac{-u_{C_2} - i_{D^j}\cdot R_{D^j}}{L_2}$$

$$\frac{d\,u_{C_1^j}}{dt} = \frac{i_{D^j} - i_{L_2^j}}{C_1}$$

$$\frac{d\,u_{C_2}}{dt} = \frac{\sum_{j=1}^{N} i_{L_2^j} - \frac{u_{C_2}}{R_o}}{C_2}$$

with

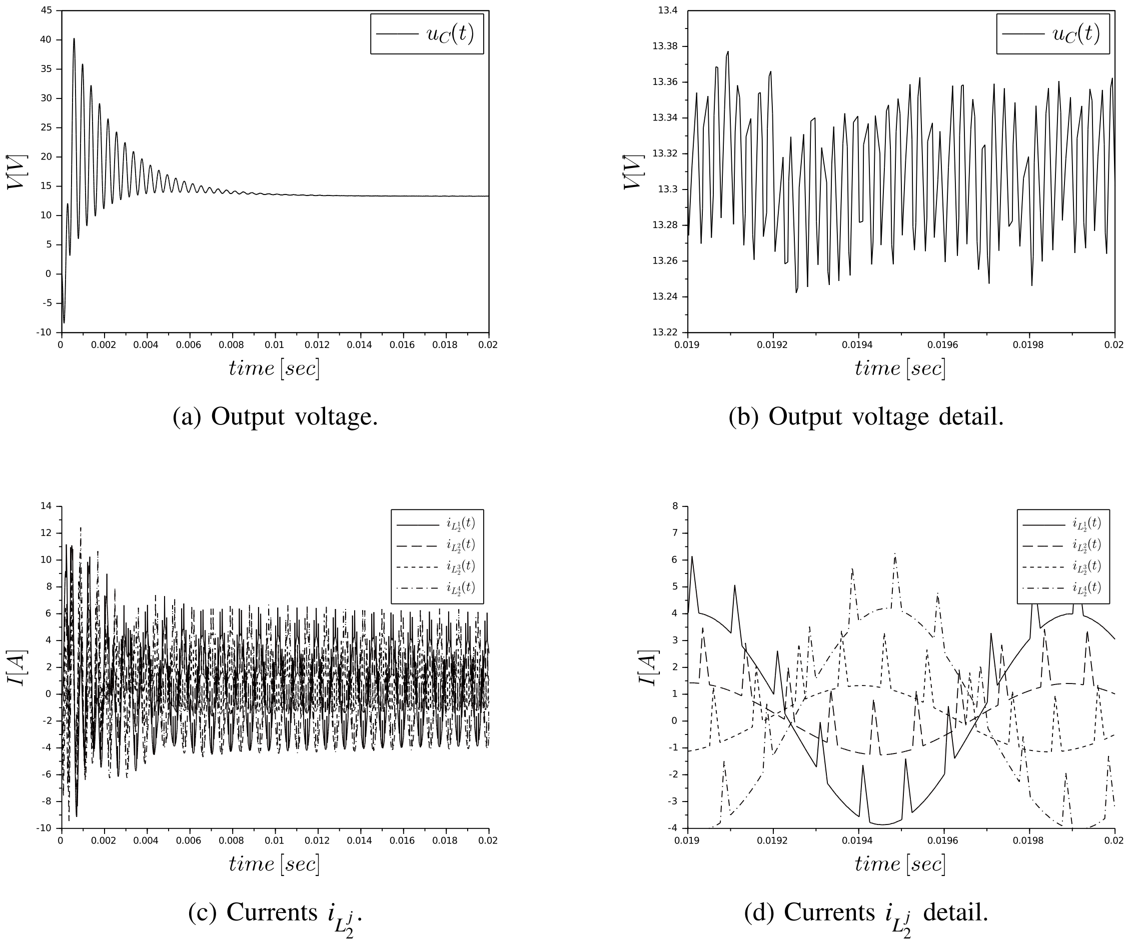
$$i_{D^j} = \frac{(i_{L_1^j} + i_{L_2^j})\cdot R_{S^j} - u_{C_1^j}}{R_{S^j} + R_{D^j}}$$

Figure 1: Four-stage Ćuk interleaved converter circuit and its ODE representation.

This converter was simulated with the following set of parameters:

- Input source voltage: $U = 24V$
- Capacities: $C_1 = 10^{-4}F$ and $C_2 = 10^{-4}F$
- Inductances $L_1 = 10^{-4}Hy$ and $L_2 = 10^{-4}Hy$
- Load resistance: $R_o = 10\Omega$
- Switch and diode On-state resistance: $R_{ON} = 10^{-5}\Omega$
- Switch and diode On-state resistance: $R_{OFF} = 10^{5}\Omega$
- Switch control signal period: $T = 10^{-4}sec$
- Switch control signal duty cycle: $DC = 0.35$

We simulated this system with the different algorithms (DASSL, LIQSS1 and modified LIQSS1) until final time of $t_f = 0.02sec$, where the trajectories reach a permanent regime, as it can be seen in the results of Fig. 2.



(a) Output voltage.

(b) Output voltage detail.

(c) Currents $i_{L_2^j}$.

(d) Currents $i_{L_2^j}$ detail.

Figure 2: Four-stage Ćuk converter simulation results.

The performance comparison for the original LIQSSS method, the modified one and the classic DASSL solver is reported on Table 1.

Integration Method		**Relative Error**	**Jacobian Eval.**	**Function f_i Evaluations**	**CPU [mseg]**
LIQSS1	$\Delta Q_i = 1 \cdot 10^{-1}$	$3.0 \cdot 10^{-}2$	–	80243117	9697.97
	$\Delta Q_i = 1 \cdot 10^{-2}$	$2.8 \cdot 10^{-}3$	–	102413128	12399.1
DASSL	$\Delta Q_i = 1 \cdot 10^{-1}$	$1.7 \cdot 10^{-}1$	26113	4928846	227.102
	$\Delta Q_i = 1 \cdot 10^{-2}$	$7.5 \cdot 10^{-}2$	22556	4402996	218.124
mLIQSS1	$\Delta Q_i = 1 \cdot 10^{-1}$	$4.4 \cdot 10^{-}2$	–	1043458	163.234
	$\Delta Q_i = 1 \cdot 10^{-2}$	$5.2 \cdot 10^{-}3$	–	5123830	676.012

Table 1: 4-Stage Interleaved Ćuk converter results comparison.

The first observation is that the original LIQSS1 algorithm performs a huge amount of steps that can be explained by the appearance of spurious oscillations between the state variables that compute the inductance currents, as its was already observed in (Migoni, Bergero, Kofman, and Fernández 2015). It is clear that the modified LIQSS1 does not suffer from this issue and it performs between 20 and 80 times less steps.

For low accuracy settings (tolerance = 10^{-1}), the modified LIQSS1 outperforms DASSL. However, when the tolerance is set equal to 10^{-2}, DASSL is faster. This can be easily explained by the fact that LIQSS1 is only first order accurate and it cannot achieve a good accuracy without increasing significantly the number of steps. Anyway, it can be noticed that, for identical tolerance settings, mLIQSS1 is much more accurate than DASSL.

The advantages of mLIQSS1 with respect to DASSL are due to the efficient discontinuity handling and sparsity exploitation, as well as the explicit treatment of stiffness (it only inverts 2 by 2 matrices irrespective of the system size n, while DASSL needs to invert an n by n matrix at each step).

These advantages should be more notorious as the size of the system grows. In order to verify this fact, we also simulated the model varying the size from 4 to 32 stages. In each of these experiments, we set the tolerance of each solver so that the measured error results the same. That way, we compare the CPU time required by each solver to simulate the system obtaining identical errors.

The results of these measurements are shown in Fig.3. As expected, the advantages of mLIQSS1 grow with the number of stages, resulting about 10 times faster than DASSL for the case with 32 stages.

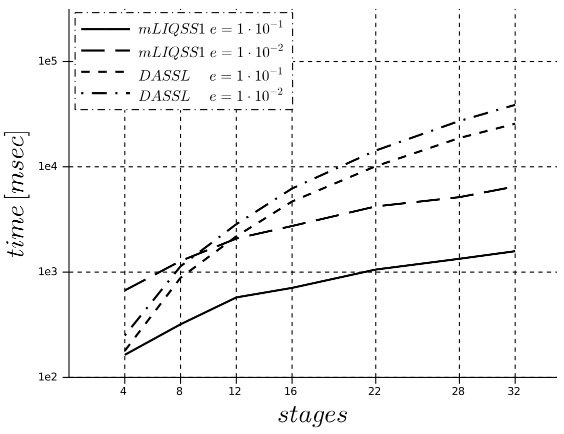


Figure 3: Four-stage Ćuk interleaved converter simulation time comparison.

5 CONCLUSIONS

A modification of the first order accurate Linearly Implicit Quantized State System Method was proposed, allowing to efficiently simulate stiff systems with more general structures than those having large entries restricted to the main diagonal of the Jacobian matrix.

The mLIQSS1 algorithm was implemented in the stand alone QSS solver and tested in the simulation of a complex power electronic converter comparing its performance with that of the original LIQSS1 method and the classic DASSL solver. The performance analysis showed that the mLIQSS1 overcomes the appearance of spurious oscillations exhibited by LIQSS1 and, for low accuracy settings, it was significantly faster than DASSL, particularly when the size of the system grows.

Besides the advantages demonstrated in the case study, the proposed method has a remarkable feature of mixing LIQSS1 and Backward Euler's algorithms. When it predicts that LIQSS1 may lead to oscillations on certain sub-model, it applies a Backward Euler step on the corresponding state variables. That way, mLIQSS1 constitutes the first approach to effectively combine Quantized State and classic discrete time ODE solvers.

Regarding potential applications, QSS algorithms are particularly efficient to simulate models with frequent discontinuities (like Power Electronic Systems) as well as large sparse models. Thus, we expect that mLIQSS ideas lead to the efficient simulation of Power Electronic circuits where LIQSS fail: the already mentioned Ćuk converter, the different Z source topologies (that have a similar structure to that of the Ćuk), as well as more general switching converters under the presence of parasitic inductances and capacitances.

The main limitation of mLIQSS1 is that it is only first order accurate. Thus, we are currently working on developing higher order versions.

Besides extending mLIQSS1 to higher order, future research should study the performance of this approach in a wider variety of applications.

FUNDING

This work was partially funded by FONCYT-ANPCYT (grant PICT 2012 number 0077) and CONICET (grant PIP 2012 number 0216).

REFERENCES

Beltrame, T., and F. E. Cellier. 2006. "Quantised state system simulation in Dymola/Modelica using the DEVS formalism". In *Proceedings 5th International Modelica Conference*, 73–82.

Bergero, F., and E. Kofman. 2011. "PowerDEVS. A Tool for Hybrid System Modeling and Real Time Simulation". *Simulation: Transactions of the Society for Modeling and Simulation International* 87 (1–2): 113–132.

Cellier, F., and E. Kofman. 2006. *Continuous System Simulation*. New York: Springer.

D'Abreu, M. C., and G. A. Wainer. 2005. "M/CD++: modeling continuous systems using Modelica and DEVS". In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*, 229–236. IEEE.

Fernández, J., and E. Kofman. 2014. "A Stand-Alone Quantized State System Solver for Continuous System Simulation.". *Simulation: Transactions of the Society for Modeling and Simulation International* 90 (7): 782–799.

Hairer, E., S. Nørsett, and G. Wanner. 1993. *Solving Ordinary Dfferential Equations I. Nonstiff Problems*. 2nd ed. Berlin: Springer.

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

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

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

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

Migoni, G., F. Bergero, E. Kofman, and J. Fernández. 2015. “Quantization-Based Simulation of Switched Mode Power Supplies.”. *Simulation: Transactions of the Society for Modeling and Simulation International* 91 (4): 320–336.

Migoni, G., M. Bortolotto, E. Kofman, and F. Cellier. 2013. “Linearly Implicit Quantization-Based Integration Methods for Stiff Ordinary Differential Equations”. *Simulation Modelling Practice and Theory* 35:118–136.

Quesnel, G., R. Duboz, É. Ramat, and M. K. Traoré. 2007. “VLE: a multimodeling and simulation environment”. In *Proceedings of the 2007 summer computer simulation conference*, 367–374. Society for Computer Simulation International.

Tiller, M. 2012. *Introduction to physical modeling with Modelica*, Volume 615. Springer Science & Business Media.