ITM Web of Conferences **3**, 02002 (2014)
DOI: 10.1051/itmconf/20140302002

A Numerical Study of Quantization-Based Integrators

Fernando Barros[1,a]

[1] *Departamento de Engenharia Informática, Universidade de Coimbra, Portugal*

Abstract. Adaptive step size solvers are nowadays considered fundamental to achieve efficient ODE integration. While, traditionally, ODE solvers have been designed based on discrete time machines, new approaches based on discrete event systems have been proposed. Quantization provides an efficient integration technique based on signal threshold crossing, leading to independent and modular solvers communicating through discrete events. These solvers can benefit from the large body of knowledge on discrete event simulation techniques, like parallelization, to obtain efficient numerical integration. In this paper we introduce new solvers based on quantization and adaptive sampling techniques. Preliminary numerical results comparing these solvers are presented.

1 Introduction

Nowadays, it is considered that the efficient integration of ordinary differential equations (ODEs) requires adaptive step-size solvers [1]. In conventional adaptive solvers the integration step is adjusted to keep numerical stability and accuracy at some desired level, and the same step size, although variable, is used in all ODEs. These methods are thus synchronous since the smallest step size required by an ODE is imposed to all others. Alternative solutions based on asynchronous techniques have been developed based on quantization [2], and generalized sampling [3]. These approaches enable each ODE to be solved by an integrator that can advance time independently from the others, making these methods asynchronous, and not driven by any particular ODE.

Quantization, is a discrete event technique based on the partitioning of signals into levels separated by a fixed interval (quanta). Step size is controlled by the time to reach the a new quantization level, making it an adaptive step-size integrator [2]. Asynchronicity is introduced by considering that an integrator I only affects its influencees, i.e., other integrators directly depending on the output signal (discrete event) of I. An approximation to the input value is made in quantized integrators. Since values are only sent when there is a change in the output signal quantization level, only that value is exact, while the others are considered to be constant. This approximation, however, enables to improve method efficiency. In particular, in distributed simulation, where values can be locally kept, quantized integrators require no communication to obtain the current input value. Quantization levels, need thus, to be chosen in order to make this approximation valid, keeping method stability and the error bounded.

Generalized sampling is an alternative technique that enables the description of asynchronous sampled-based systems, where each component can have its own sampling rate, while keeping the

[a] e-mail: barros@dei.uc.pt

Article available at http://www.itm-conferences.org or http://dx.doi.org/10.1051/itmconf/20140302002

ability to modify component sampling rate [3]. These features enable asynchronous methods, where ODEs are assigned to integrators that can control their sampling rate based, for example, on local error. Conventional synchronous solvers can also be represented through the use of synchronous influenceers, enabling the representation of a wide range of integration solvers [4].

Given these two approaches for representing asynchronous solvers we consider that the development of new methods exploiting synergies may lead to improved numerical solvers. We consider here the representation of the quantized solvers in the Heterogeneous Flow System Specification (HFSS) and we develop two new solvers based on quantization and generalized sampling concepts. The first solver, uses sampling to provide the current value for all variables when an integration step is made. An improved accuracy is expected, while the computation is also increased to obtain the exact input values. A second integrator was developed to exploit the exact input value used in each integration step. Since each transition uses now the current input value and not an approximation, we study an alternative quantization-based method that sets dynamically the next quantization level based on the current output value, as opposed to fixed quantization levels used in the original method. Our preliminary results point that this new approach presents a good balance between accuracy and efficiency leading to fewer transitions than the original method, for an improved accuracy. In these comparisons energy have been used for assessing integration error.

2 The HFSS Formalism

The Heterogeneous Flow System Specification (HFSS) is a formalism aimed to represent piecewise constant partial state systems. This characteristic enables its implementation on digital computers. HFSS achieves the representation of continuous variables using the concept of generalized sampling [5], while the representation of discrete events is based on the Discrete Event System Specification (DEVS) [6]. HFSS has two types of models: basic and network. Basic models provide state representation and state transition functions. Network models are a composition of basic models and/or other network models. Given its definition, a network provides an abstraction for representing hierarchical systems.

2.1 HFSS Basic Model

We consider $\widehat{B}$ as the set of names corresponding to basic HFSS models. A HFSS basic model associated with name $B \in \widehat{B}$ is defined by

$$M_B = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \bar{X} \times \ddot{X}$ is the set of input flow values

$\bar{X}$ is the set of continuous input flow values
$\ddot{X}$ is the set of discrete input flow values

$Y = \bar{Y} \times \ddot{Y}$ is the set of output flow values

$\bar{Y}$ is the set of continuous output flow values
$\ddot{Y}$ is the set of discrete output flow values

P is the set of partial states (p-states)
$\rho : P \longrightarrow \mathbf{H}_0^{+\infty}$ is the time-to-input function

$\omega : P \longrightarrow \mathbf{H}_0^{+\infty}$ is the time-to-output function

$S = \{(p, e) | p \in P, 0 \leq e \leq v(p)\}$ is the state set

with $v(p) = \min\{\rho(p), \omega(p)\}$, representing the time to transition function

$s_0 = (p_0, e_0) \in S$ is the initial state

$\delta : S \times X^{\varnothing} \longrightarrow P$ is the transition function

where $X^{\varnothing} = \bar{X} \times \ddot{X}^{\varnothing}$ and $\ddot{X}^{\varnothing} = \ddot{X} \cup \{\varnothing\}$
and $\varnothing$ represents the null value (absence of value)

$\bar{\Lambda} : S \longrightarrow \bar{Y}$ is the continuous output function

$\lambda : P \longrightarrow \ddot{Y}$ is the partial discrete output function

HFSS components have their behavior ruled by HFSS models and their semantics are presented in [5].

Continuous Flow Generator

Function generation plays an important role in the representation of many continuous systems. Entity trajectories or forces be described with the help of a HFSS function generator. For example, a force described by the function $f : \mathbf{R} \longrightarrow \mathbf{R}$ can be represented by the HFSS model:

$$M_f = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \{\varnothing\} \times \{\varnothing\}$

$Y = \mathbf{R} \times \{\varnothing\}$

$P = \{\varnothing\}$

$\rho(\varnothing) = \omega(\varnothing) = \infty$

$s_0 = (\varnothing, 0)$

$\delta((\varnothing, 0), (\varnothing, \varnothing)) = (\varnothing, 0)$

$\bar{\Lambda}(\varnothing, e) = f(e_{std})$, where e_{std} is the standard part of the hyperreal e

$\lambda(\varnothing) = \varnothing$

This model can represent arbitrary continuous functions. For example, an exponential signal can be described by the function $f(t) = \alpha e^{-\beta t}$. This generator uses only continuous flows to describe the signal. The generator is a passive component with no autonomous behavior, and where all information is retrieved through sampling.

Square Wave Generator

Hybrid generators can also be represented in HFSS. A square wave can be described by an autonomous model without no inputs and two outputs: a discrete flow to signal a new output value and a continuous flow holding the last output value. The generator of a square wave of amplitudes -1, +1 and period π is given by:

$$M_{\square} = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)_{\pi}$$

where

$X = \{\varnothing\} \times \{\varnothing\}$
$Y = \{-1, 1\} \times \{-1, 1\}$
$P = \{(\beta, y_c, y_d) | \beta \in \mathbf{H}, y_c, y_d \in \mathbf{R}\}$
$\rho(\beta, y_c, y_d) = \infty$
$\omega(\beta, y_c, yd) = \beta$
$s_0 = (0, -1, 1)$
$\delta((\beta, 1, -1), (\varnothing, \varnothing)) = (\pi, \pi, -1, 1)$
$\delta((\beta, -1, 1), (\varnothing, \varnothing)) = (\pi, \pi, 1, -1)$
$\bar{\Lambda}((\beta, y_c, y_d), e) = y_c$
$\lambda(\beta, y_c, y_d) = y_d$

Typical generator trajectories are depicted in Figure 1. The discrete flow signals a discontinuity, while the wave value is described by the continuous output flow. Models can sample this value and they can also be aware of value change. We note that a DEVS models can produce discrete events but they cannot hold these values as piecewise constant signals.

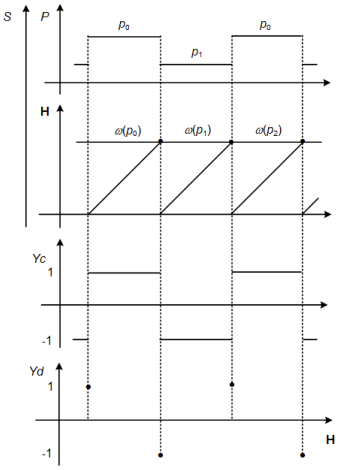

Figure 1. Square wave generator.

The persistence of the output values are used in Section 3 to represent quantized-based integrators, removing the need to store event values, required by discrete event implementations of quantized systems [6].

2.2 HFSS Network Model

HFSS networks models are compositions of HFSS models (basic or other HFSS networks models). Let $\widehat{N}$ be the set of names corresponding to HFSS network models, with $\widehat{N} \cap \widehat{B} = \{\}$. Formally, a HFSS network model associated with name $N \in \widehat{N}$ is defined by

$$M_N = (X, Y, \eta)$$

where

N is the network name

$X = \bar{X} \times \ddot{X}$ is the set of network input flows

$\bar{X}$ is the set of network continuous input flows

$\ddot{X}$ is the set of network discrete input flows

$Y = \bar{Y} \times \ddot{Y}$ is the set of network output flows

$\bar{Y}$ is the set of network continuous output flows

$\ddot{Y}$ is the set of network discrete output flows

$\eta \in \widehat{\eta}$ is the name of the dynamic topology network executive

with

$\eta \in \widehat{\eta}$ representing the set of all names associated with HFSS executive models, constrained to $\widehat{\eta} \cap \widehat{B} = \widehat{\eta} \cap \widehat{N} = \{\}$

Executives are uniquely assigned to network models, i.e.,

$\forall_{i,j \in \widehat{N}, i \neq j} \eta_i \neq \eta_j$ with $M_k = (X_k, Y_k, \eta_k), \forall_{k \in \widehat{N}}$

The model of the executive is a modified HFSS model, defined by

$$M_\eta = (X_\eta, Y_\eta, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda, \widehat{\Sigma}, \gamma), \text{ for } \eta \in \widehat{\eta}$$

where

$\widehat{\Sigma}$ is the set of network topologies

$\gamma : P \longrightarrow \widehat{\Sigma}$ is the topology function

The network topology $\Sigma_\alpha \in \widehat{\Sigma}$, corresponding to the p-state $p_\alpha \in P$, is given by the 4-tuple

$$\begin{aligned}\Sigma_\alpha =& \gamma(p_\alpha) = (C_\alpha, \\ & \{I_{i,\alpha}\} \cup \{I_{\eta,\alpha}, I_{N,\alpha}\}, \\ & \{E_{i,\alpha}\} \cup \{E_{\eta,\alpha}, E_{N,\alpha}\}, \\ & \{F_{i,\alpha}\} \cup \{F_{\eta,\alpha}, F_{N,\alpha}\})\end{aligned}$$

where

C_α is the set of names associated with the executive state p_α

for all $i \in C_\alpha \cup \{\eta\}$

$I_{i,\alpha}$ is the sequence of asynchronous influencers of i

$E_{i,\alpha}$ is the sequence of synchronous influencers of i

$F_{i,\alpha}$ is the input function of i

$I_{N,\alpha}$ is the sequence of asynchronous network influencers

$E_{N,\alpha}$ is the sequence of synchronous network influencers

$F_{N,\alpha}$ is the network output function

For all $i \in C_\alpha$

$M_i = (X_i, Y_i, P_i, \rho_i, \omega_i, s_{0,i}, \delta_i, \bar{\Lambda}_i, \lambda_i)$ if $i \in \widehat{B}$

$M_i = (X_i, Y_i, \eta_i)$ if $i \in \widehat{N}$

Variables are subjected to the following constraints for every $p_\alpha \in P_\alpha$

$N \notin C_\alpha, N \notin I_{N,\alpha}, \eta \notin C_\alpha$

$N \notin E_{i,\alpha}$ for all $i \in C_\alpha \cup \{\eta, N\}$

$F_{N,\alpha} : \times_{k \in I_{N,\alpha}} Y_k \longrightarrow Y^\varnothing$

$F_{i,\alpha} : \times_{k \in I_{i,\alpha}} V_k \longrightarrow X_i^\varnothing$

where

$$V_k = \begin{cases} Y_k^\varnothing & \text{if } k \neq N \\ X^\varnothing & \text{if } k = N \end{cases}$$

$F_{N,\alpha}((\bar{v}_{k_1}, \varnothing), (\bar{v}_{k_2}, \varnothing), ...) = (\bar{y}_N, \varnothing)$

$F_{i,\alpha}((\bar{v}_{k_1}, \varnothing), (\bar{v}_{k_2}, \varnothing), ...) = (\bar{x}_i, \varnothing)$

These two last constraints are characteristics of discrete systems and impose that non-null values cannot be created from a sequence composed exclusively by null values.

Contrarily, to other modeling hybrid formalisms [6], HFSS uses the set of influenceers and not the set of influencees to describe model interaction. This choice enables the interoperabilty and reuse of models having different interfaces. In this paper we exploit the ability to combine several output values into a single value required by integrators, making them reusable since they are independent on the set of influencers and the specific ODE. This is not the case for the integrators described by DEVS that need to be modified according the these parameters [6].

3 Quantization-Based Integrators

The efficient integration of ODEs requires the use of adaptive step size solvers that adjust the step in order to keep error within some bound. Conventional solvers adjust the sampling rate, performing an integration step when any of the equations exceeds the maximum allowed error. This type of integrator is synchronous, since all equations are integrated at the same (although variable) time step. Discrete event integrators are based on the concept of quantization, that independently sets the integration step to each equation based on the variation of integrator output value. A quantized-integrator (QI) computes the time to reach the next quantization level and at that time then it sends the output value as a discrete event value. When a QI receives a value it uses the remaining stored input values to compute the new time to reach the next quantization value. Given this algorithm we provide now its description in the HFSS formalism [5]. HFSS provides two main advantages over a conventional discrete event implementation. First, there is no need to store input values in the integrator, and second, the integrator can be made independent of the ODE. Let us take as an example the ODE $\dot{y} = \cos(x) + z^3$, that depends on variables x and z, that are computed by components X and Y, respectively. When x crosses its own quantization value, component X sends a discrete event to Y, that performs the computation

$\cos(x_{new}) + z^3_{curr}$, where x_{new} is the new value of x and z_{curr} the last received value of z. A discrete event representation requires the integrator to be aware of the ODE, limiting its reuse. This kind of representation needs also to store the last value of each received input. A HFSS version of the integrator enables a simplification of this procedure. Since HFSS components have continuous output flows, if we design integrator outputs as piecewise constant flows, we can remove the need for local storage of input values. We can also exploit HFSS input function to describe ODE functions, making integrators domain independent an fully reusable.

Classical Quantized Integrator (QI-1)

We start our description by the classical quantized integrator [2]. The HFSS model of an integrator associated with quantum Q can be defined by:

$$M_{QI-1} = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \mathbf{R} \times \mathbf{R}$

$Y = \mathbf{R} \times \mathbf{R}$

$P = \{(\alpha, \beta, x, y, q) | \alpha, \beta \in \mathbf{H}_0^+; x, y, q \in \mathbf{R}\}$

$\rho(\alpha, \beta, x, y, q) = \alpha$

$\omega(\alpha, \beta, x, y, q) = \beta$

$s_0 = ((0, \infty, 0, y_0, y_0), 0)$

$\delta(((\alpha, \beta, x, y, q), e), (x_c, x_d)) =$

$$\begin{cases} (\infty, |\frac{Q}{x_c}|, x_c, y', y') & \text{if } e = \beta + \epsilon \\ (\infty, \tau, x_c, y', q) & \text{otherwise} \end{cases}$$

where $y' = y + x \cdot e_{std}$ and

$$\tau = \begin{cases} \frac{q+Q-y'}{x_c} & \text{if } x_c \geq 0 \\ \frac{q-Q-y'}{x_c} & \text{otherwise} \end{cases}$$

$\bar{\Lambda}((\alpha, \beta, x, y, q), e) = y$

$\lambda(\alpha, \beta, x, y, q) = y$

where q is the last discrete event output sent, and used to compute the time interval for the next quantum cross. To obtain quantization semantics, we need also to combine events with the piecewise constant values from the unchanged integrators. This can be achieved by component input function, that needs to replace the continuous output flow of a changing component by its discrete output value.

Quantized Integrator based on Exact Inputs (QI-2)

Another possibility is to use the concept of quantization while providing a continuous output flow to the solvers. The new integrator can be described by:

$$M_{QI-2} = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \mathbf{R} \times \mathbf{R}$

$Y = \mathbf{R} \times \mathbf{R}$

$P = \{(\alpha, \beta, x, y, q) | \alpha, \beta \in \mathbf{H}_0^+; x, y, q \in \mathbf{R}\}$

$\rho(\alpha, \beta, x, y, q) = \alpha$

$\omega(\alpha, \beta, x, y, q) = \beta$

$s_0 = ((0, \infty, 0, y_0, y_0), 0)$

$\delta(((\alpha, \beta, x, y, q), e), (x_c, x_d)) =$

$$\begin{cases} (\infty, |\frac{Q}{x_c}|, x_c, y', y') & \text{if } e = \beta + \epsilon \\ (\infty, \tau, x_c, y', q) & \text{otherwise} \end{cases}$$

where $y' = y + e_{std} \cdot x$ and

$$\tau = \begin{cases} \frac{q+Q-y'}{x_c} & \text{if } xc \geq 0 \\ \frac{q-Q-y'}{x_c} & \text{otherwise} \end{cases}$$

$\bar{\Lambda}((\alpha, \beta, x, y, q), e) = y + x \cdot e_{std}$

$\lambda(\alpha, \beta, x, y, q) = y$

The difference from the previous integrator is in the continuous output function, that now provides a continuous output, as opposed to the piecewise constant value of the previous version. This integrator does not require any replacement in the input value set since all values, including the one corresponding to the active integrator, are exact.

Dynamic Quantization Integrator (QI-3)

The last version exploits the exact input used by the previous integrator when it makes a transition. Since all inputs are updated when sampled, we consider that the next threshold crossing level can be also modified. This makes the next quantization level dynamic, depending on the value set by the last transition.

$$M_{QI-3} = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \mathbf{R} \times \mathbf{R}$

$Y = \mathbf{R} \times \mathbf{R}$

$P = \{(\alpha, \beta, x, y) | \alpha, \beta \in \mathbf{H}_0^+; x, y \in \mathbf{R}\}$

$\rho(\alpha, \beta, x, y) = \alpha$

$\omega(\alpha, \beta, x, y) = \beta$

$s_0 = ((0, \infty, 0, y_0), 0)$

$\delta(((\alpha, \beta, x, y), e), (x_c, x_d)) =$

$$(\infty, |\frac{Q}{x_c}|, x_c, y + e_{std} \cdot x)$$

$\bar{\Lambda}((\alpha, \beta, x, y), e) = y + x \cdot e_{std}$

$\lambda(\alpha, \beta, x, y) = y$

Comparing Discrete Events and Hybrid Flows

The original integrator QI-1 was mainly designed to offer an efficient implementation in distributed computing environments [2]. In this case, a piecewise constant value can be buffered avoiding its transmission to a component that can be located in a different computer, an expensive operation due to communication overhead. This overhead, however, is usually negligible when components are in the same computer. The use of of continuous output values, as provide by HFSS, introduces usually a small overhead, specially when simulations are run on non-distributed computers, enabling the use of exact value instead of approximations. The quantization supported by QI-1 makes it very convenient to a distributed implementation since only one bit (up, down) is required to convey the information. However, this feature can become irrelevant in non-distributed simulations. HFSS obtains the exact output value when a integrator transition is scheduled. This approach my increase integration accuracy when compared to quantization-based solutions that use approximated output values [7].

4 Numerical Results

To provide a preliminary assessment of the described quantization-based integrators we have used two simple systems: an harmonic oscillator and a pendulum. Comparisons use energy for assessing method accuracy.

Harmonic oscillator

As a first example, we consider the harmonic oscillator described by the equation:

$$\ddot{x} = -x$$

and energy given by

$$E = \frac{1}{2}\dot{x}^2 + \frac{1}{2}x^2$$

Considering $x(0) = 0$ and $v(0) = 4$ we have $E = 8J$. Since this oscillator is an Hamiltonian system, energy should remain constant. Simulation results are depicted in Figure 2 that describes position and velocity. (For simplicity, parameters like elasticity, mass, and gravity, required to characterize this system have been set to one.)

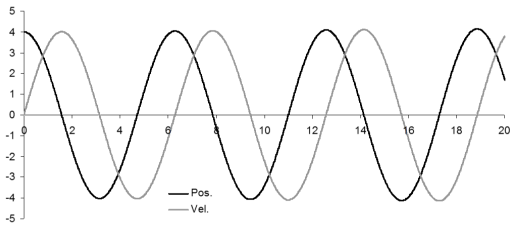

Figure 2. Harmonic oscillator position and velocity.

Oscillator energy for the three methods is represented in Figure 3, where a quantum size of 0.01 was used. It is well know that general purpose integrators have poor characteristics for simulating

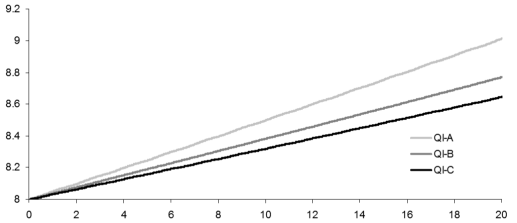

Figure 3. Energy for the harmonic oscillator.

Table 1. Number of transitions in harmonic oscillator simulation.

QI-1	QI-2	QI-3
10341	10309	7923

Hamiltonian systems [1], and we expect energy to change. We note however, that integrators error accumulate at different rates, being dependent on the solver.

The number of transitions is given in Table 1. The best accuracy was obtained by QI-2 while QI-3 has fewer transitions and better accuracy than QI-1.

Planar Pendulum

We consider next the pendulum described by the equation:

$$\ddot{\theta} = -\sin(\theta)$$

and energy given by

$$E = \frac{1}{2}\dot{\theta}^2 + 1 - \cos(\theta)$$

Initial conditions are: $v = 4$, $\theta = 0$, and for simplicity $g = 1, m = 1, L = 1$, where L is pendulum length. For these values $E = 0.5J$. Pendulum energy is depicted in Figure 4, where a quantum of 0.01 was used. The original QI-1 has the largest error, QI-2 has the smallest error, while QI-3 shows an intermediate accuracy.

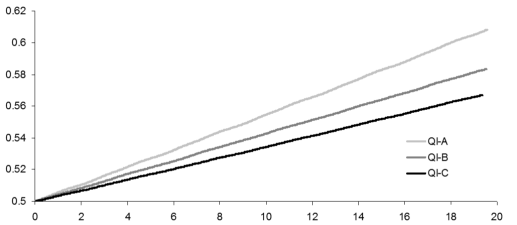

Figure 4. Energy for the pendulum system.

Table 2. Number of transitions in pendulum simulation.

QI-1	QI-2	QI-3
2456	2426	1711

The number of transitions is given in Table 2. Likewise the previous system, the best accuracy is obtained by QI-2, while QI-3 exhibits fewer transitions and better accuracy than QI-1.

5 Conclusion and Future Work

Quantized-based integrators have a large potential for reducing computation effort. We have described two new integrators based on quantization that, according to our preliminary results, have a good performance when compared with the original quantization integrator. The HFSS formalism have shown a good degree of flexibility by providing a framework for representing there different types of quantization-based integrators. Additional work is required to confirm our preliminary numerical results. A general proof of the characteristics of the presented quantization-based integrators needs also to be addressed by future research.

References

[1] E. Hairer, S. Nørsett, G. Wanner, *Solving Ordinary Differential Equations I: Non Stiff Problems* (Springer, 2000)

[2] B. Zeigler, J. Lee, *Theory of Quantized Systems: Formal Basis for DEVS/HLA Distributed Simulation Environment*, in *Enabling Technology for Simulation Science II* (1998), Vol. 3369 of *SPIE*, pp. 49–58

[3] F. Barros, *Towards a Theory of Continuous Flow Models*, in *International Journal of General Systems* (2002), Vol. 31, pp. 29–39

[4] F. Barros, *Comparing Synchronous and Asynchronous Variable Step Size Explicit ODE Solvers: A Simulation Study*, in *21st International Workshop on Principles of Advanced and Distributed Simulation* (2007), pp. 32–37

[5] F. Barros, *Semantics of Discrete Event Systems*, in *Distributed Event-Based Systems* (2008), pp. 252–258

[6] B. Zeigler, H. Praehofer, T. Kim, *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems* (Academic Press, 2000)

[7] G. Migoni, M. Bortolotto, E. Kofman, F.E. Cellier, *Linearly implicit quantization-based integration methods for stiff ordinary differential equations*, in *Simulation Modelling Practice and Theory* (2013), Vol. 35, p. 118–136