Soft Computing (2023) 27:8189–8208
https://doi.org/10.1007/s00500-022-07715-3

APPLICATION OF SOFT COMPUTING

Advantages of the usage of the Infinity Computer for reducing the Zeno behavior in hybrid system models

Alberto Falcone[1] · Alfredo Garro[1] · Marat S. Mukhametzhanov[1] · Yaroslav D. Sergeyev[1,2,3]

Accepted: 27 November 2022 / Published online: 16 December 2022

Abstract
To capture the dynamics of modern Cyber-Physical Systems, hybrid system models are introduced to combine their continuous dynamics with the discrete ones. Unfortunately, one important negative issue can affect hybrid system models: the so-called *Zeno phenomenon*, which results in an infinite number of discrete transitions in a finite amount of time occurring during the model's simulation that leads to inconsistent results. In this context, the paper investigates the use of a recently proposed numerical algorithm, based on the *Infinity Computer* methodology, to handle the *Zeno phenomenon* and evaluate it with respect to standard numerical methods by considering the hybrid system models of two exemplary Cyber-Physical Systems: the *Water tanks* and the *Thermostat*.

Keywords Hybrid systems · Zero-crossing · Zeno effect · Infinity Computer · Numerical infinitesimals

1 Introduction

To design and analyze Cyber-Physical Systems (CPSs), presenting both continuous and discrete dynamics, *hybrid system* models have been introduced and are widely adopted [see, e.g., Bocciarelli et al. (2019), Falcone and Garro (2019), Grossman et al. (1993); Lunze and Lamnabhi-Lagarrigue (2009)]. A hybrid system can be defined as a system whose dynamic is regulated through continuous and discrete behaviors. Ordinary differential equations (ODEs) are often used to govern the continuous behavior, whereas control graphs can be used to describe the discrete behavior. The values of continuous variables in a particular discrete mode define the state of a hybrid system [see, e.g., Bouskela et al. (2021), Platzer (2008)]. More in detail, continuous states can change their values in two ways: (i) through a discrete transition or (ii) based on the differential equations results. Discrete states can only alter values through discrete transitions. As a consequence, hybrid systems can be considered an efficient formalism for modeling and simulating CPSs, where digital components interact with a physical environment, and their dynamics are regulated through continuous and discrete behaviors.

One of the most important issues unique to hybrid systems is the *Zeno phenomenon* that can happen when the system undergoes an unbounded number of discrete transitions in a finite and bounded length of time [see, e.g., Johansson et al. (1999), Zhang et al. (2001)]. To explain this phenomenon and its effects, suppose to model and simulate a ball that is dropped from a predefined height $h_0 > 0$ with an initial velocity $v_0 = 0$. The ball in free fall hits the ground after a certain time t in partially elastic way, loses energy, bounces back into the air, and then starts falling again until it stops. Starting from this physical system, a hybrid automata is designed in order to model the behavior of the bouncing ball [see Falcone et al. (2022), Johansson et al. (1999) for details]; in particular, ODEs are used to model the variation of h and v during the motion of the ball, whereas transitions rules are

✉ Yaroslav D. Sergeyev
yaro@dimes.unical.it; yaro@icar.cnr.it

Alberto Falcone
alberto.falcone@dimes.unical.it

Alfredo Garro
alfredo.garro@dimes.unical.it

Marat S. Mukhametzhanov
m.mukhametzhanov@dimes.unical.it

1 Department of Informatics, Modeling, Electronics and Systems Engineering (DIMES), University of Calabria, 87036 Rende, Cosenza, Italy

2 Institute of Information Technology, Mathematics and Mechanics, Lobachevsky State University of Nizhni Novgorod, Nizhni, Novgorod, Russia 603950

3 Institute of High Performance Computing and Networking of the National Research Council of Italy, Rende, Italy

introduced to model the hit of the ball with the ground causing the ball to bounce. As the ball loses energy when it hits the ground, with the progress of time, a large number of discrete transitions occur progressively in the hybrid automata in smaller and smaller time intervals; thus, the automata experiences the Zeno phenomenon. This behavior leads to have, in a simulated environment, incorrect data (e.g., negative values for h) resulting in unfeasible behavior (e.g., the ball goes below the ground level) [see Fritzson (2014) for details].

It is important to note that the real-world CPSs do not show Zeno behavior, but their models may exhibit Zeno executions due to modeling abstractions [see, e.g., Heymann et al. (2005)]. The Zeno phenomenon is difficult to characterize and mitigate because it leads to incorrect simulation results since the system behavior is intrinsically ill-conditioned beyond the point in time (called Zeno point) in which such a phenomenon starts [see, e.g., Branicky (2005), Zhang et al. (2000)]. In this situation, it is necessary to carefully check the prerequisites for discrete events' transitions.

Different approaches and techniques for addressing the Zeno phenomenon have been proposed in the literature. In order to assess the effect on real-world systems, some researchers have examined this issue when hybrid system automata are derived as abstractions of the underlying physical systems and defined the rules associated with this phenomenon [see, e.g., Ames and Sastry (2004), Falcone and Garro (2020), Heymann et al. (2005), Lygeros et al. (2003)]. Other researchers have used *regularization* and *sliding mode* approaches to extend the simulation of Zeno hybrid systems beyond the Zeno point. For example, in Johansson et al. (1999), the authors delineated the characteristics of hybrid automata and employed regularization approaches to simulate and evaluate these automata beyond the Zeno point. However, in order to apply regularization approaches it is necessary to alter the original system by perturbing it with a small quantity ϵ to obtain a non-Zen solution. If $\epsilon = 0$, the regularized system tends to be the original one. However, such perturbation ϵ may invalidate the notion of instantaneous discrete transitions and the simulation performance may degrade. The other approach to handle the Zeno phenomenon regards the use of a *sliding mode* algorithm that is based on detecting regions on the switching manifold on which the Zeno phenomenon occurs, and then forcing the system to slide on the manifold in these regions [see, e.g., Utkin (2013), Weiss et al. (2015), Yu et al. (2011)]. When infinitely fast transitions occur, a smooth sliding action takes place on the switching surface to eliminate the Zeno phenomenon. It is also possible to modify the original hybrid system by adding the extra mode to represent the dynamics during the mode transitions, but *sliding mode* algorithms can only be applied to study special classes of hybrid systems [see, e.g., Biák et al. (2013), Filippov (2013)].

From a numerical point of view, extending the simulation of Zeno hybrid systems beyond the Zeno point means the definition of fast and efficient procedures for the determination of zero-crossings of the transition conditions. Indeed, classical numerical algorithms for simulating hybrid system models generate observations $t_k = t_{k-1} + \Delta t_{k-1},\ k \in \{1, 2, ...\}$, of time $t \in [t_0, T]$ in different ways. The standard method consists of using a fixed stepsize $\Delta t_k = \Delta t = \omega$ with $k \in \{0, 1, 2, ...\}$, but there exist other methods for generating observations dynamically, such as MATLAB/Simulink procedures using Dormand–Prince methods [see, e.g., Kimura (2009)]. However, the main objective of these methods is to solve the ODEs used to describe the system in an efficient way, and not to determine the zero-crossings. This means that these methods are useful to simulate hybrid system models under a fixed state and a fixed set of ODEs, but they do not allow determining whether a zero-crossing occurs between two consecutive observations t_k and t_{k+1}.

To overcome the issues presented above, in Falcone et al. (2022), an efficient method, called *Infinity Computer algorithm using Runge–Kutta method of the fourth order (IC_RK4)*, to simulate hybrid system models by generating observations $t_k,\ k \in \{0, 1, 2, ...\}$, dynamically has been proposed. The proposed method allows studying better the regions, where zero-crossings can occur, without spending a lot of computational resources on "stable" regions. This is done because, in a hybrid system, the continuous behavior is performed as long as invariants remain, whereas discrete transitions (or events) occur when a specific jump condition is satisfied. As a consequence, in the IC_RK4 method, a jump condition is associated with zero-crossings of a given function $g(x(t))$. This indicates that an event occurs at a time t, when there have been generated the system variables $x = x(t)$, such that $g(x(t)) = 0$ [see, e.g., Ames et al. (2006)]. The IC_RK4 method has been developed taking into consideration the following characteristics:

- *Zero-crossing detection*. Zero-crossings are detected by evaluating approximations of the original function $g(x(t))$, without resorting to the original hybrid system. This characteristic allows efficient detection of zero-crossings without requiring additional computing resources, even if the original system is difficult to simulate;
- *Dynamic generation of observations*. The built-in algorithm allows generating observations more frequently around zero-crossing regions, and less frequently elsewhere, avoiding unnecessary computations in "stable" regions, where no zero-crossings can happen;
- *Extensibility*. It can be easily extended to integrate any fast and efficient method for detecting zero-crossings, such as global optimization algorithms [see, e.g., Casado et al. (2002), Molinaro and Sergeyev (2001)].

The IC_RK4 method adopts the Infinity Computer that represents a new kind of supercomputer that allows one to work numerically with infinite and infinitesimal numbers [see, e.g., Sergeyev (2010, 2017)] in a novel framework different w.r.t. nonstandard analysis [see Sergeyev (2019)].

The Infinity Computer allows one to work numerically with finite, infinite, and infinitesimal numbers. It introduces a new numeral ①, called *grossone*, that represents the number of elements of the set, $\mathbb{N}$, of natural numbers and it is used as the radix of a positional numeral system, in which a number C (finite, infinite, or infinitesimal), known as *grossnumber*, is expressed in the following form [see, e.g., Falcone et al. (2020b), Sergeyev (2017) for details]:

$$C = c_1 \cdot ①^{p_1} + c_2 \cdot ①^{p_2} + \cdots + c_N \cdot ①^{p_N}, \tag{1}$$

where c_i, $i = 1, ..., N$, named *grossdigits*, are finite positive or negative real numbers, whereas p_i, $i = 1, ..., N$, known as *grosspowers*, are arranged in decreasing order; thus, $p_1 > p_2 > \cdots > p_N$, and they can in turn be finite, infinite, or infinitesimal quantities of the same form (1). According to this methodology, one of the following three situations can happen:

- $p_1 > 0$. The number C is infinite; thus, $① = 1①^1$ is the basic infinite number;
- $p_1 < 0$. The number C is infinitesimal; thus, $①^{-1}$ represents the simplest infinitesimal;
- $p_1 = 0$. The number C is finite. Specifically, if $N = 1$, then C is known as *purely finite*, i.e., there are no infinite nor infinitesimal quantities.

Arithmetic operations between grossnumbers are carried out in the Infinity Computer as follows. Given the *grossnumbers* A, B, and C specified as:

$$A = \sum_{i=1}^{K} a_{k_i} ①^{k_i},\ B = \sum_{j=1}^{M} b_{m_j} ①^{m_j},\ C = \sum_{i=1}^{L} c_{l_i} ①^{l_i}. \tag{2}$$

The *addition* operation is carried out by including in the result C the items of A, $a_{k_i}①^{k_i} : k_i \neq m_j$ with $1 \leq j \leq M$, with those of B, $b_{m_j}①^{m_j} : m_j \neq k_i$ with $1 \leq i \leq K$, whereas the items $a_{k_i}①^{k_i}$, $b_{m_j}①^{m_j}$ with $k_i = m_j$ generate the *grossdigit* $(a_{k_i} + b_{k_i})①^{k_i}$ in the result C.

The operation of *subtraction* is directly derived from *addition*.

The result of multiplying A and B produces a grossnumber C, defined as follows:

$$C = \sum_{j=1}^{M} C_j, \quad 1 \leq j \leq M, \tag{3}$$

where

$$C_j = b_{m_j} ①^{m_j} \cdot A = \sum_{i=1}^{K} a_{k_i} b_{m_j} ①^{k_i + m_j}. \tag{4}$$

Finally, the *division* operation of A by B yields a result C with a reminder R, where the first *grossdigit* is $c_{k_K} = a_{l_L}/b_{m_M}$, the highest exponent is $k_K = l_L - m_M$, and the first partial reminder R^* is produced as follows: $R^* = A - c_{k_K}①^{k_K} \cdot B$. If $R^* = 0$ or the default accuracy is achieved, the division operation is complete; otherwise, the computation is restarted by substituting A with R^*.

To briefly describe how the Infinity Computer performs the arithmetic operations described above, given two grossnumbers $A = 5.0①^{7.2}3.4①^{0} - 2.1^{-1.5}$ and $B = 3.0①^{3.0}2.0①^{0}$. The addition operation $C = A + B$ returns the result $C = 5.0①^{7.2}3.0①^{3.0}5.4①^{0} - 2.1①^{-1.5}$; the subtraction operation $C = A - B$ returns the result $C = 5.0①^{7.2} - 3.0①^{3.0}1.4①^{0} - 2.1①^{-1.5}$; multiplying $C = A \cdot B$ yields the result $C = 15.0①^{10.2}10.0①^{7.2}10.2①^{3.0} - 6.3①^{1.5}6.8①^{0} - 4.2①^{-1.5}$; and finally, the operation $C = A \div B$ returns the result:

$$C = 1.6667①^{4.2} - 1.1111①^{1.2}0.7407①^{-1.8}1.1333①^{-3.0} - 0.7000①^{-4.5} - 0.4938①^{-4.8}.$$

The Infinity Computer has been largely exploited in many research domains for solving remarkable problems, such as numerical problems involving ODEs [see, e.g., Amodio et al. (2017), Iavernaro et al. (2020), Iavernaro et al. (2021)]; game theory, paradoxes of infinity, probability, statistics, and random processes [see, e.g., Calude and Dumitrescu (2020), Fiaschi and Cococcioni (2018), Sergeyev (2022), Rizza (2018), Sergeyev (2023)]; high performance computing [see, e.g., Amodio et al. (2020); Sergeyev (2016)]; optimization problems under constraints [see, e.g., De Cosmis and De Leone (2012), De Leone (2018), Sergeyev and De Leone (2022); Žilinskas (2012)], teaching [see, e.g., Ingarozza et al. (2020), Mazzia (2022)] and handling ill-conditioning constraints in optimization problems [see, e.g., Gaudioso et al. (2018), Sergeyev et al. (2018)], etc. Additionally, in the recent papers (Falcone et al. 2020a, b, c), the *Software Solution to the Infinity Computer (SSIC)* has been proposed for managing the concepts delineated by the Infinity Computer within the MATLAB/Simulink tool. SSIC provides a user-friendly library that provides blocks to work with the Infinity Computer. In Falcone et al. (2020a), an additional module, named *Differentiation Blocks Module*, has been integrated in SSIC to perform standard, partial, and Lie higher-order derivative computations.

In this paper, the *IC_RK4* method proposed in Falcone et al. (2022) is further exploited to study complex Zeno hybrid systems. More specifically, two well-known Zeno hybrid systems, i.e., *Water tanks* [see Johansson et al. (1999)] and *Thermostat* [see Johnson et al. (2004)] are considered in this work. Both the systems are modeled with constant and nonlinear functions in order to stress the *IC_RK4* method and evaluate its performance in detecting zero-crossings through a dynamical generation of time observations by using the infinite quantity ① [see Sergeyev (2017) for its detailed description]. To show the validity of the *IC_RK4* method, these hybrid systems have been studied and results are gathered from simulations and compared with the standard method.

The rest of the paper is structured as follows. Section 2 provides an introduction to hybrid systems and the IC_RK4 method. Section 3 presents the *Water tanks* and *Thermostat* hybrid systems taken from the literature that exhibit Zeno behavior. Section 4 describes the conducted numerical experiments on such systems along with a comparison of simulation results obtained with the exploited *IC_RK4* method and a standard one. Conclusions are presented in Sect. 5. Finally, some additional results related to the conducted experiments are reported in Appendix.

2 Hybrid system models and the IC_RK4 method

The paper uses notions and concepts from hybrid systems and the IC_RK4 method along with the related algorithm and concepts, as presented in the following subsections.

2.1 Hybrid system models

The structure of a hybrid system model can be formally represented as [see Lunze and Lamnabhi-Lagarrigue (2009)] follows

$$H = (X, Q, f, Z, \delta, K, R, I), \tag{5}$$

where

$X = \mathbb{R}^n$ is the continuous state space;
Q is a finite set of discrete states;
f is a set of vector fields describing the continuous dynamics for all $q \in Q$;
Z is a set of initial states;
δ is the discrete state transition function;
K is a set of guards describing when a discrete state transition occurs;
R is a reset map defining the state jumps;
I invariants of the discrete states.

The discrete behavior is modeled through a control graph $G = (Q, E)$, where the set of vertices Q represents the discrete states (also called *"operation modes"*), whereas the edges $E \subseteq \{(q_i, q_j) | (q_i, q_j) \in Q^2 \wedge i, j = [0, 1, ..., n]\}$ represent state transitions managed by the function $\delta : Q \times \mathbb{R}^n \to Q$, which determines the discrete successor state q_j when the system is in the discrete state q_i. Every discrete state $q_i \in Q$ is linked to a vector field $f : Q \times \mathbb{R}^n \to \mathbb{R}^n$ that belongs to it. The evolution of the continuous state in the discrete state $q(t)$ is described by the equation $\dot{x}(t) = f(q(t), x(t))$.

The collection of initial states $Z \subset Q \times \mathbb{R}^n$ is defined within the hybrid system model. Through the associated vector field $f(q, \cdot)$, a discrete state q affects the continuous dynamics, while the collection of guards K serves as a representation of the impact of continuous dynamics on the evolution of discrete states. Thus, given a continuous state space X and $K \subset X$, a guard $k \in K$ is a region where a discrete state transition may take place if the state x is in k.

The reset map $R : Q \times Q \to 2^{\mathbb{R}^n} \times 2^{\mathbb{R}^n}$ and the invariants of the discrete states $I : Q \to 2^{\mathbb{R}^n}$ allow to include state jumps and complete the representation of the interaction between the continuous and the discrete dynamics. Specifically, each discrete state q has associated an invariant $i \in I$ that describes the conditions that the continuous state x has to satisfy at q. Invariants and guards play complementary roles; specifically, invariants characterize when a transition must take place, whereas guards represent "enabling conditions" that identify when a particular transition may take place. R is a function that specifies how new continuous states are related to previous ones for a given transition.

2.2 The IC_RK4 method and related modeling and simulation process

To design and simulate dynamic systems that exhibit the Zeno phenomenon, a modeling and simulation (M&S) process has been defined. This M&S process consists of a set of interconnected activities for the management of such systems. Starting from the description and requirements of a Zeno dynamic system (input), this M&S process allows one to produce an enhanced version of the system that can be simulated by managing the Zeno phenomenon (output). Figure 1 depicts the *Business Process Model and Notation (BPMN)* diagram of the defined process [see, e.g., Falcone et al. (2017), Falcone et al. (2018), Garro et al. (2018), von Rosing et al. (2015)].

The M&S process, of which the IC_RK4 method is part, involves two experienced engineers: *system engineers* and *modeling and simulation engineers*. System engineers are in charge of delineating the requirements and characteristics of the real system, in a semi-structured way, by using specialized software and technical documents to define the structure and behavior of the system along with its components, whereas modeling and simulation engineers, starting

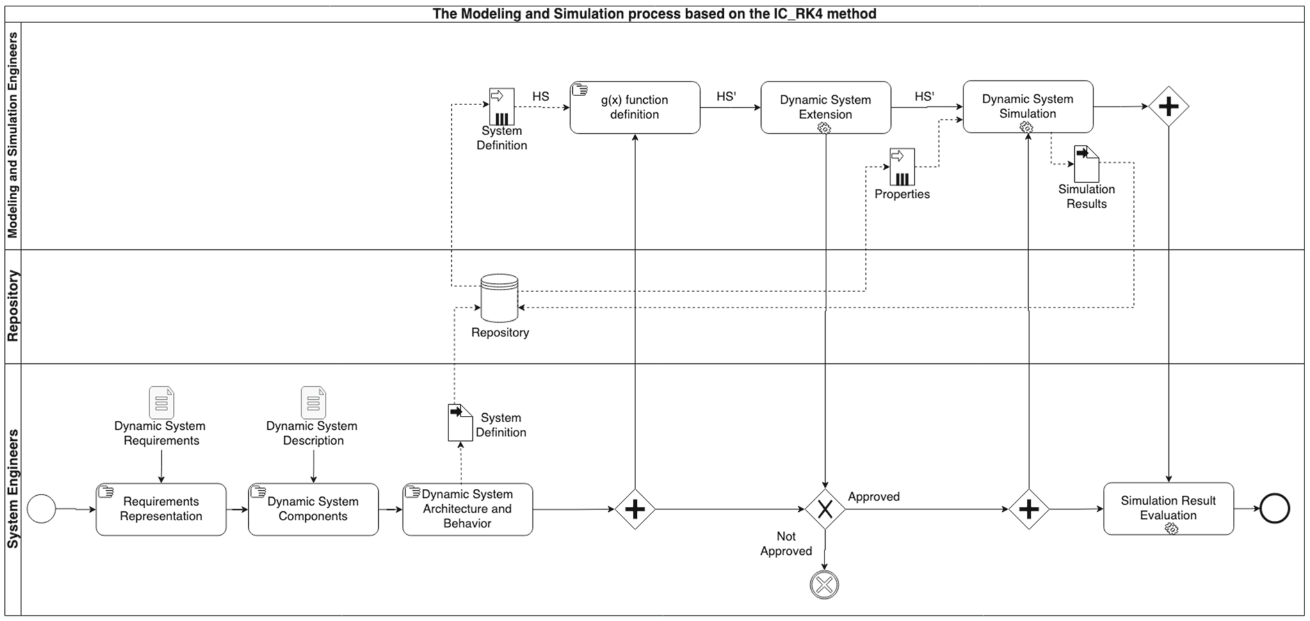

The Modeling and Simulation process based on the IC_RK4 method
Modeling and Simulation Engineers
Repository
System Engineers
HS
System Definition
g(x) function definition
HS'
Dynamic System Extension
HS'
Dynamic System Simulation
Properties
Simulation Results
Repository
Dynamic System Requirements
Dynamic System Description
System Definition
Requirements Representation
Dynamic System Components
Dynamic System Architecture and Behavior
Approved
Not Approved
Simulation Result Evaluation

Fig. 1 BPMN diagram of the modeling and simulation process of a dynamic system

from the technical reports and resources produced by system engineers, define the corresponding hybrid system automata, simulate it, and then gather the simulation results. The results coming from the performed experiments are jointly evaluated by both engineer experts to evaluate the system performance and explore the system's design before building it.

The M&S process begins inside the swimlane of the *system engineers*, where the real-world dynamic system is defined by capturing the system's requirements and definitions, and the stakeholders' needs in technical documents (*"Dynamic System Requirements"* and *"Dynamic System Description"*). This step ensures that the dynamic system's structure and behavior are proper, rational, and effective by making it explicit and complete in its components.

The obtained technical documents represent the input of the *"Requirements Representation"* and *"Dynamic System Components"* steps. In the first step, the system's requirements are formally defined to automate their verification through simulation, whereas the components that make up the system are specified in the *"Dynamic System Components"* step. After defining the hybrid system in terms of requirements and components, the whole system architecture and behavior are derived in the *"Dynamic System Architecture and Behavior"* step, and the system specifications are stored into a repository. At these steps, two languages are adopted to formalize the dynamic system: the *Unified Modeling Language (UML)* and/or *Systems Modeling Language (SysML)* (Johnson et al. 2007).

UML is a modeling language developed and maintained by the *Object Management Group (OMG)* that allows one to specify, design, implement, and document the artifacts of software systems. UML is not a programming language but it supports the entire life cycle of a software system in different application domains (e.g., finance, aerospace, and industry 4.0). UML provides visual notations, based on diagrams, that are not only intended for developers but also for stakeholders, and anybody involved in the project. The provided diagrams allow one to capture the characteristics of a system along with the relations among its components. SysML is a general-purpose architecture modeling language, maintained by OMG, for system engineering applications. The SysML language supports the definition, analysis, design, verification and validation of a wide range of systems and systems of systems. The SysML language allows system engineers to create effective SysML models able to capture the characteristics of a system at different levels of abstraction. SysML extends a subset of the UML diagrams by using the UML profile mechanism and provides additional diagrams for managing requirements and parametric constraints. As a result, SysML is more adaptable and expressive than UML for modeling dynamic systems.

Upon obtaining the formal representation of the hybrid system automata HS in terms of UML/SysML diagrams, control graph, and differential equations, the *"g(x) function definition"* step is executed by modeling and simulation engineers inside the corresponding swimlane. In this step, a $g(x)$ function that captures the discrete transitions of HS is defined and included in the hybrid system automata. This step produces as output an extended version of HS, i.e., HS' that is evaluated by system engineers for approval. If the design is approved, the *"Dynamic System Simulation"* step is performed by taking two input arguments: (i) a property file,

Algorithm 1: Simulation of a hybrid system automata

```
Data: The infinitesimal quantity $①^{-1}$;
      The time interval
      $[\Delta t_{min}; \Delta t_{max}] \mid 0 < \Delta t_{min} < \Delta t_{max} << T - t_0$;
      The simulation stop time $T$;
      The initial discrete state $q$.
1  $k \leftarrow 0$;
2  $t_k \leftarrow 0$;
3  $\Delta t \leftarrow \Delta t_{max}$;
4  while $t_k < T$ do
5      $\overline{x}_{k+1,①^{-1}} \leftarrow$ computeApproximation($x_{t_k+\Delta t-①^{-1}}$);
6      $\overline{g}_{k+1,①^{-1}} \leftarrow$ evaluateFunction($g(\overline{x}_{k+1,①^{-1}}, q)$);
7      $\overline{g}(y) \leftarrow$ extractCoefficients($\overline{g}_{k+1,①^{-1}}$);
8      $y^* \leftarrow$ findLargestValue($\overline{g}(y), \Delta t$);
9      if $\overline{g}(y) > 0$ at the whole interval $[0, \Delta t]$ then
10         $x_{k+1} \leftarrow$ finitePart($x_{k+1,①^{-1}}$);
11         $t_{k+1} \leftarrow t_k + \Delta t$;
12         $\Delta t = \Delta t_{max}$;
13     else
14         if $\overline{g}(y) < 0$ for all $t \in [0, \Delta t]$ or $y^* > \Delta t - \Delta t_{min}$ then
15             $y^* \leftarrow \Delta t - \Delta t_{min}$;
16         end
17         $x_{k+1} \leftarrow$ computeApproximation($x_{k+1}, y^*$);
18         $t_{k+1} \leftarrow t_k + (\Delta t - y^*)$;
19         $\Delta t = y^*$;
20         ACTION($varargs$);
21     end
22     $k \leftarrow k + 1$;
23 end
```

i.e., *"Properties"* that delineates the parameters required to configure the simulation scenario (e.g., simulation time step, solver type, log folder, etc.) and (ii) the extended hybrid system automata HS'. Inside this step, the simulation is performed according to Algorithm 1. Finally, in the *"Simulation Result Evaluation"* step, the simulation results are analyzed to study the system's behavior and evaluate different design alternatives before building it.

Algorithm 1 allows one to execute the simulation of a hybrid system automata HS' also in the presence of the Zeno behavior. The algorithm takes as parameters: (i) the infinitesimal quantity $①^{-1}$ provided by the Infinity Computer [see Sergeyev (2010)]; (ii) the time interval $[\Delta t_{min}; \Delta t_{max}]$, where Δt_{min} represents the smallest time value between two discrete transitions, and Δt_{max} is the time interval between two consecutive observations on "good" and "stable" regions; (iii) the simulation stop time T; and, finally, (iv) the initial discrete state of HS', q.

Until the simulation of HS' is completed, the approximation $\overline{x}_{k+1,①^{-1}}$ of the value $x(t)$ at the point $t = t_k + \Delta t - ①^{-1}$ with $\Delta t \in [\Delta t_{min}, \Delta t_{max}]$ is calculated by using the infinitesimal quantity $①^{-1}$ (see Algorithm 1-line 5). Note that x is an n-dimensional vector, whereas $①^{-1}$ is an infinitesimal scalar. Thus, all the arithmetic operations are component-wise vector operations, and they are performed numerically by adopting the Infinity Computer and not in a symbolic way [as it is done in standard methodologies, e.g., in Shamseddine and Berz (2000)].

Once the value $\overline{x}_{k+1,①^{-1}}$ is derived, the approximation $\overline{g}_{k+1,①^{-1}}$ is calculated, and then, the function $\overline{g}(y)$, $y \in [0, \Delta t]$, is defined using the coefficients of $①^{-1}$ extracted from $\overline{g}_{k+1,①^{-1}}$ (see Algorithm 1—lines 6, 7):

$$\overline{g}(y) = g_0 + g_1 \cdot y + g_2 \cdot y^2 + \cdots + g_N \cdot y^N. \tag{6}$$

After the search for zero-crossings of $\overline{g}(y)$ for $y \in [0, \Delta t]$, two situations can arise:

- *no zero-crossings* No zero-crossings have been detected, this means that the function $\overline{g}(y)$ is positive over the whole interval $[0, \Delta t]$. In this situation, the interval $[t_k, t_k + \Delta t]$ is considered "stable"; thus, it is no necessity to use a smaller stepsize Δt to better analyze this interval. The value x_{k+1} is obtained by using the function $FinitePart(t)$, which is provided by SSIC, at the point $t = x_{k+1,①^{-1}}$. Since the interval is "stable" for the next simulation step, the value of Δt is set to Δt_{max} (see Algorithm 1-lines 9–12);
- *zero-crossings detected* Zero-crossings have been detected, this means that the function $\overline{g}(y)$ can be negative in the interval $[0, \Delta t]$. To avoid numerical issues related to generation of observations too closely, the value of y^* is set to $\Delta t - \Delta t_{min}$ if $\overline{g}(y)$ is negative for all $t \in [0, \Delta t]$ or y^* is too close to Δt, i.e., $0 < \Delta t - y^* < \Delta t_{min}$. After that, the value x_{k+1} is calculated from the value $x_{k+1,①^{-1}}$ substituting $①^{-1}$ by y^* (without resolution of the ODEs and re-simulation of the system at $t = t_{k+1}$). Once the value x_{k+1} is obtained, the observation t_{k+1} is computed starting from $t_k + (\Delta t - y^*)$. Finally, the switch is performed by $ACTION(varargs)$ and the simulation continues decreasing $\Delta t = y^*$. More in detail, the function $ACTION(varargs)$ is responsible for performing a discrete transition on the hybrid automata and, if required, updating the values of its discrete variables (e.g., the hybrid automata's state). Here, the function's parameter $varargs$ holds all the information required to carry out the discrete transition (e.g., the current state q, the last computed variable x_k, and the previously calculated variables $x_{k-1}, x_{k-2}, \ldots$,). At the next iteration, if zero-crossings are not detected then the default value $\Delta t = \Delta t_{max}$ is automatically restored (see Algorithm 1-lines 14–20).

It is worth noting that the function $\overline{g}(y)$ can be negative for $y \in [0, \Delta t]$ but no zero-crossings are present in such interval. This situation may occur when a zero-crossing was not correctly determined during the previous iterations for different reasons. In this case, since by construction of $g(x, q)$ the system works correctly only when $g(x, q) \geq 0$, then the simulation can be incorrect at this interval and some additional actions are required. Since the value of Δt decreases with each iteration, the algorithm continues to simulate the hybrid systems with an ever smaller value of Δt up to the minimum allowed value of Δt_{min} (see Algorithm 1-line 19), this behavior allows to stabilize the system after some iterations, but, in general, more actions can be required to solve this issue.

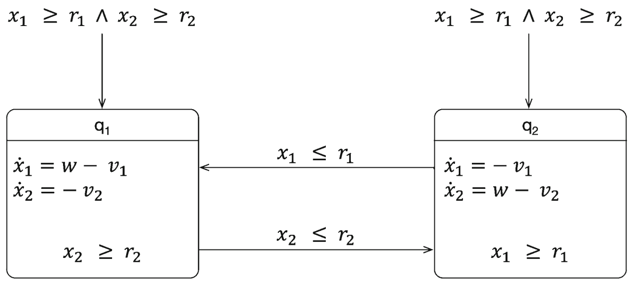


Fig. 2 Water tanks system hybrid automaton model

3 Hybrid systems used in the experiments

This section presents two well-known dynamic hybrid systems that exhibit the Zeno phenomenon, i.e., the *Water tanks* and *Thermostat* [see Johansson et al. (1999), Johnson et al. (2004)]. These systems have been considered in this work, since they are important both from practical and numerical points of view. More in detail, both the systems are *linear hybrid systems*, which means that the rate of change of each variable describing the system is constant, and the terms and conditions involved in the invariants, guards, and assignments are linear [see, e.g, Alur et al. (1995)].

Due to the Zeno phenomenon, the simulation of these systems fails because of numerical errors that start growing beyond the Zeno point. Such errors lead to incorrect simulation, and therefore, simulation results can be meaningless, especially when the systems are modeled with nonlinear functions.

3.1 Water tanks

Water tanks is an hybrid system that consists of two tanks t_1 and t_2 containing water [see Johansson et al. (1999)]. With reference to a generic Water tank t_i with $i \in I = \{1, 2\}$, define x_i as the water level, r_i as the critical threshold, $v_i > 0$ as the constant water flow going out. Define w as the constant flow of water that goes exclusively to either tank through a pipe at a given time t.

Figure 2 shows the hybrid automaton model of the Water tanks system. The two blocks represent the two discrete states, whereas the physical behavior is described with the following differential equations:

$$\begin{cases} \begin{cases} \dot{x_1} = w - v_1, \\ \dot{x_2} = -v_2, \end{cases} & \text{, if } x_2 \geq r_2, \\ \begin{cases} \dot{x_1} = -v_1, \\ \dot{x_2} = w - v_2. \end{cases} & \text{, if } x_1 \geq r_1, \end{cases} \tag{7}$$

The objective of the system is to keep, for the tank t_1 the water level x_1 above r_1, and for the tank t_2 the water level x_2 above r_2, assuming that $x_1(0) > r_1 \wedge x_2(0) > r_2$. To pursue this objective while the water levels x_1 and x_2 keep dropping, when in one of the tanks, for example t_1 the water level drops below the critical threshold, i.e., $x_1 < r_1$, the pipe switches to deliver the water, with the constant flow w, to the tank t_1. As the water rapidly goes out of the tanks, the switching frequency of the pipe increases until it reaches the limit point that occurs when for each tank t_i becomes $x_i = r_i$.

Let $x = x(t) \in \mathbb{R}^2$, $t \in [0, T]$, $q \in \{0, 1\}$, w, v_1, v_2 are positive constants, r_1 and r_2 are real constants in the following model:

$$\begin{cases} \begin{cases} \dot{x_1} = w - v_1, \\ \dot{x_2} = -v_2, \end{cases} & \text{, if } q = 0, \\ \begin{cases} \dot{x_1} = -v_1, \\ \dot{x_2} = w - v_2, \end{cases} & \text{, if } q = 1, \\ g(x) = (x_1 - r_1) \cdot q + (x_2 - r_2) \cdot (1 - q), \\ ACTION(q): \ q := 1 - q. \end{cases} \tag{8}$$

If $w = v_1 + v_2$, then the system should stabilize on the level $x_1 = r_1$ and $x_2 = r_2$, making infinitely many switches of the state starting from the time t_c. If $w < v_1 + v_2$, the system should also stabilize in one Water tank, but the level of the water in the other tank can decrease with time, making infinitely many switches as well. If $w > v_1 + v_2$, then the system will not stabilize, and the water level in the tanks will grow. (In the latter case, the system is not Zeno, since the intervals between the switches will also grow.)

The Water tanks hybrid system is also very important from a practical point of view. First, it has two different states $q \in \{0, 1\}$ and for each fixed state q, there are two independent systems, i.e., Water tanks. Second, this system involves Zeno behavior already from the first occurrence $g(x(t), q) = 0$. In this case, the errors related to the Zeno phenomenon can arise very quickly, leading so to wrong states, incorrect simulation, and/or large oscillations of the water around the desired level r ($r = r_1$ and r_2, respectively, for the first and the second tank). In the Water tanks system, the water level can be below the required level r. In this case, the simulation is correct only if the current state q leads to the growth of the water level in this Water tank. However, if the required water level r is not a constant, but a nonlinear function $r = r(t)$, then there could be more than one Zeno point and the system's behavior becomes more sophisticated. In this case, it can be inefficient to generate observations too dense already from the first occurrence of $g(x(t), q) = 0$, and as will be shown below, the IC_RK4 method can be extremely efficient from the computational point of view.

3.2 Thermostat

A Thermostat is a digital component that senses the temperature of a room and performs actions so that the room's temperature x is always maintained near a desired set point σ by turning "off" and "on" a heater.

When the Thermostat system starts, the heater is assumed to be "on" with an initial room temperature x, such that $x < \sigma$. In this situation, the room temperature increases according to the equation $\dot{x} = w - x$, where w is the heater constant. The heating phase continues until $x < \sigma$. Upon the temperature reaches σ, the Thermostat turns the heater "off," and then, the room's temperature starts decreasing according to the equation $\dot{x} = -x$.

Similar to the heating phase, the cooling one continues until $x \geq \sigma$. When the temperature value becomes less than σ, the Thermostat turns the heater "on" and starts heating again. Note that, even if the Thermostat system looks simpler than the Water tanks one, it allows to better appreciate specific features of IC_RK4 method as the presence of the single threshold σ makes the handling of the Zeno phenomenon even more challenger [see Johansson et al. (1999), Johnson et al. (2004)].

Figure 3 shows the hybrid automaton model of the Thermostat system. The two blocks represent the two discrete states, i.e., "off" and "on." In each discrete state, the room's temperature x evolves according to the following differential equation:

$$\begin{cases} \dot{x} = -x, & \text{if } x \geq \sigma, \\ \dot{x} = w - x, & \text{if } x < \sigma. \end{cases} \tag{9}$$

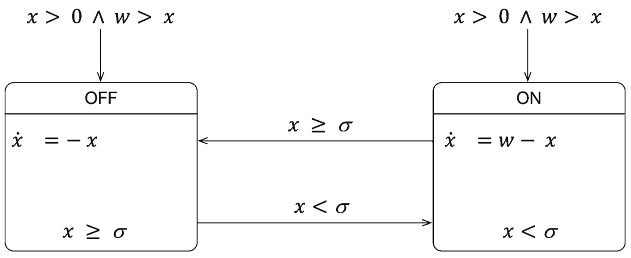


Fig. 3 Thermostat system hybrid automaton model

Let $x = x(t) \in \mathbb{R}^2$, $t \in [0, T]$, $q \in \{0, 1\}$, w be a nonnegative real constant large enough, whereas r be a real constant, $rand()$ generates a random number in the interval [0, 1]:

$$\begin{cases} \dot{x} = -x, \quad \text{if } q = 0, \\ \dot{x} = w - x, \quad \text{if } q = 1, \\ g(x) = (x - r + rand()) \cdot (1 - q) \\ \qquad + (r - x + rand()) \cdot q, \\ ACTION(q): \ q := 1 - q. \end{cases} \tag{10}$$

Formula (10) describes the hybrid dynamics of a Thermostat for monitoring the internal temperature of a building. During the simulation, two outputs are provided, i.e., x and q that represent the temperature and the operation mode, respectively. In each mode, a specific differential equation regulates the temperature. The initial mode is "on" with an initial temperature value w, and the transition conditions between the discrete state $q = 0$ ("off") and $q = 1$ ("on") are regulated by $ACTION(q)$.

The Thermostat system is also very important from the practical point of view, since it also has a clear physical real-world interpretation. Even if it is similar to the Water tanks system, it has several important differences with respect to it. First, it has only one variable $x(t)$, instead of two independent systems $x_1(t)$ and $x_2(t)$, as in the Water tanks. Second, the function $g(x, q)$ is randomized, so the zero-crossings are determined with a random error $\xi(t)$, which can be considered as a standard white noise and can add some difficulties both for the proposed and standard methods. However, as will be shown below, the proposed algorithm can be useful in this case as well, showing promising behavior with respect to standard methods.

4 Numerical experiments

4.1 Description of Numerical experiments

All numerical experiments have been executed in MATLAB version R2016b. A software simulator of the Infinity Computer has been used for this purpose [see Falcone et al. (2020b) for details]. The initial and final times t_0 and T have been taken to 0 and 20, respectively, for all test problems. All parameters for each test problem have been set to the presented values only for simplicity: The obtained results and conclusions with other initial values and system parameters are similar to the presented ones.

The parameters of the Infinity Computer have been set for all test problems as follows. The simplest infinitesimal ①$^{-1}$ has been used to set up the IC_RK4 method and the precision N has been set to 5, since the Runge–Kutta method of order 4 (RK4) method has the global error of order 4.

Table 1 Number of observations performed by the RK4_F and IC_RK4 methods to simulate the hybrid system with $r = const$. Specifically, for the Water tanks system $r_1 = r_2 = 1$, and for the Thermostat system $r = 70$

Case study	RK4_F		IC_RK4	
	Δt	Observations	Δt_{min}; Δt_{max}	Observations
Water tanks	0.005	4001	0.005; 0.5	3189
	0.0005	40,001	0.0005; 0.05	31,869
	0.00005	400,001	0.00005; 0.005	318,669
Thermostat	0.005	4001	0.005; 0.5	851
	0.0005	40,001	0.0005; 0.05	1266
	0.00005	400,001	0.00005; 0.005	5157

For finding the zero-crossing points of $\overline{g}(y)$ for $y \in [0, \Delta t]$, the standard bisection method up to machine precision has been used just for simplicity, since the variables $x(t)$ are always monotonically decreasing or increasing around the zero-crossings $g(x, q) = 0$ in the studied test problems. (Other methods for the determination of zero-crossings based on global optimization algorithms from Casado et al. (2002), Molinaro and Sergeyev (2001) and Sergeyev et al. (1999) on more difficult real-life problems can be exploited.)

Each of the hybrid systems presented above has the parameter(-s) r representing the simplest case for the zero-crossing function g: We want to keep the evaluations along the direction $x = r$ (for the Water tanks system, there are two parameters r_1 and r_2 of the same meaning: $x_1 = r_1$ should be kept, if the current state is 1, and $x_2 = r_2$, otherwise). However, in practice, the zero-crossing function g can have a nonlinear behavior: e.g., the cooling of the Thermostat along a nonlinear rule can be required. For this reason, for each test problem, we considered the following two series of experiments.

First, a simple case, when r is equal to a constant value ω, i.e., $r = \omega$ ($r_1 = \omega$ and $r_2 = \omega$ for Water tanks) is considered. Second, a more complicated case, when $r = r(t)$ ($r_1 = r_1(t)$ and $r_2 = r_2(t)$ for Water tanks), is considered, where $r(t)$, $r_1(t)$, $r_2(t)$ are nonlinear functions.

The already mentioned method RK4 has been used for solving ODEs. For each test problem, first, the standard RK4 method with the *fixed* stepsize Δt has been applied (we will call this method as *RK4_F*, hereinafter). Then, the *IC_RK4* method described above with the *dynamic* stepsizes and parameters Δt_{min} and Δt_{max} is applied for the same problems.

It should be noticed that in this paper, the internal MATLAB methods with dynamic stepsizes (e.g., Dormand–Prince methods) are not considered for several reasons. First, these methods are just related to different numerical algorithms for solving ODEs and not to the methods of generating observations t_k: These methods do not generate the observations dynamically, but use more sophisticated numerical methods for solving ODEs, with respect to the standard RK4 method, thus, comparing them with the RK4 method is not correct. Moreover, these methods are not adapted well for the case, when $g(x, q)$ is nonlinear, allowing only to determine simple zero-crossings of a type $x = 0$. However, it should be also noted that the algorithm for generating the observations proposed in this paper can be used with the above-mentioned dynamic methods, as well.

For each test problem, the standard method RK4_F with the fixed stepsize Δt has been applied for simulating the system using different values of Δt: 0.5, 0.05, 0.005, 0.0005, and 0.00005. Then, the IC_RK4 method has been applied for the same problems using three different values of the parameters Δt_{min} and Δt_{max}: $[\Delta t_{min}, \Delta t_{max}] = [0.005, 0.5]$, $[0.0005, 0.05]$, and $[0.00005, 0.005]$ in order to compare the obtained results with the standard method RK4_F with the stepsizes Δt of the same ranges. In the following subsections, only the figures related to the stepsizes $\Delta t = 0.5$, 0.05, and 0.005 for RK4_F, and $[\Delta t_{min}, \Delta t_{max}] = [0.5, 0.005]$ for IC_RK4 are presented; to improve the readability of the paper, the figures with the simulation results of other stepsizes are presented in Appendix (avoiding a lot of additional figures in the main text). This is done because these additional figures show the same behavior of the algorithms, but they substantiate the conclusions presented in this section, so they should be also attached to the paper. The numbers of generated observations by both methods are presented in each figure. They are also summarized in Tables 1, 2 for the sake of completeness.

4.2 Water tanks

4.2.1 Case $r_1 = r_2 = \omega$

In this section, the test problem studied numerically is the Water tanks system described in Sect. 3.1. The following parameters have been set for this system: $x(t_0) = [x_1(t_0), x_2(t_0)] = [2, 0]$, the initial state $q = 0$, and the parameters w, v_1, and v_2 have been chosen in the way that $w = v_1 + v_2$ (since it is required to keep the water levels x_1 and x_2 at the constant levels r_1 and r_2: $w = 5$, $v_1 = 2$, $v_2 = 3$). The desirable water levels r_1 and r_2 for both the

Table 2 Number of observations performed by the RK4_F and IC_RK4 methods to simulate the hybrid system with r set to a nonlinear function

Case study	RK4_F		IC_RK4	
	Δt	Observations	$\Delta t_{min};\Delta t_{max}$	Observations
Water tanks	0.005	4001	0.005; 0.5	282
	0.0005	40,001	0.0005; 0.05	1156
	0.00005	400,001	0.00005; 0.005	6387
Thermostat	0.005	4001	0.005; 0.5	978
	0.0005	40,001	0.0005; 0.05	1403
	0.00005	400,001	0.00005; 0.005	5314

Specifically, for the Water tanks system with $r_1 = sin(t) - t$ and $r_2 = sin(t)$, and for the Thermostat system $r(t) = \omega + sin(t) - t$ with $\omega = 70$

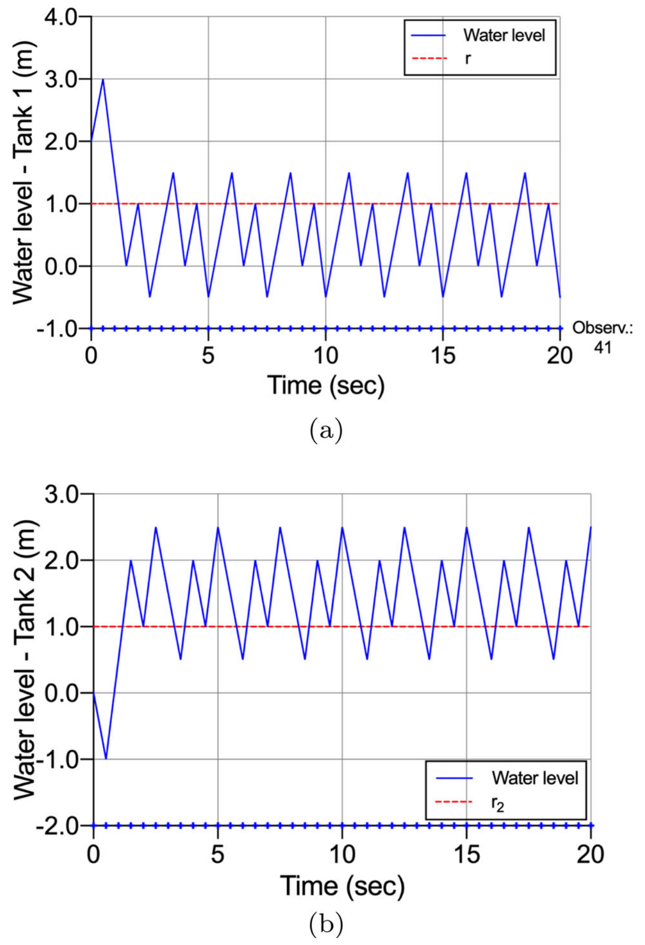


Fig. 4 Simulation of the Water tanks system performed with the standard method RK4_F with $\omega = 1$; thus, $r_1 = r_2 = \omega$ and the fixed stepsizes $\Delta t = 0.5$. Simulation results show that the standard method can lead to wrong states of the systems and its incorrect behavior, which can be resolved only using smaller Δt. Generated observations $t_k,\ k = 0, ..., N$, with $N = 40$) are dense everywhere (the observations are indicated below the graphs by signs "+")

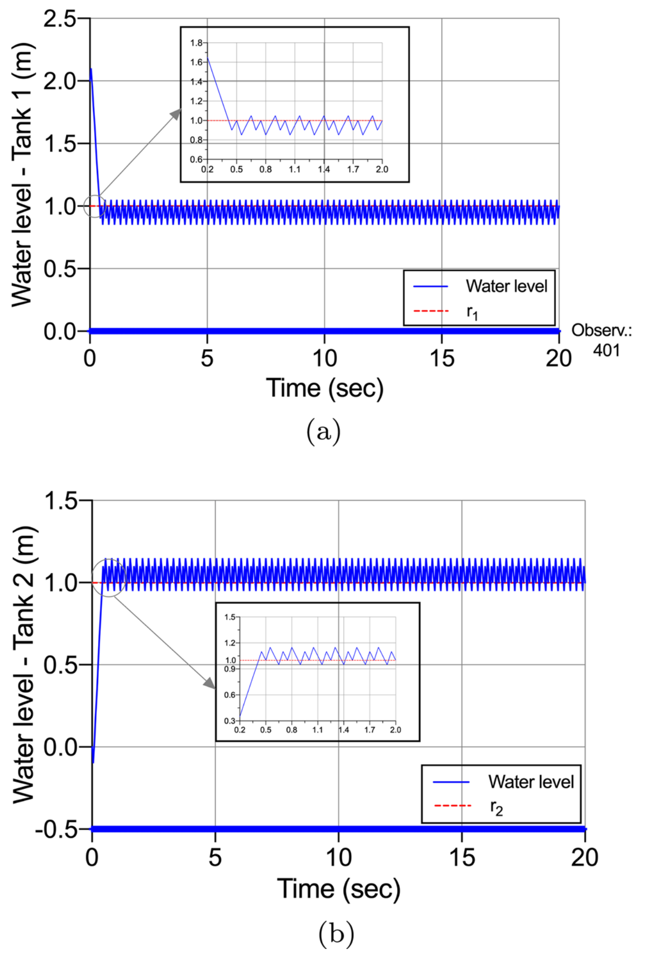
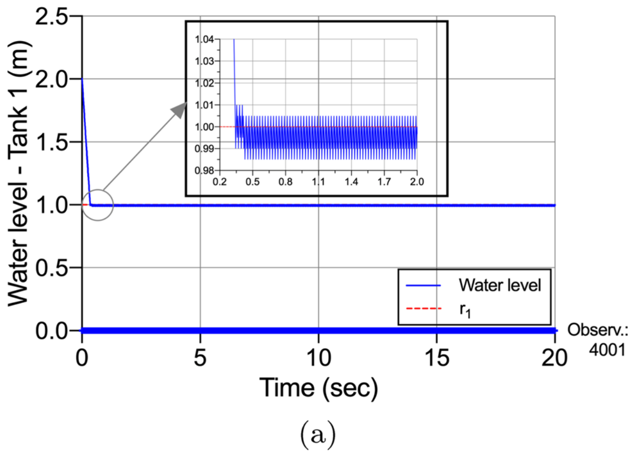


Fig. 5 Simulation of the Water tanks system performed with the standard method RK4_F with $\omega = 1$; thus, $r_1 = r_2 = \omega$ and the fixed stepsizes $\Delta t = 0.05$. Also in this case, the standard method RK4_F leads to incorrect system's behavior. Generated observations $t_k,\ k = 0, ..., N$, with $N = 400$ are dense everywhere (the observations are indicated by signs "+")

Water tanks have been set to a constant value $\omega = 1$, i.e., $r_1 = r_2 = 1$.

Results of the simulation by RK4_F using $\Delta t = 0.5$, 0.05, and 0.005 are presented in Figs. 4, 5, and 6, results of IC_RK4 using $[\Delta t_{min}, \Delta t_{max}] = [0.5, 0.005]$ are presented in Fig. 7. The remaining results are presented in Appendix (see Figs. 16, 17).



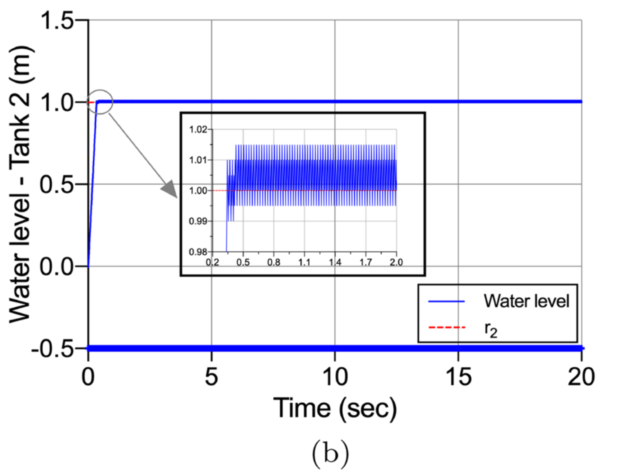
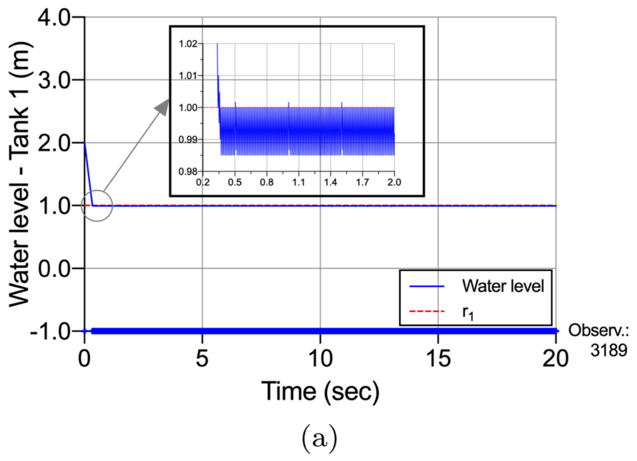


Fig. 6 Simulation of the Water tanks system performed with the standard method RK4_F with $\omega = 1$; thus, $r_1 = r_2 = \omega$ and the fixed stepsizes $\Delta t = 0.005$. The same considerations, which were described in both the cases $\Delta t = 0.5$ and $\Delta t = 0.05$ about the results of the standard method, can be done here. Generated observations t_k, $k = 0, ..., N$, with $N = 4000$ are dense everywhere (the observations are indicated by signs "+")



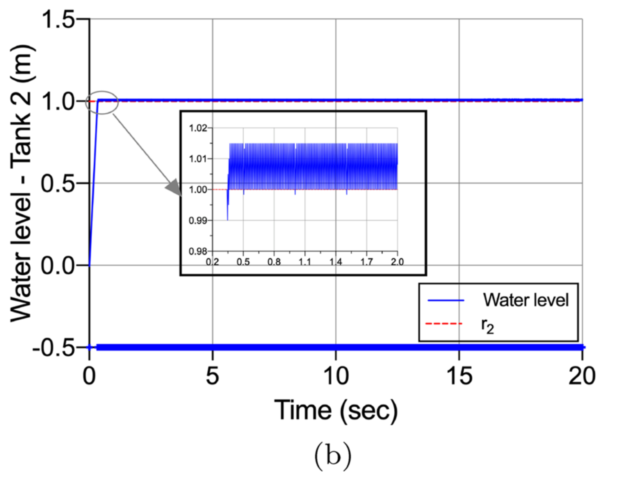
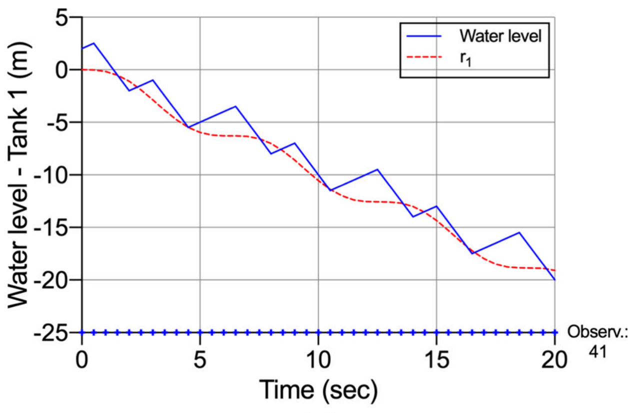


Fig. 7 Simulation of the Water tanks system performed with the evaluated method IC_RK4 with $\omega = 1$; thus, $r_1 = r_2 = \omega$ and $\Delta t_{min} = 0.005$, $\Delta t_{max} = 0.5$. The simulation results show that the proposed IC_RK4 method with the dynamic stepsizes Δt and automatic zero-crossing checking allows to keep the same precision as the fixed stepsize method with small Δt, but generates less observations. Generated observations t_k, $k = 0, 1, ..., 3188$, are dense only after the first zero-crossing (the observations are indicated below the graphs by signs "+")

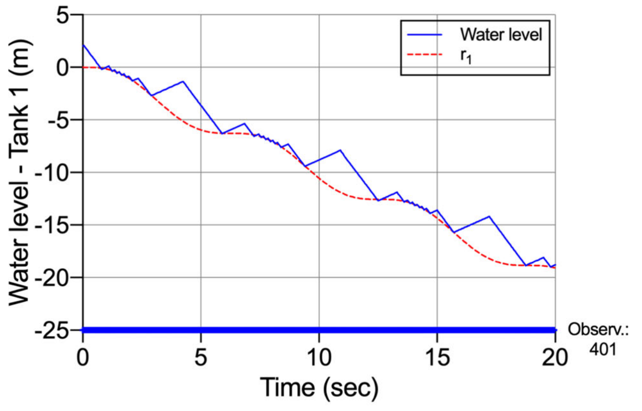
Figures 4, 5, and 6 show that the standard fixed stepsize method RK4_F also can lead to an incorrect system's state and, as a consequence, to incorrect behavior. In particular, Fig. 4 shows the incorrect behavior of the Water tanks system in the interval [0, 0.5]. More in detail, in *Tank 1* the water level decreases and remains below the desired value r_1, but it should increase to reach and stay on r_1 (see Fig. 4a). On the other tank, i.e., *Tank 2*, the water level increases above the desired value r_2, while it should decrease to reach and remain on r_2. However, with smaller values Δt, the method converges to the correct simulation (see Figs. 5, 6).

In its turn, the IC_RK4 method (see Fig. 7) simulates the system similarly to the standard method with the smallest value $\Delta t = 0.005$, switching right away the system from the "wrong" initial state to the correct one at the first iteration. Again, since the IC_RK4 method generates the observations t_k more frequently, where it is required, and less frequently on "good" and "stable" regions, then it generates fewer observations obtaining at least the same accuracy that the standard fixed stepsize method does. One can see also that the IC_RK4 method generates always fewer evaluations keeping at least no worse precision of the simulation. Both Table 1 and Fig. 7 show that the IC_RK4 method starts to generate the dense observations only after the first zero-crossing. This allows to use fewer computational resources on the initial subintervals where there are no zero-crossings and the system is well conditioned.

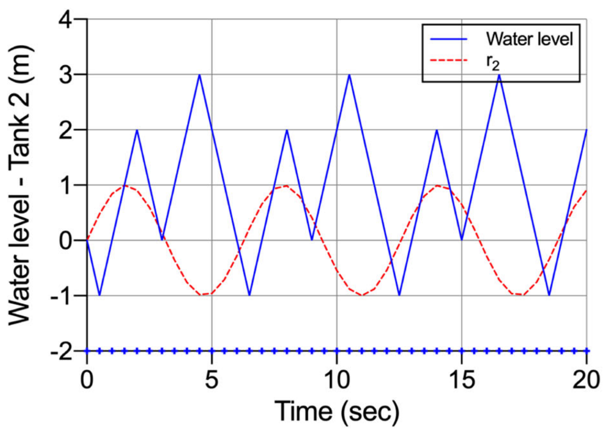
However, it should be noted that the IC_RK4 method does not resolve completely the Zeno phenomenon: It also can lead to incorrect behavior beyond the Zeno points. In Fig. 7,



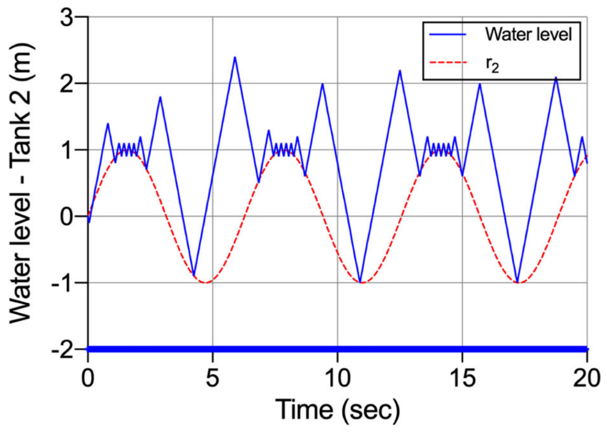
(a)



(a)



(b)



(b)

Fig. 8 Simulation of the Water tanks system performed with the standard method RK4_F with $r_1(t) = sin(t) - t$, $r_2(t) = sin(t)$, and fixed stepsizes $\Delta t = 0.5$. Simulation results show that the standard method leads to wrong states and incorrect system's behavior, which can be resolved only using smaller Δt. Generated observations t_k $k \in [0, ..., N]$, with $N = 40$ are dense everywhere (the observations are indicated below the graphs by signs "+")

Fig. 9 Simulation of the Water tanks system performed with the standard method RK4_F with $r_1(t) = sin(t) - t$, $r_2(t) = sin(t)$, and fixed stepsizes $\Delta t = 0.05$. Also in this case, the standard method leads to incorrect simulation of the system. Generated observations t_k, $k \in [0, ..., N]$, with $N = 400$ are dense everywhere (the observations are indicated with signs "+")

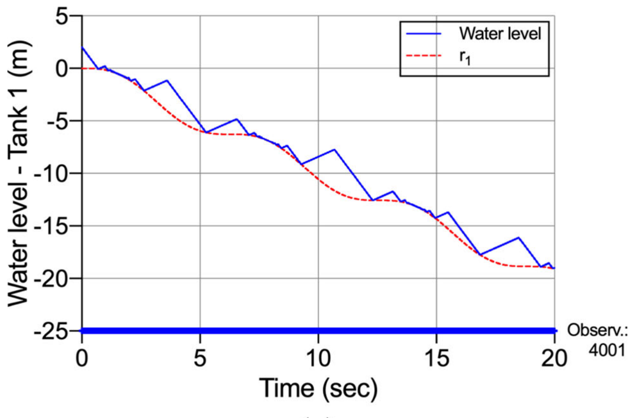
for example, the water level in the second Water tank can "jump" a little below the level r_2 at some moments t_k (at the same moments, the water level in the first tank "jumps" above the level r_1), this is related only to the Zeno behavior of the system and is due to Algorithm 1, lines 14–15: At some moments, the zero-crossing point y^* becomes too close to Δt, so it is replaced by $\Delta t - \Delta t_{min}$ to avoid numerical issues related to ill-conditioning, which leads to these "jumps."

4.2.2 Case $r_1(t) = sin(t) - t$ and $r_2(t) = sin(t)$

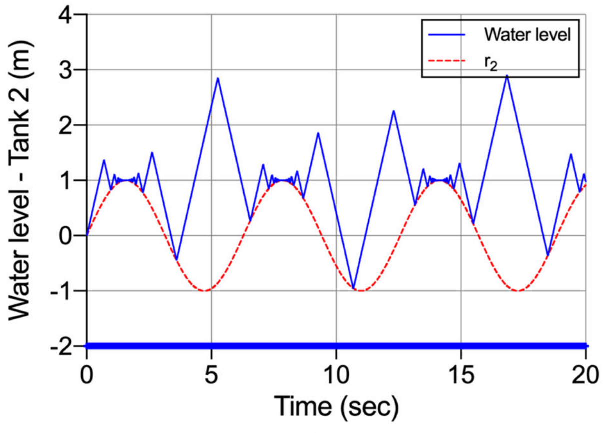
A numerical experiment has been conducted by changing the parameters r_1 and r_2 to nonlinear functions $r_1(t) = sin(t) - t$ and $r_2(t) = sin(t)$. The functions $r_1(t)$ and $r_2(t)$ have been chosen to study the case when the Water tank as should behave differently: We want to decrease the water level x_1 in the first tank along with the level $r_1(t)$, while we would like to keep the level x_2 in the second tank higher than $r_2(t)$. For this reason, the velocities of the flows w, v_1, and v_2 have been fixed in the way that $w < v_1 + v_2$.

Results of the simulation by RK4_F using $\Delta t = 0.5, 0.05$, and 0.005 are presented in Figs. 8, 9, and 10, and results of IC_RK4 using $[\Delta t_{min}, \Delta t_{max}] = [0.5, 0.005]$ are presented in Fig. 11. The remaining results are presented in Appendix (see Figs. 18, 19).

Also in this case, the standard method RK4_F was able to simulate the system behavior only using smallest values of Δt (see Figs. 8, 9, and 10), while the proposed one successfully simulated the system by generating much fewer observations than the standard one (see Fig. 11). One can see that the IC_RK4 method has generated more dense observations only in "unstable" and ill-conditioned regions. Table 2 reports the number of observations performed by the two methods to

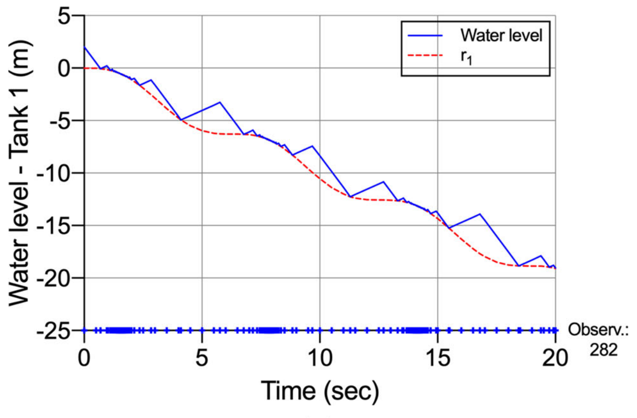


(a)

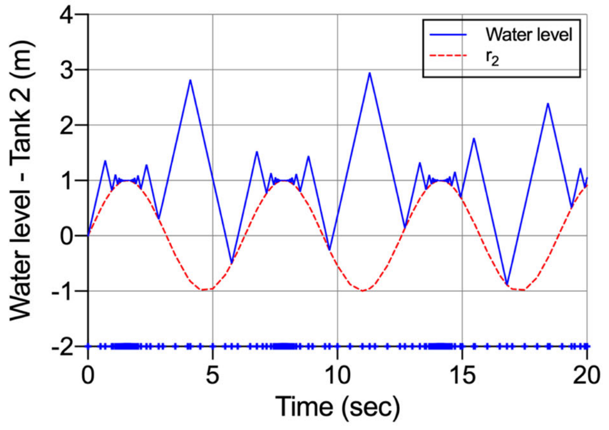


(b)

Fig. 10 Simulation of the Water tanks system performed with the standard method RK4_F with $r_1(t) = sin(t) - t$, $r_2(t) = sin(t)$, and fixed stepsizes $\Delta t = 0.005$. Same considerations, which were described in both the cases $\Delta t = 0.5$ and $\Delta t = 0.05$ about the adoption of the standard method to simulate the system. Generated observations t_k, $k \in [0, ..., N]$, with $N = 4000$ are dense everywhere (the observations are indicated with signs "+")



(a)



(b)

Fig. 11 Simulation of the Water tanks system performed with the evaluated method IC_RK4 with $r_1(t) = sin(t) - t$, $r_2(t) = sin(t)$, and $\Delta t_{min} = 0.005$, $\Delta t_{max} = 0.5$. The simulation results show that the IC_RK4 method with the dynamic stepsizes Δt and automatic zero-crossing checking allows to maintain the correct behavior of the system. Generated observations t_k, $k = 0, 1, ..., 281$, are more dense around zero-crossings (the observations are indicated below the graphs by signs "+")

simulate the system with $r_1(t) = sin(t) - t$ and $r_2(t) = sin(t)$.

Also in this case, the advantages of the IC_RK4 method become more evident in the case of a nonlinear function $g(x, q)$: It has generated almost 14, 34, and 62 times less observations than the standard one for $\Delta t = \Delta t_{min} = 0.005$, 0.0005, and 0.00005, respectively (282 vs 4001, 1156 vs 40,001, and 6387 vs 400,001 observations, respectively, see Table 2).

4.3 Thermostat

4.3.1 Case $r = \omega$

The example described in Sect. 3.2 is different with respect to the previous one due to the presence of random perturbations. It has been simulated using the following parameters: $x(t_0) = x_0 = 60$, initial state $q = 0$, $w = 100$, the desirable temperature r has been set to a constant value $\omega = 70\ ^oF$, random numbers have been generated on the interval [0, 1] using the internal MATLAB function $rand()$ (the seed of the generator has been set to 1 by the command $rng(1)$).

Results of the simulation by RK4_F using $\Delta t = 0.5, 0.05$, and 0.005 are presented in Fig. 12, results of IC_RK4 using $[\Delta t_{min}, \Delta t_{max}] = [0.5, 0.005]$ are presented in Fig. 13. The remaining results are presented in Appendix (see Figs. 20, 21).

Figures 12 and 13 show that the standard method RK4_F leads to an incorrect simulation of the system with biased mean temperature: it is required to keep the temperature of the Thermostat at the level of 70 ^oF, while one can see that

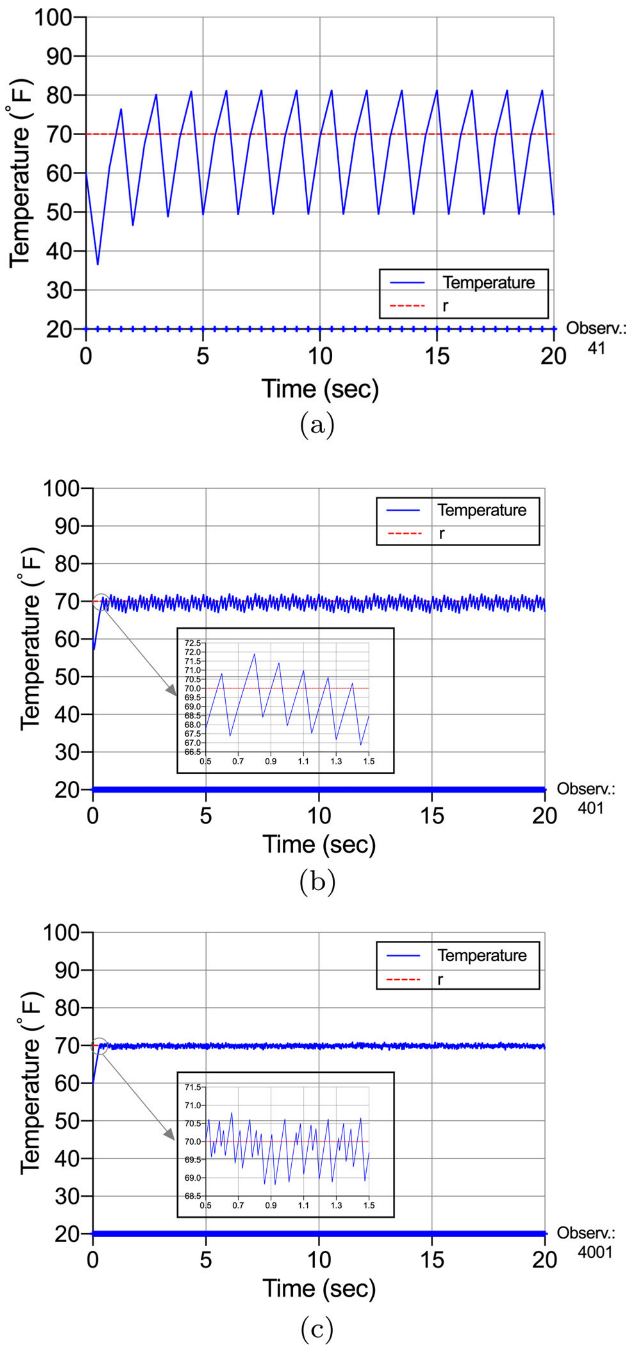


Fig. 12 Simulation of the Thermostat system performed with the standard method RK4_F with the constant value $\omega = 70$; thus, $r = \omega$ and $\Delta t = 0.5$ (**a**), $\Delta t = 0.05$ (**b**), and $\Delta t = 0.005$ (**c**). The simulation results show that the standard method RK4_F can lead to wide oscillations of the system's variables. These oscillations can be stabilized only using smaller Δt, and, therefore, can lead to biased average temperature with respect to the desirable level $r^o F$. Generated observations $t_k,\ k = 0, ..., N$, ($N = 40,\ 400,$ and 4000, respectively for **a**-**c**) are dense everywhere (the observations are indicated below the graphs by signs "+")

the mean level of the temperature generated by RK4_F is below this level (this is avoided only using smaller Δt, see

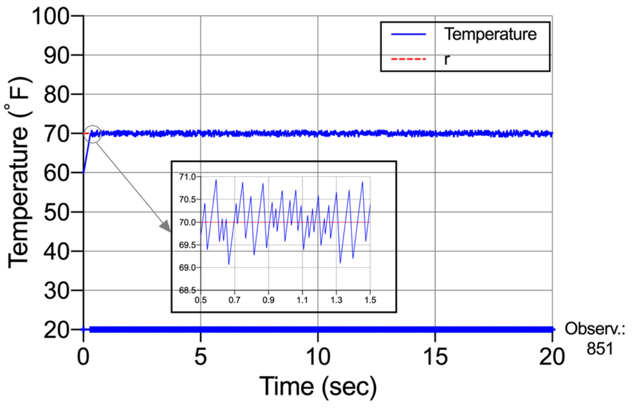


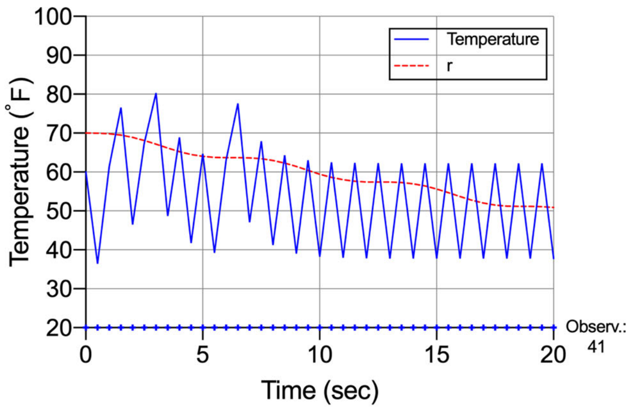
Fig. 13 Simulation of the Thermostat system performed with the evaluated method IC_RK4 with the constant value $\omega = 70$; thus, $r = \omega$ and $[\Delta t_{min},\ \Delta t_{max}] = [0.005,\ 0.5]$. Unlike the standard method RK4_F, the IC_RK4 one with the dynamic stepsizes Δt and automatic zero-crossing checking allows improving the accuracy of the simulation decreasing the oscillations and keeping them around the desirable level $r^o F$. Generated observations $t_k,\ k = 0, ..., 850$, are more dense after the first zero-crossing (the observations are indicated below the graphs by signs "+")

the respective figures in Appendix). The simulation behavior in Fig. 20 seems to be deterministic just because the value Δt, in this case, is too large and therefore dominates the stochastic perturbations. On the other hand, the IC_RK4 method simulates the system correctly generating the temperature of the desirable level on average (see Fig. 13). Table 1 shows that the proposed algorithm has generated significantly less observations than the standard method RK4_F: almost 5, 31, and 77 times less for $\Delta t = \Delta t_{min} = 0.005$, 0.0005, and 0.00005, respectively (851 vs 4001, 1266 vs 40,001, and 5157 vs 400,001 observations, respectively, see Table 1).

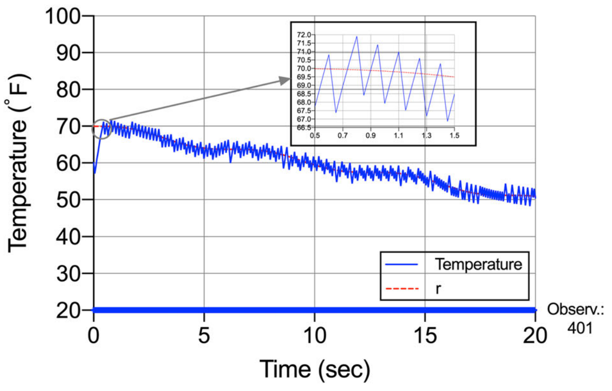
It should be noted that the IC_RK4 method determines the zero-crossings in this problem with a random error, since the function $g(x, q)$ contains random numbers. Thus, oscillations of the variable $x(t)$ with smaller Δt_{max} and Δt_{min} have higher amplitudes with respect to the standard method RK4_F (see Fig. 21). However, first, as for the standard method, smaller values of Δt_{min} and Δt_{max} allow reducing these amplitudes. Second, the proposed algorithm is able to maintain a correct simulation in this case as well as for the previous hybrid systems, decreasing the number of observations significantly: e.g., as was mentioned above it has generated almost 77 times fewer observations using $[\Delta t_{min}, \Delta t_{max}] = [0.00005, 0.005]$ than the standard method using $\Delta t = \Delta t_{min} = 0.00005$.

4.3.2 Case $r = \omega + sin(t) - t$

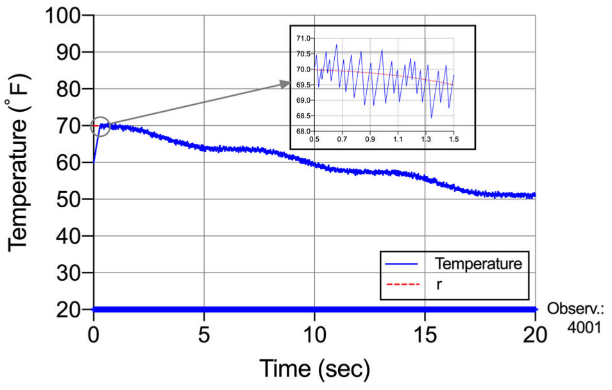
A further numerical experiment has been conducted by changing the parameter $r = \omega$ to $r(t) = \omega + sin(t) - t$ with $\omega = 70$ similarly to the previous experiments. From the



(a)



(b)



(c)

Fig. 14 Simulation of the Thermostat system performed with the standard method RK4_F with the constant value $\omega = 70$; thus, $r(t) = \omega + sin(t) - t$ and $\Delta t = 0.5$ (**a**), $\Delta t = 0.05$ (**b**), $\Delta t = 0.005$ (**c**). The simulation results show that the standard method RK4_F can lead to wide oscillations of the system's variables, is able to stabilize them only using smaller values of Δt, and can lead to biased cooling. Generated observations $t_k, \; k \in [0, ..., N]$, with $N = 40, \; 400$, and 4000, respectively for **a**–**c**) are dense everywhere (the observations are indicated below the graphs by signs "+")

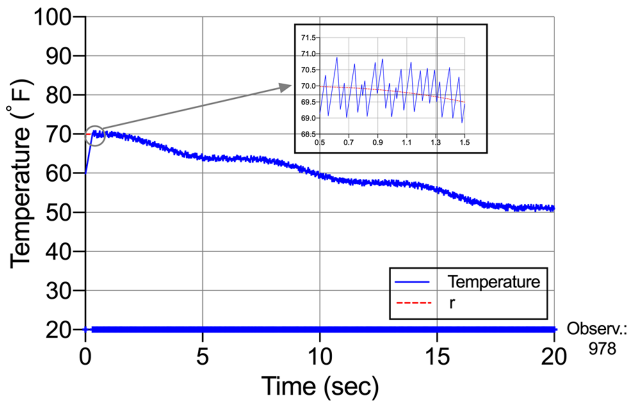


Fig. 15 Simulation of the Thermostat system performed with the evaluated method IC_RK4 with the constant value $\omega = 70$; thus, $r(t) = \omega + sin(t) - t$ and $\Delta t_{min} = 0.005, \; \Delta t_{max} = 0.5$. Unlike the standard method RK4_F, the IC_RK4 method with the dynamic stepsizes Δt and automatic zero-crossing checking allows to improve the accuracy of the simulation decreasing the oscillations and keeping the desirable cooling. Generated observations $t_k, \; k = 0, 1, ..., 977$, are dense after the first zero-crossing (the observations are indicated below the graphs by signs "+")

physical point of view, this case can be described as follows. At the time $t = 0$, the required temperature is $\omega \, ^oF$. After that, a cooling along with a nonlinear rule $r(t)$ is required (for example, a too fast or slow cooling can damage the simulated devices).

The results of the simulation by RK4_F using $\Delta t = 0.5$, 0.05, and 0.005 are presented in Fig. 14, the results of IC_RK4 using $[\Delta t_{min}, \Delta t_{max}] = [0.5, 0.005]$ are presented in Fig. 15. The remaining results are presented in Appendix (see Figs. 22, 23).

In this case, the simulation results confirm that the standard method was able to simulate the system behavior correctly only using the smallest values of Δt (see Fig. 14), while the IC_RK4 method successfully simulated it by generating significantly fewer observations (see Fig. 15).

The obtained results confirm the advantages of the IC_RK4 method: it is able to generate fewer observations with respect to the standard one simulating correctly the desirable cooling. In particular, the IC_RK4 method has generated almost 4, 29, and 75 times less observations for $\Delta t = \Delta t_{min} = 0.005$, 0.0005, and 0.00005, respectively (978 vs 4001, 1403 vs 40,001, and 5314 vs 400,001 observations, respectively, see Table 2). However, again, for the same reasons as in the previous case, i.e., due to random errors in the determination of zero-crossings, the IC_RK4 method simulates the system with a higher amplitude with small Δt_{min} and Δt_{max}, than the standard method, maintaining always a correct behavior of the system and generating a significantly smaller number of observations with respect to the standard method.

5 Conclusion

To support the design, development, and operation of modern Cyber-Physical Systems, many research efforts are focusing on the definition of methods, models, and techniques capable of capturing the interactions between the physical and cyber components through the definition of hybrid system models. Unfortunately, one of the most important issues in hybrid systems is the *Zeno phenomenon* consisting in the identification of zero-crossings represents a crucial aspect to adequately simulate these systems. The numerical methods adopted by classical modeling and simulation techniques are slow because they require solving the ordinary differential equations at each step. To overcome this issue, and, therefore, avoid redundant computations, the *IC_RK4* method working numerically with finite, infinite, and infinitesimal on the Infinity Computer has been used.

The *IC_RK4* method has been exploited to study two well-known Zeno hybrid systems, i.e., *Water tanks* and *Thermostat*. Both the systems have been modeled with constant and nonlinear threshold functions to stress the *IC_RK4* method, and evaluate its performance in detecting zero-crossings by generating time observations using the infinite quantity ① offered by the Infinity Computer, dynamically.

Simulation results have confirmed the validity and performance of the *IC_RK4* method in simulating Zeno hybrid systems. Moreover, it has been shown in the performed experiments that the *IC_RK4* method allows improving the accuracy of the simulation, also in the presence of nonlinear threshold functions, because the effective detection of zero-crossings directly influences the efficiency of the simulation.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability All data supporting the results reported in the article are present in the paper.

Declarations

Conflict of interest All authors declare that they have no conflict of interest.

Human and animal rights. This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix

A Water tanks system

See Appendix Figs. 16, 17, 18, and 19.

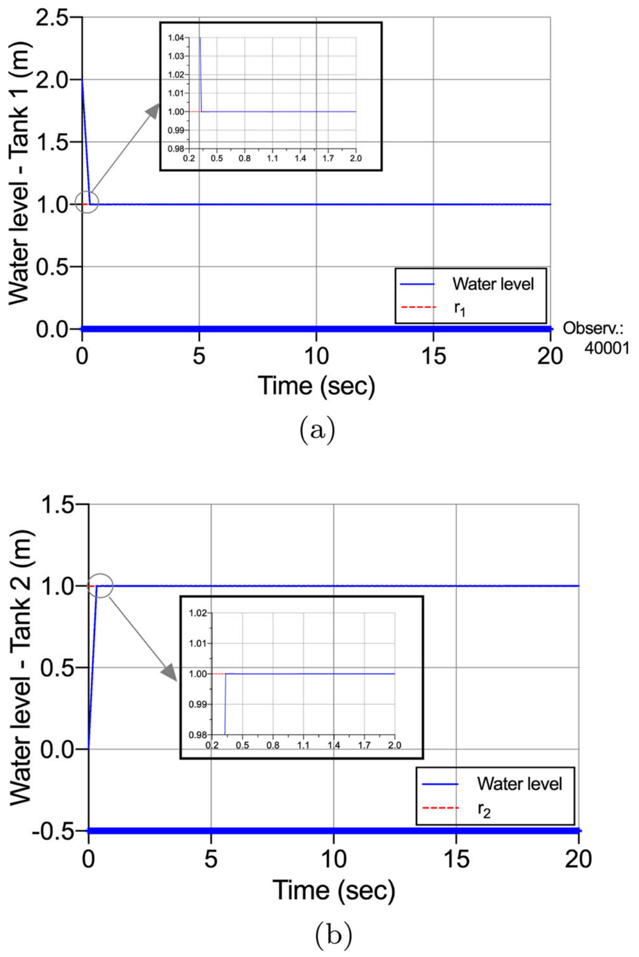
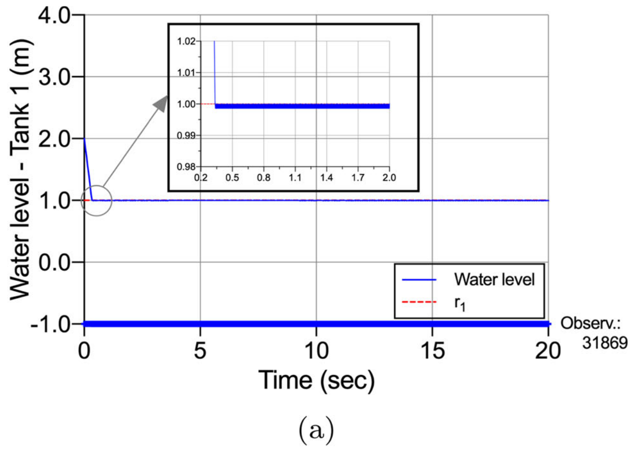


Fig. 16 Simulation of the Water tanks system performed with the standard method $RK4_F$ with the constant value $\omega = 1$; thus, $r1 = r2 = \omega$ and the fixed stepsizes $\Delta t = 0.00005$. Trend of the water level related to Tank 1(a), Tank 2(b)



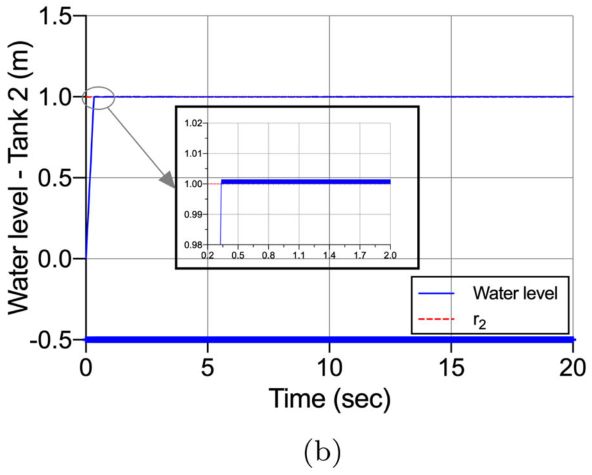
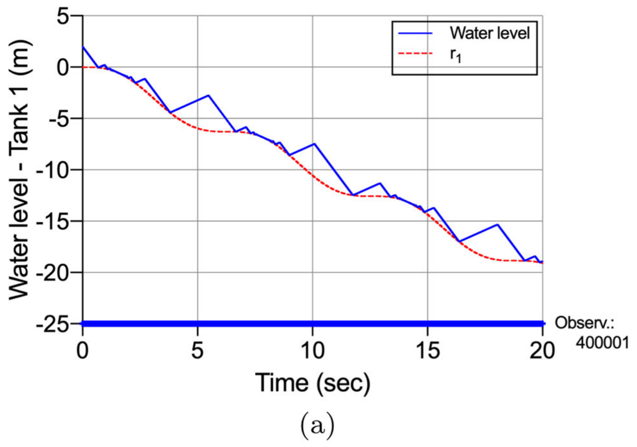


Fig. 17 Simulation of the Water tanks system performed with the evaluated method IC_{RK4} with the constant value $\omega = 1$; thus, $r1 = r2 = \omega$ and $[\Delta tmin, \Delta tmax] = [0.0005, 0.05]$. Trend of the water level related to Tank 1(a), Tank 2(b)



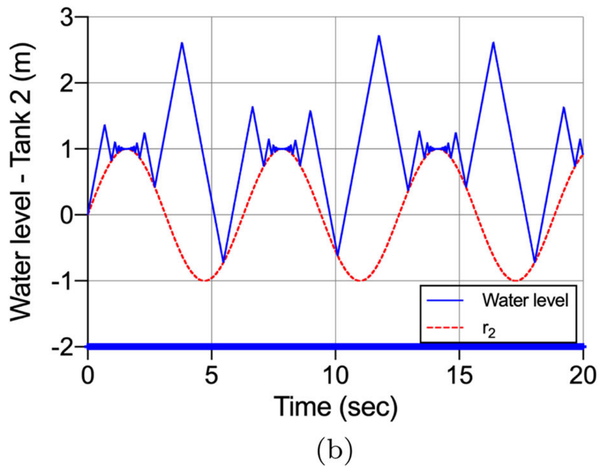
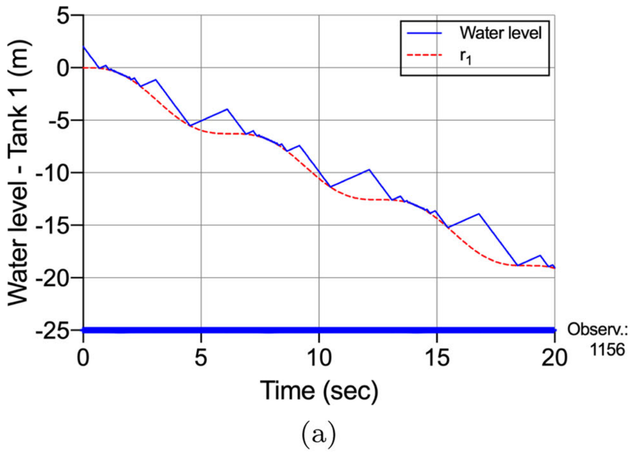


Fig. 18 Simulation of the Water tanks system performed with the standard method $RK4_F$ with $r1 = \sin(t) - t, r2 = \sin(t)$ and fixed stepsizes $\Delta t = 0.00005$. Trend of the water level related to Tank 1(a), Tank 2(b)



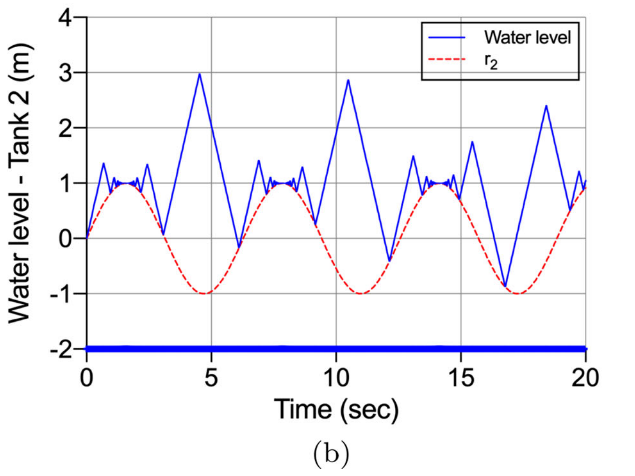


Fig. 19 Simulation of the Water tanks system performed with the evaluated method IC_{RK4} with $r1 = \sin(t) - t$ and $r2 = \sin(t)$ and $[\Delta tmin, \Delta tmax] = [0.0005, 0.05]$. Trend of the water level related to Tank 1(a), Tank 2(b)

Thermostat system

See Appendix Figs. 20, 21, 22, and 23.

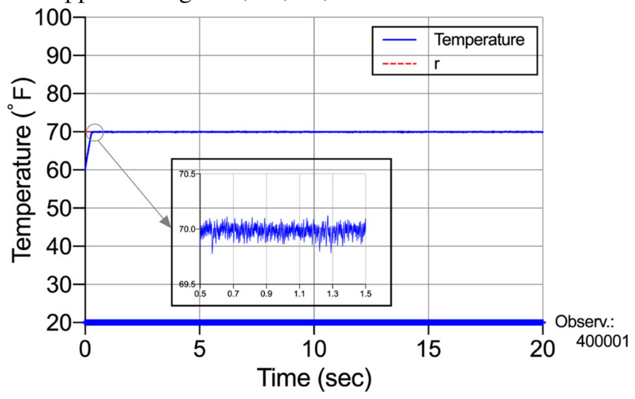


Fig. 20 Simulation of the Thermostat system performed with the standard method RK4_F with the constant value $\omega = 70$; thus, $r = \omega$ and the fixed stepsizes $\Delta t = 0.00005$

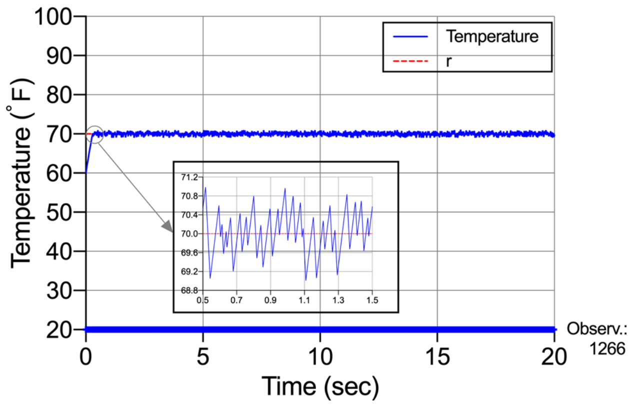


Fig. 21 Simulation of the Thermostat system performed with the evaluated method IC_RK4 with the constant value $\omega = 70$; thus, $r = \omega$ and $[\Delta t_{min}, \Delta t_{max}] = [0.0005, 0.05]$

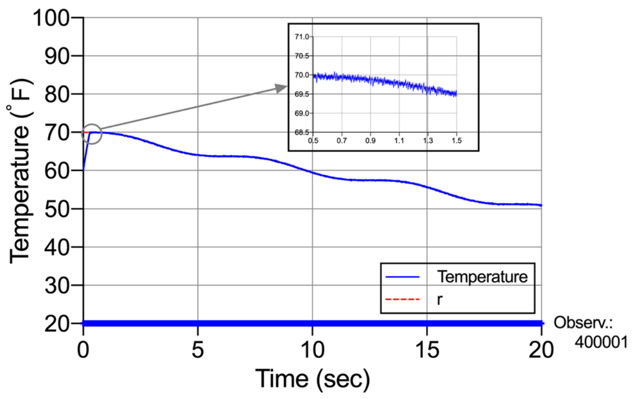


Fig. 22 Simulation of the Thermostat system performed with the standard method RK4_F with the constant value $\omega = 70$; thus, $r = \omega + sin(t) - t$ and the fixed stepsizes $\Delta t = 0.00005$

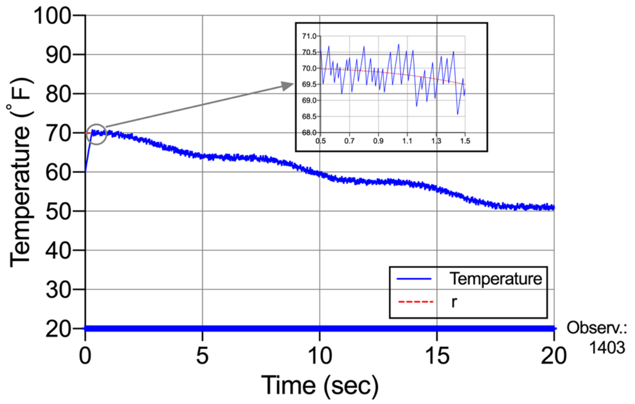


Fig. 23 Simulation of the Thermostat system performed with the evaluated method IC_RK4 with the constant value $\omega = 70$; thus, $r = \omega + sin(t) - t$ and $[\Delta t_{min}, \Delta t_{max}] = [0.0005, 0.05]$

References

Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, Nicollin X, Olivero A, Sifakis J, Yovine S (1995) The algorithmic analysis of hybrid systems. Theoret Comput Sci 138(1):3–34

Ames AD, Sastry SS (2004) Blowing up affine hybrid systems. In: 2004 43rd IEEE conference on decision and control (CDC) (IEEE Cat. No. 04CH37601), vol 1. IEEE, pp 473–478

Ames AD, Zheng H, Gregg RD, Sastry SS (2006) Is there life after Zeno? Taking executions past the breaking (Zeno) point. In: 2006 American control conference. IEEE, 6 pp

Amodio P, Iavernaro F, Mazzia F, Mukhametzhanov MS, Sergeyev YD (2017) A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic. Math Comput Simul 141:24–39

Amodio P, Brugnano L, Iavernaro F, Mazzia F (2020) On the use of the Infinity Computer architecture to set up a dynamic precision floating-point arithmetic. Soft Comput 24(23):17589–17600

Biák M, Hanus T, Janovská D (2013) Some applications of Filippov's dynamical systems. J Comput Appl Math 254:132–143

Bocciarelli P, D'Ambrogio A, Falcone A, Garro A, Giglio A (2019) A model-driven approach to enable the simulation of complex systems on distributed architectures. Simul: Trans Soc Model Simul Int **95**(12), 1185–1211

Bouskela D, Falcone A, Garro A, Jardin A, Otter M, Thuy N, Tundis A (2021) Formal requirements modeling for cyber-physical systems engineering: an integrated solution based on form-l and modelica. Requir Eng 27(1):1–30

Branicky MS (2005) Introduction to hybrid systems. In: Handbook of networked and embedded control systems. Springer, pp 91–116

Calude CS, Dumitrescu M (2020) Infinitesimal probabilities based on Grossone. SN Comput Sci 1:1–8

Casado LG, García I, Sergeyev YD (2002) Interval algorithms for finding the minimal root in a set of multiextremal one-dimensional nondifferentiable functions. SIAM J Sci Comput 24(2):359–376

De Cosmis S, De Leone R (2012) The use of Grossone in mathematical programming and operations research. Appl Math Comput 218(16):8029–8038

De Leone R (2018) Nonlinear programming and grossone: quadratic programming and the role of constraint qualifications. Appl Math Comput 318:290–297

Falcone A, Garro A (2019) Distributed co-simulation of complex engineered systems by combining the high level architecture and functional mock-up interface. Simul Model Pract Theory 97(August):101967

Falcone A, Garro A, Mukhametzhanov MS, Sergeyev YD (2020a) A Simulink-based software solution using the Infinity computer methodology for higher order differentiation. Appl Math Comput 409:125606

Falcone A, Garro A, Mukhametzhanov MS, Sergeyev YD (2020b) Representation of Grossone-based arithmetic in simulink for scientific computing. Soft Comput 24(23):17525–17539

Falcone A, Garro A, Mukhametzhanov MS, Sergeyev YD (2020c) A simulink-based infinity computer simulator and some applications. In: 3rd international conference and summer school 'numerical computations: theory and algorithms', NUMTA 2019, Le Castella, Crotone, Italy, June 15–21, 2019. Springer Nature, Switzerland, pp 362–369

Falcone A, Garro A, Mukhametzhanov MS, Sergeyev YD (2022) Simulation of hybrid systems under Zeno behavior using numerical infinitesimals. Commun Nonlinear Sci Numer Simul 111:106443

Falcone A, Garro A (2020) Pitfalls and remedies in modeling and simulation of cyber physical systems. In: 24th IEEE/ACM international symposium on distributed simulation and real time applications, DS-RT 2020, Prague, Czech Republic, September 14–16, 2020. IEEE, pp 1–5

Falcone A, Garro A, D'Ambrogio A, Giglio A (2017) Engineering systems by combining BPMN and HLA-based distributed simulation. In: The 2017 IEEE international conference on systems engineering symposium, ISSE 2017, Vienna, Austria, October 11–13, 2017. IEEE, pp 1–6

Falcone A, Garro A, D'Ambrogio A, Giglio A (2018) Using BPMN and HLA for engineering SoS : lessons learned and future directions. In: the 2018 IEEE international conference on systems engineering symposium, ISSE 2018, Rome, Italy, October 1–3, 2018. IEEE, pp 1–8

Fiaschi L, Cococcioni M (2018) Numerical asymptotic results in game theory using Sergeyev's Infinity Computing. Int J Unconv Comput 14(1):1–25

Filippov AF (2013) Differential equations with discontinuous righthand sides: control systems, vol 18. Springer

Fritzson P (2014) Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach. Wiley

Garro A, Falcone A, D'Ambrogio A, Giglio A (2018) A model-driven method to enable the distributed simulation of BPMN models. In: The 27th IEEE international conference on enabling technologies: infrastructure for collaborative enterprises, WETICE 2018, Paris, France, June 27–29, 2018. IEEE, pp 121–126

Gaudioso M, Giallombardo G, Mukhametzhanov MS (2018) Numerical infinitesimals in a variable metric method for convex nonsmooth optimization. Appl Math Comput 318:312–320

Grossman RL, Nerode A, Ravn AP, Rischel H (1993) Hybrid systems, vol 736. Springer

Heymann M, Lin F, Meyer G, Resmerita S (2005) Analysis of Zeno behaviors in a class of hybrid systems. IEEE Trans Autom Control 50(3):376–383

Iavernaro F, Mazzia F, Mukhametzhanov MS, Sergeyev YD (2020) Conjugate-symplecticity properties of Euler-Maclaurin methods and their implementation on the infinity computer. Appl Numer Math 155:58–72

Iavernaro F, Mazzia F, Mukhametzhanov MS, Sergeyev YD (2021) Computation of higher order lie derivatives on the infinity computer. J Comput Appl Math 383:113135

Ingarozza F, Adamo MT, Martino M, Piscitelli A (2020) A Grossone-based numerical model for computations with Infinity: A case study in an Italian high school, Lecture Notes in Computer Science, LNCS 11973, 451–462

Johansson KH, Egerstedt M, Lygeros J, Sastry SS (1999) On the regularization of Zeno hybrid automata. Syst Control Lett 38(3):141–150

Johnson KH, Lygeros J, Sastry S (2004) Modeling of hybrid systems. In: Control systems, robotics and automation, vol XV

Johnson TA, Jobe JM, Paredis CJ, Burkhart R (2007) Modeling continuous system dynamics in SysML. In: ASME international mechanical engineering congress and exposition, vol 42975, pp 197–205

Kimura T (2009) On Dormand–Prince method. Jpn Malays Tech Inst 40:1–9

Lunze J, Lamnabhi-Lagarrigue F (2009) Handbook of hybrid systems control: theory, tools, applications. Cambridge University Press

Lygeros J, Johansson KH, Simic SN, Zhang J, Sastry SS (2003) Dynamical properties of hybrid automata. IEEE Trans Autom Control 48(1):2–17

Mazzia F (2022) A computational point of view on teaching derivatives. Inform pduc 37(1):79–86

Molinaro A, Sergeyev YD (2001) An efficient algorithm for the zero-crossing detection in digitized measurement signal. Measurement 30(3):187–196

Platzer A (2008) Differential dynamic logic for hybrid systems. J Autom Reason 41(2):143–189

Rizza D (2018) A Study of mathematical determination through Bertrand's Paradox. Philosophia Math 26(3):375–395

Sergeyev YD (2023) Lower and upper estimates of the quantity of algebraic numbers. Mediterr J Math 20(1):12

Sergeyev YD. Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them. US patent 7,860,914 (2010), EU patent 1,728,149 (2009), RF patent 2,395,111 (2010)

Sergeyev YD, De Leone R (eds) (2022) Numerical infinities and infinitesimals in optimization. Springer

Sergeyev YD (2016) The exact (up to infinitesimals) infinite perimeter of the Koch snowflake and its finite area. Commun Nonlinear Sci Numer Simul 31(1–3):21–29

Sergeyev YD (2017) Numerical infinities and infinitesimals: methodology, applications, and repercussions on two Hilbert problems. EMS Surv Math Sci 4:219–320

Sergeyev YD (2019) Independence of the Grossone-based infinity methodology from non-standard analysis and comments upon logical fallacies in some texts asserting the opposite. Found Sci 24(1):153–170

Sergeyev YD (2022) Some paradoxes of infinity revisited. Mediterr J Math 19(3):1–28

Sergeyev YD, Daponte P, Grimaldi D, Molinaro A (1999) Two methods for solving optimization problems arising in electronic measurements and electrical engineering. SIAM J Optim 10(1):1–21

Sergeyev YD, Kvasov DE, Mukhametzhanov MS (2018) On strong homogeneity of a class of global optimization algorithms working with infinite and infinitesimal scales. Commun Nonlinear Sci Numer Simul 59:319–330

Shamseddine K, Berz M (2000) The differential algebraic structure of the Levi–Civita field and applications. Int J Appl Math 3:449–464.

Utkin VI (2013) Sliding modes in control and optimization. Springer

von Rosing M, White S, Cummins F, de Man H (2015) Business process model and notation-BPMN

Weiss D, Küpper T, Hosham HA (2015) Invariant manifolds for nonsmooth systems with sliding mode. Math Comput Simul 110:15–32

Yu L, Barbot JP, Benmerzouk D, Boutat D, Floquet T, Zheng G (2011) Discussion about sliding mode algorithms, Zeno phenomena and observability. In: Sliding modes after the first decade of the 21st century. Springer, pp 199–219

Zhang J, Johansson KH, Lygeros J, Sastry SS (2001) Zeno hybrid systems. Int J Robust Nonlinear Control: IFAC-Aff J 11(5):435–451

Zhang J, Johansson KH, Lygeros J, Sastry SS (2000) Dynamical systems revisited: hybrid systems with Zeno executions. In: International workshop on hybrid systems: computation and control. Springer, pp 451–464

Žilinskas A (2012) On strong homogeneity of two global optimization algorithms based on statistical models of multimodal objective functions. Appl Math Comput 218(16):8131–8136