Quantization-based Integration of Ordinary Differential Equation Systems.

Prof. Dr. François E. Cellier

Institute of Computational Science
ETH Zurich.

04/18/2007

Outline

Outline

Numerical Integration of ODEs

Problem Formulation

We look for numerical solutions of an initial value problem given in its state-space representation:

$$\begin{aligned} \dot{x}(t) &= f(x(t), t) \\ x(t_0) &= x_0. \end{aligned} \qquad (1)$$

Here, $x \in \Re^n$ is the state vector, and x_0 is the known initial condition.

Numerical Integration of ODEs
Usual Solutions

Conventional numerical methods lead to solutions of the form:

$$x(t_{k+1}) = x_{k+1} = F(x_k, t_k) \tag{2}$$

or more generally

$$F(x_{k+1}, x_k, t_k) = 0 \tag{3}$$

or similar expressions.

These kind of solutions explicitely or implicitely define a discrete time simulation model, where the numerical solution x_k is only defined for $t = t_1, t_2, \cdots, t_k, \cdots$.

Numerical Integration of ODEs
Some Important Concepts and Problems

In order to rely on the solutions given by a method, it is important to analyze:

- Numerical stability
- Approximation accuracy

The following special cases must be treated carefully:

- Discontinuous systems
- Stiff systems
- Marginally stable systems

Quantization–based Integration

Introductory Example

Consider the second order LTI system

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) \end{aligned} \tag{4}$$

with initial conditions $x_1(0) = 4.5,\ x_2(0) = 0.5$.

Let us see what happens if instead of discretizing the time, we discretize the states in the following way:

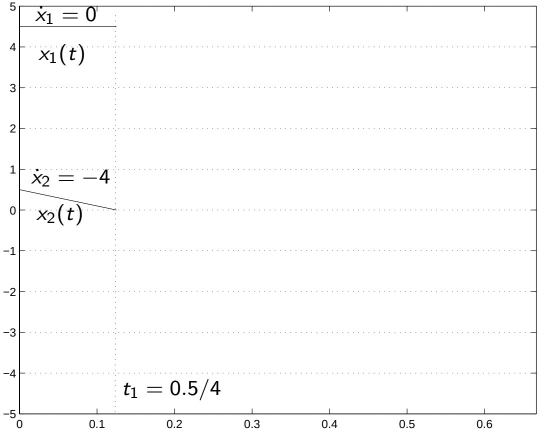
$$\begin{aligned} \dot{x}_1(t) &= \text{floor}(x_2(t)) = q_2(t) \\ \dot{x}_2(t) &= -\text{floor}(x_1(t)) = -q_1(t) \end{aligned} \tag{5}$$

Quantization–based Integration

Introductory Example – Cont.

We can easily solve the quantized system

$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -q_1(t)$$

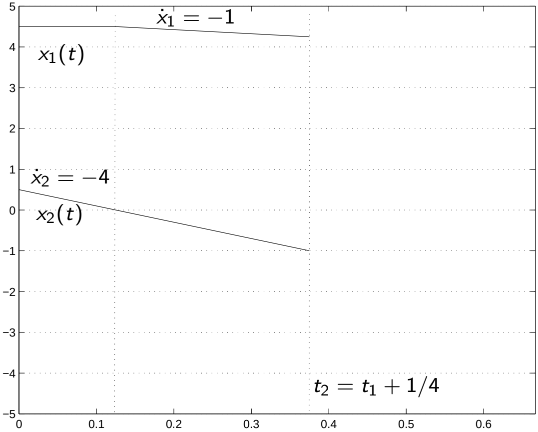


Quantization–based Integration

Introductory Example – Cont.

We can easily solve the quantized system

$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -q_1(t)$$

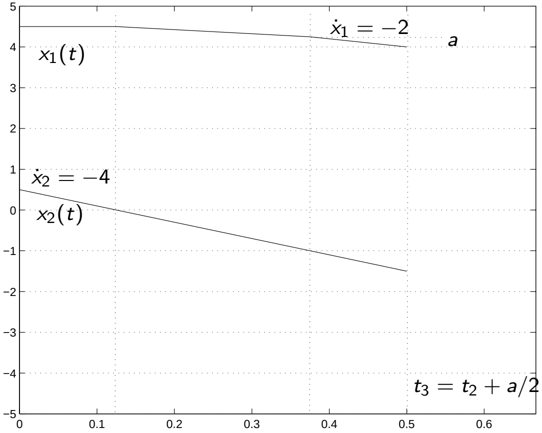


Quantization–based Integration

Introductory Example – Cont.

We can easily solve the quantized system

$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -q_1(t)$$

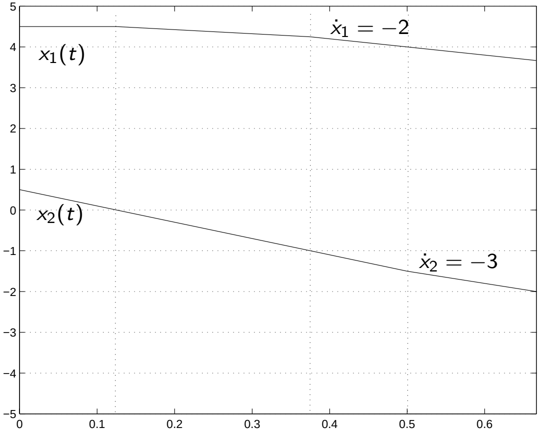


Quantization–based Integration

Introductory Example – Cont.

We can easily solve the quantized system

$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -q_1(t)$$

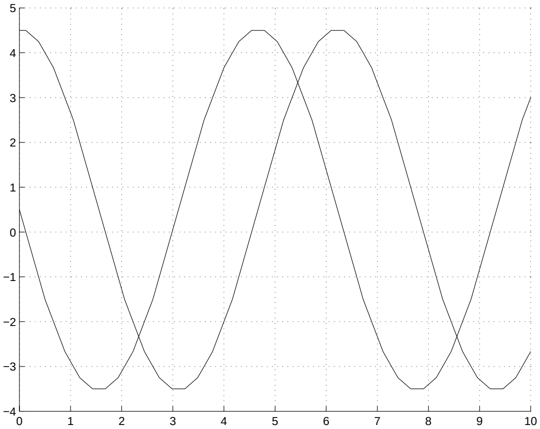


Quantization–based Integration

Introductory Example – Cont.

We can easily solve the quantized system

$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -q_1(t)$$



Quantization–based Integration

Discrete Events vs. Discrete Time

- Apparently, by replacing x_k by q_k on the right hand side of any ODE, we obtain a new method for simulating it.
- However, this new method does not fit the form of Eq.(2) or Eq.(3), i.e., it does not define a Discrete Time simulation model.
- Thus, we will not be able to apply this method in a standard way.

We shall see that this idea leads to a Discrete Event simulation model in terms of the DEVS formalism.

DEVS Formalism
Basic Notions

DEVS is a formalism proposed by Bernard Zeigler to represent systems that have input and output event trajectories.

A DEVS model processes an input trajectory, specified as a series of input events, and according to these events and to its own internal state, provokes an output trajectory.

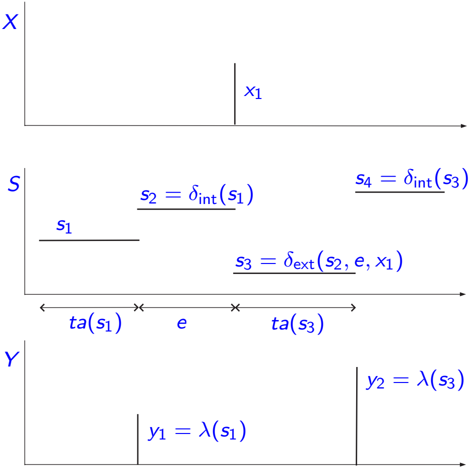
DEVS models can be coupled similarly to block diagrams.

DEVS Formalism

Atomic Model Definition

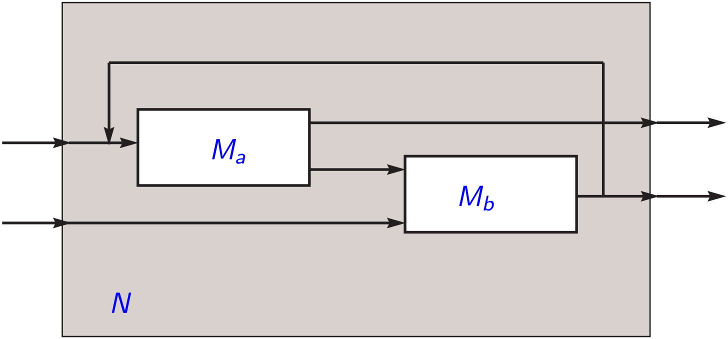
$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

- X : set of input values.
- Y : set of output values.
- S : set of state values.
- δ_{int} : internal trans. func.
- δ_{ext} : external trans. func.
- λ : output func.
- ta : time advance func.



DEVS Formalism
Coupling

DEVS models can be coupled in a hierarchical way. DEVS closure under coupling ensures that coupled DEVS models are equivalent to atomic DEVS models.



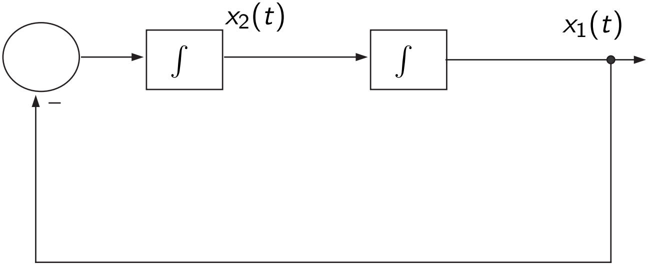
DEVS Formalism

Simulation

An ad–hoc computer program to simulate a DEVS model can be easily written using any programming language. However, there exist a number of software tools specifically conceived to simulate DEVS models:

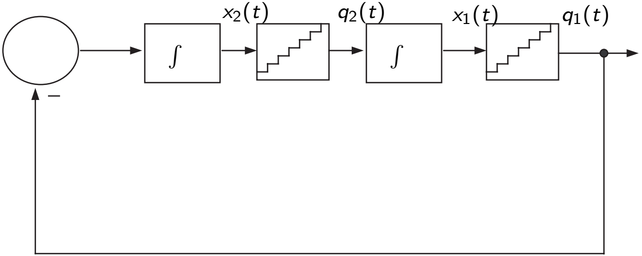
- DEVS–Java (University of Arizona)
- CD++ (Carleton University)
- JDEVS (Université de Corse)
- PowerDEVS (Universidad Nacional de Rosario)
- PyDEVS (McGill University)
- etc.

Quantized Systems and DEVS
Block Diagram of the Original System



$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = -x_1(t)$$

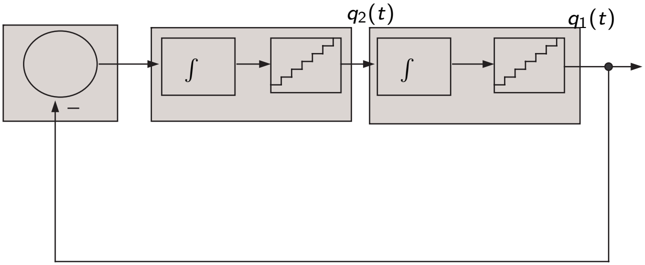
Quantized Systems and DEVS
Block Diagram of the Quantized System



$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -q_1(t)$$

Quantized Systems and DEVS

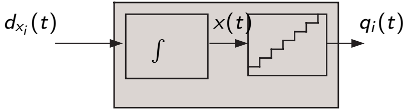
Block Diagram of the Quantized System



$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -q_1(t)$$

Quantized Systems and DEVS

Quantized Integrator

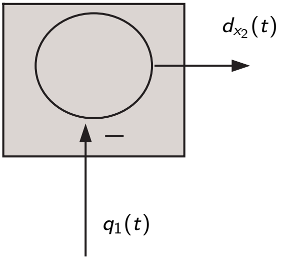


- Each change in the piecewise constant input trajectory $d_{x_i}(t)$ can be thought of as an input event.
- Each change in the piecewise constant output trajectory $q_i(t)$ can be thought of as an output event.
- The piecewise linear state $x_i(t)$ can be treated as part of the internal DEVS state, and is updated at event times.

We can easily build a DEVS atomic model that emulates the behavior of the Quantized Integrator.

Quantized Systems and DEVS

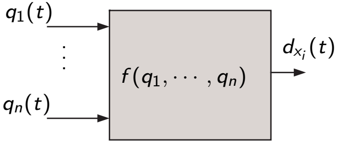
Static Function



- Each change in the piecewise constant input trajectory $q_1(t)$ can be thought of as an input event.
- Each change in the piecewise constant output trajectory $d_{x_2}(t)$ can be thought of as an output event.

We can easily build a DEVS atomic model that emulates the behavior of this particular Static Function.

Quantized Systems and DEVS
Static Function



- Each change in any of the piecewise constant input trajectories $q_j(t)$ can be thought of as an input event.
- Each change in the piecewise constant output trajectory $d_{x_i}(t)$ can be thought of as an output event.

We can easily build a DEVS atomic model that emulates the behavior of a general Static Function.

Quantized Systems
General Idea

Given a continuous system

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

the quantized system

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

is equivalent to a DEVS model and, at least in principle, can be simulated by coupling quantized integrators, static functions, and source blocks.

Quantized Systems
A Little Drawback

Let us analyze what happens with the following first order system

$$\dot{x}(t) = -x(t) - 0.5$$

and its associated Quantized System:

$$\dot{x}(t) = -floor(x(t)) - 0.5$$

around the initial condition $x(0) = 0$.

Evidently, this idea does not work. The DEVS model is illegitimate, and the simulation will get stuck performing an infinite number of steps without advancing time.

Quantized Systems
A Little Drawback

Let us analyze what happens with the following first order system

$$\dot{x}(t) = -x(t) - 0.5$$

and its associated Quantized System:

$$\dot{x}(t) = -floor(x(t)) - 0.5$$

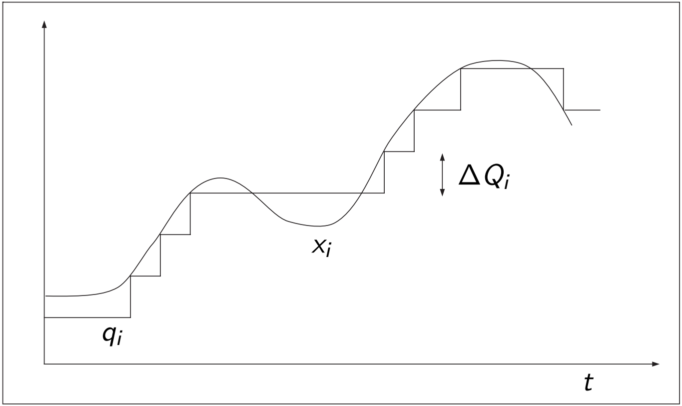
around the initial condition $x(0) = 0$.

Evidently, this idea does not work. The DEVS model is illegitimate, and the simulation will get stuck performing an infinite number of steps without advancing time.

Outline

Quantized State Systems Method

Hysteretic Quantization Function



The basic idea to avoid infinitely fast oscillations is the use of hysteresis in the quantization.

QSS Method
Definition

Given an ODE in its state-space representation

$$\dot{x}_a(t) = f(x_a(t), u(t)) \tag{6}$$

with $x_a \in \Re^n$, $u \in \Re^m$ and $f : \Re^n \rightarrow \Re^n$, the QSS method approximates it by

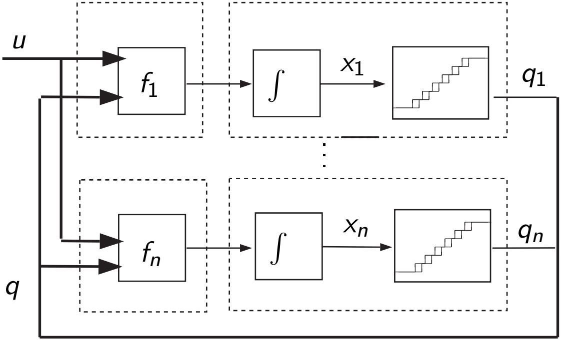
$$\dot{x}(t) = f(q(t), u(t)) \tag{7}$$

where $q(t)$ y $x(t)$ are related componentwise by hysteretic quantization functions.

The QSS of Eq.(7) is equivalent to a legitimate DEVS model.

QSS Method

Block Diagram of a Generic QSS



The QSS method can be applied coupling DEVS models of hysteretic quantized integrators and static functions.

QSS Method – Properties

Perturbed Representation

Defining $\Delta x(t) \triangleq q(t) - x(t)$, we can rewrite Eq.(7) as

$$\dot{x}(t) = f(x(t) + \Delta x(t), u(t)) \tag{8}$$

which is similar to Eq.(6) except for the perturbation term Δx . Notice also that

$$|\Delta x_i| \leq \Delta Q_i, \quad i = 1, \ldots, n \tag{9}$$

Properties related to convergence, stability, and accuracy can be studied as effects of bounded perturbations.

QSS Method – Properties

Linear Time Invarying Systems

When we apply the QSS method to an LTI asymptotically stable system, defining the simulation error as $e(t) \triangleq x(t) - x_a(t)$, it results that

$$|e(t)| \leq |V| \cdot |\mathbb{R}\mathrm{e}(\Lambda)^{-1}\Lambda| \cdot |V^{-1}| \cdot \Delta Q \tag{10}$$

Thus,

- QSS gives always practically stable results. This property is very important taking into account that the method is fully explicit.
- We can calculate a simulation global error bound.

QSS Method
Example

The following equations represent a mass–spring–damper system.

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = -\frac{k}{m}x_1(t) - \frac{b}{m}x_2(t) + \frac{1}{m}F(t)$$

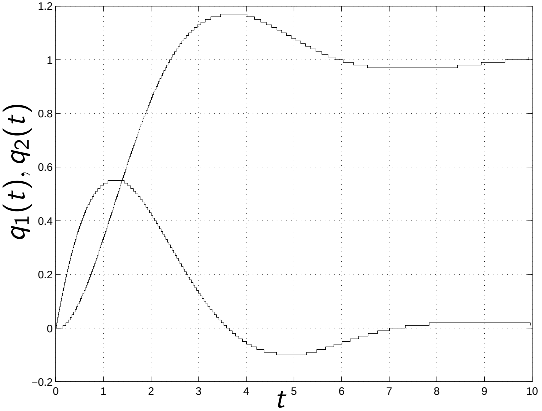
and the QSS approximation is

$$\dot{x}_1(t) = q_2(t)$$
$$\dot{x}_2(t) = -\frac{k}{m}q_1(t) - \frac{b}{m}q_2(t) + \frac{1}{m}F(t)$$

For the parameters $m = b = k = 1$, the simulation error bound is

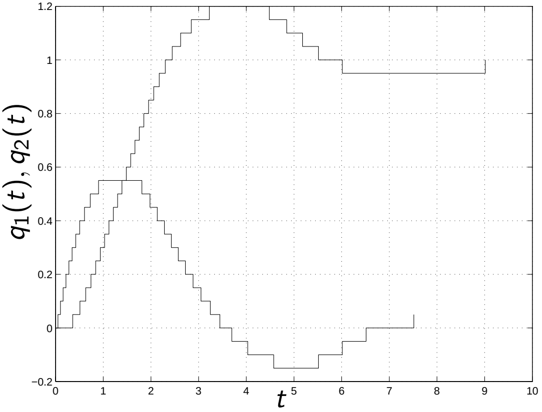
$$\begin{bmatrix} |e_1(t)| \\ |e_2(t)| \end{bmatrix} \leq 2.3094 \cdot \begin{bmatrix} \Delta Q_1 + \Delta Q_2 \\ \Delta Q_1 + \Delta Q_2 \end{bmatrix}$$

QSS Method
Simulation Results



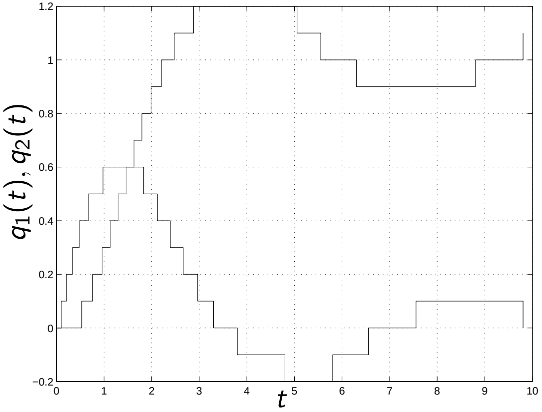
$$\Delta Q_i = 0.01 \ .$$

QSS Method
Simulation Results



$\Delta Q_i = 0.05$.

QSS Method
Simulation Results



$\Delta Q_i = 0.1$.

QSS Method
Main Features

Advantages

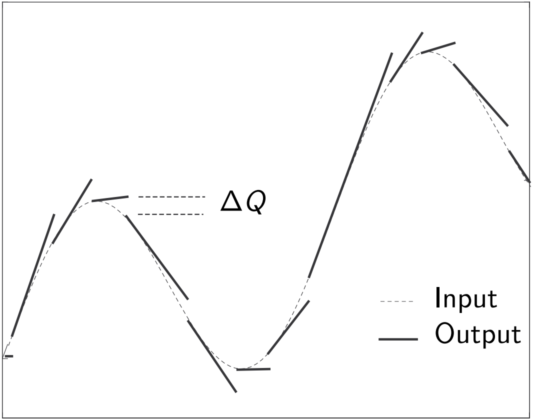
- Stability and Error Bound.
- Decentralization (only local calculations at each step). Sparsity exploitation
- Can reduce the number of iterations in some DAEs.
- Very efficient discontinuity handling

Disadvantages

- Problems with stiff systems.
- We have to choose the quantum.
- **The number of steps grows linearly with the accuracy.**

QSS2 Method
First Order Quantization

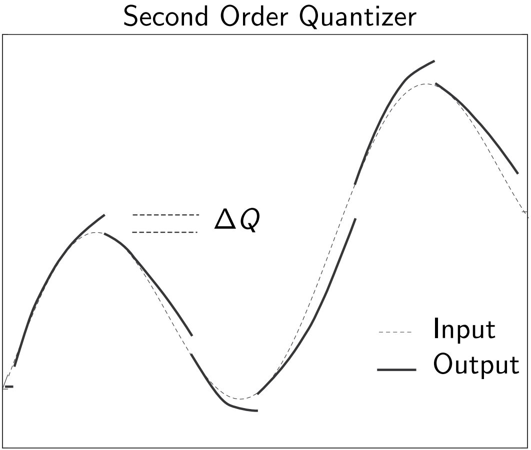
First Order Quantizer



QSS2 Method
Main Features

- It follows the same idea of QSS, but the quantized trajectories are now piecewise linear instead of piecewise constant.
- Each event now must carry two numbers: the initial value and the slope of each trajectory segment.
- The quantized integrators and static functions are more complex, because they must consider and compute the slopes.
- Since the perturbations terms in Eq.(9) are still bounded by ΔQ_i , QSS2 has the same error bound as QSS.
- Now, the number of steps grows with the square root of the accuracy.

QSS3 Method
Second Order Quantization



QSS Methods and Stiff Systems
Introductory example

The LTI system

$$\begin{aligned} \dot{x}_1(t) &= 0.01\,x_2(t) \\ \dot{x}_2(t) &= -100\,x_1(t) - 100\,x_2(t) + 2020 \end{aligned} \tag{11}$$

has eigenvalues $\lambda_1 \approx -0.01$ and $\lambda_2 \approx -99.99$, so it is stiff.

The QSS method approximates this system by

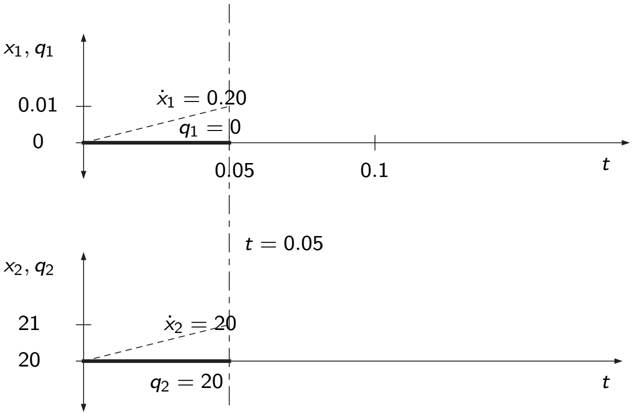
$$\begin{aligned} \dot{x}_1(t) &= 0.01\,q_2(t) \\ \dot{x}_2(t) &= -100\,q_1(t) - 100\,q_2(t) + 2020 \end{aligned} \tag{12}$$

Taking initial conditions $x_1(0) = 0$, $x_2(0) = 20$, and quantization $\Delta Q_1 = \Delta Q_2 = 1$, the QSS integration does the following:

QSS Methods and Stiff Systems

Introductory Example – QSS Simulation

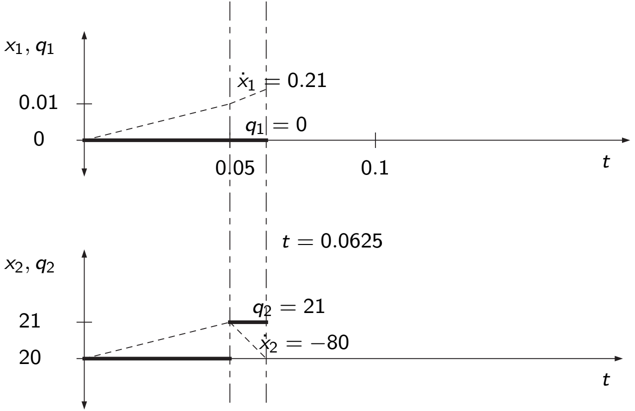
$\dot{x}_1(t) = 0.01\, q_2(t)$ $\dot{x}_2(t) = -100\, q_1(t) - 100\, q_2(t) + 2020$



QSS Methods and Stiff Systems

Introductory Example – QSS Simulation

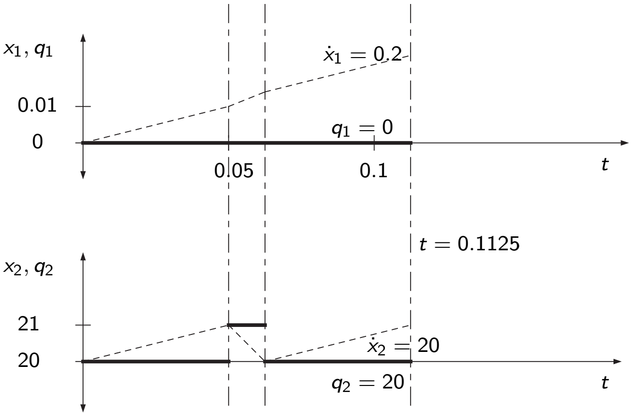
$$\dot{x}_1(t) = 0.01\, q_2(t) \qquad \dot{x}_2(t) = -100\, q_1(t) - 100\, q_2(t) + 2020$$



QSS Methods and Stiff Systems

Introductory Example – QSS Simulation

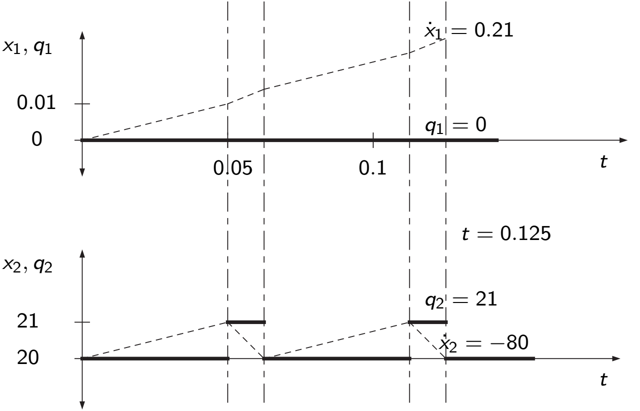
$$\dot{x}_1(t) = 0.01\, q_2(t) \qquad \dot{x}_2(t) = -100\, q_1(t) - 100\, q_2(t) + 2020$$



QSS Methods and Stiff Systems

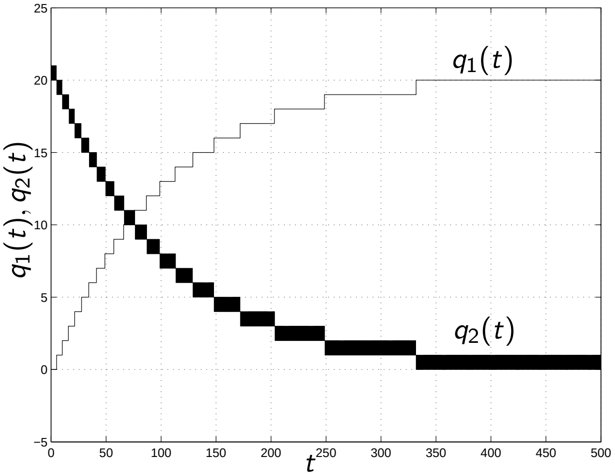
Introductory Example – QSS Simulation

$$\dot{x}_1(t) = 0.01\, q_2(t) \qquad \dot{x}_2(t) = -100\, q_1(t) - 100\, q_2(t) + 2020$$



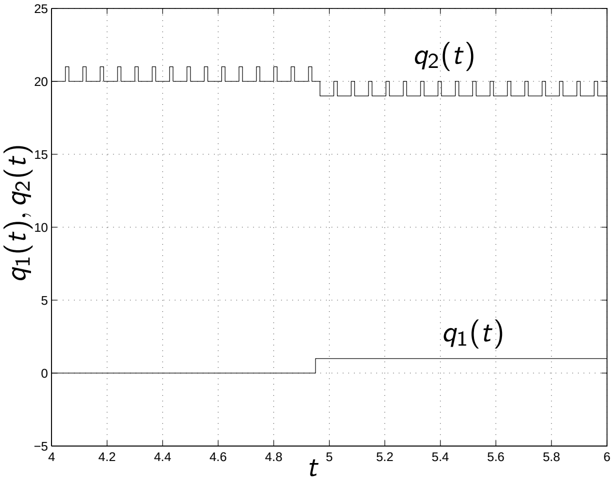
QSS Methods and Stiff Systems

Introductory Example – QSS Simulation



QSS Methods and Stiff Systems

Introductory Example – QSS Simulation (Detail)



QSS Method and Stiff Systems

- Stiff systems provoke fast oscillations on the QSS solutions.
- Thus, the number of steps is huge. In the simulated example there were 21 changes in q_1 and 15995 in q_2 , for a final simulation time $t_f = 500$.

Evidently, the QSS method is not appropriate for the simulation of stiff systems.

Backward QSS

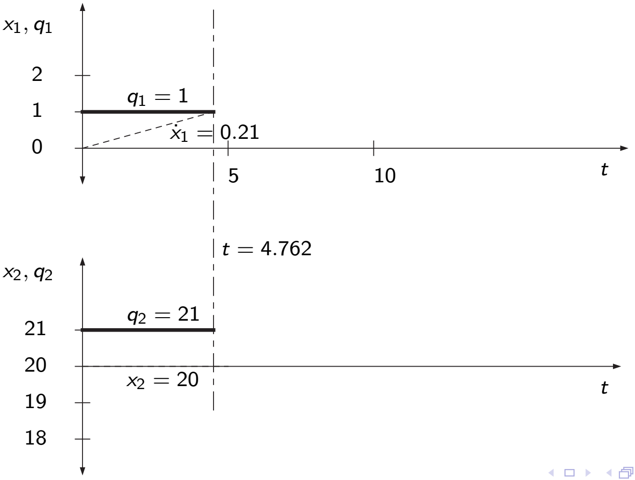
Basic Idea

The idea is that each quantized variable q_j has always a future value of the corresponding state x_j . This is,

- State trajectories always go to the corresponding value of q
- Although this is backward integration, it does not call for iterations, since each variable q_j can only take two values (one from below and the other from above of x_j).

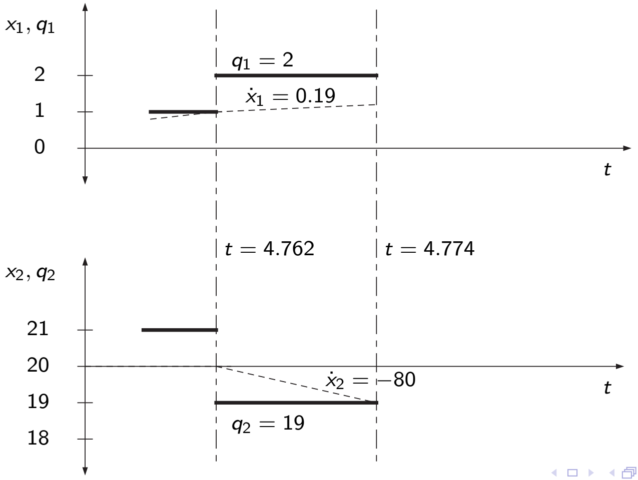
Backward QSS
Example

$\dot{x}_1(t) = 0.01\, q_2(t) \qquad \dot{x}_2(t) = -100\, q_1(t) - 100\, q_2(t) + 2020$

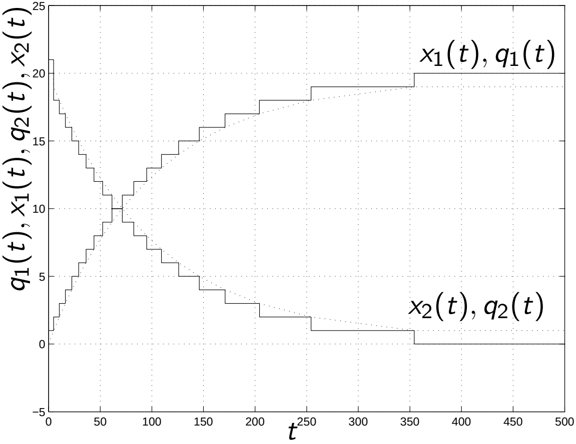


Backward QSS
Example

$\dot{x}_1(t) = 0.01\,q_2(t) \qquad \dot{x}_2(t) = -100\,q_1(t) - 100\,q_2(t) + 2020$



Backward QSS
Example



BQSS Simulation (43 steps).

Backward QSS
Main Fatures

Advantages:

- BQSS is actually an explicit method.
- It shares the main properties of QSS (practical stability, global error bound, etc.).
- Additionally, it can deal with stiff systems.

Disadvantages:

- Like QSS, BQSS is only first order accurate.
- In some nonlinear systems, BQSS finds non existing equilibrium points.

QSS Methods and Marginally Stable Systems
Centered QSS

QSS methods have the same problems that Euler's methods have regarding marginal stability:

- Forward QSS gives unstable simulation results.
- Backward QSS gives asymptotically stable simulation results.

Forward and Backward Euler can be combined to form an F–Stable integration method (the Trapezoidal Rule). Similarly, we can blend QSS and BQSS:

- The idea that each quantized variable takes the mean value of the corresponding QSS and BQSS quantized variables, namely, $q_i = 0.5(q_{i_{QSS}} + q_{i_{BQSS}})$.
- The resulting method is called CQSS (Centered QSS).
- Unlike the trapezoidal rule, CQSS is only first order accurate.

PowerDEVS and QSS Methods

Main Features

- PowerDEVS is a general purpose DEVS simulation tool developed at the Universidad Nacional de Rosario.
- It has block libraries with Quantized Integrators, Static Functions, Hybrid and Source Blocks that implement the whole QSS family (QSS, QSS2, QSS3, BQSS and CQSS).
- PowerDEVS has a GUI that permits drawing Block Diagrams, similar to Simulink.
- It is a free tool.
- PowerDEVS can be downloaded from www.fceia.unr.edu.ar/lsd/powerdevs

Outline

1. Introduction
2. QSS Methods
3. Examples
4. Conclusions and Open Problems

Examples
Mass–Spring–Damper System

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = -\frac{k}{m}x_1(t) - \frac{b}{m}x_2(t) + \frac{1}{m}F(t)$$

See PowerDEVS Model

Examples

Ball Bouncing Down Some Stairs

$$\dot{x} = v_x, \quad \dot{v}_x = -\frac{b_a}{m} \cdot v_x, \quad \dot{y} = v_y$$

$$\dot{v}_y = -g - \frac{b_a}{m} \cdot v_y - s_w \cdot [\frac{b}{m} \cdot v_y + \frac{k}{m}(y - h(x))]$$

where $h(x)$ gives the height of the current stair, and $s_w(t)$ switches between 0 (ball in the air) and 1 (ball on the floor). Namely,

$$h(x) = floor(11 - x), \quad s_w(t) = \begin{cases} 0 & \text{if } y > h(x) \\ 1 & \text{otherwise} \end{cases}$$

See PowerDEVS Model

Examples
Lossless Transmission Line

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

We consider an input pulse entering the line:

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

and a nonlinear load:

$$g(u_n(t)) = (10000 \cdot u_n)^3$$

See PowerDEVS Model

Outline

Conclusions

Main Features of QSS Methods

- Discretizing states instead of time, QSS methods offer a new way of simulating continuous systems.
- They have strong theoretical properties (stability and global error bound).
- QSS methods offer dense output.
- QSS methods show important advantages when handling discontinuities.
- The capability of the explicit methods BQSS and CQSS to deal with stiff and marginally stable systems represents one of the most promising results.

Conclusions
Open Problems

- BQSS and CQSS must be extended to obtain higher order accurate methods.
- PowerDEVS only admits Block Diagram models. It is very important that the QSS methods are implemented to work with object-oriented modeling languages, such as Modelica.
- The use of uniform quantization implicitely controls the absolute error. It is better to have control over the relative error. This issue might be resolved with the usage of logarithmic quantization.
- QSS methods seem to be appropriate for parallel and also real-time simulation. However, these problems have not been studied yet in greater detail.

Bibliography

Main References about QSS Methods

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

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

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

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

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

Bibliography

More References about QSS Methods

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

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

E. Kofman.
Discrete Event Simulation and Control of Continuous Systems.
PhD thesis, Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional de Rosario., 2003.

E. Kofman.
Quantization-based simulation of differential algebraic equation systems.
Simulation, 79(7):363–376, 2003.

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