Non-constructive interval simulation of dynamic systems

Wei Pang, Allan M. Bruce, George M. Coghill
Department of Computing Science
University of Aberdeen, UK
{pang.wei, g.coghill}@abdn.ac.uk

Abstract

Inspired by non-constructive simulation developed in qualitative reasoning, we present a non-constructive interval simulation algorithm for simulating dynamic systems. We recast two integration methods from traditional numerical simulation to make them suitable for non-constructive interval simulation. We then proposed an iterative interval narrowing algorithm to control the growth of intervals during simulation, and we also designed several simulation modes. The simulation algorithm has been both theoretically and experimentally validated.

1 Introduction

Numerical simulation of dynamic systems has been widely used in many engineering and scientific fields [Angermann, 2011]. It numerically simulates differential equation models, such as Ordinary Differential Equation (ODE) and Differential Algebraic Equation (DAE) models, thereby describing how the dynamic system evolves with time.

However, for many real-world problems, due to incomplete knowledge and data, the initial values of model variables and/or the model parameter values have to been given as intervals with lower and upper bounds. Simulating differential equation models as such necessitates the use of *interval analysis* [Moore, 1966], which led to the development of *interval simulation*. In history Interval simulation has attracted the interest of both Qualitative Reasoning (QR) [Kuipers, 1994] and Numerical Simulation (NS) communities. This resulted in the parallel development of interval simulation (in QR the subfield is called semi-quantitative simulation), and therefore there is a gap between these two fields.

In this research we aim to bridge this gap through a novel interval simulation framework. This framework is based on QR but employs techniques developed in NS. Furthermore, we have designed the simulation algorithm to be *non-constructive*, a simulation approach originating from QR (see details in Section 3). This enables the proposed approach to straightforwardly handle models with algebraic loops, which are normally dealt with by NS algorithms through additional, and often complicated and unreliable, operations [Cellier, 1991a].

In the rest of the paper, we first describe the development of interval simulation within the NS and QR fields in Section 2. Then in Section 3 we present the motivations of the research. In Section 4 we introduce the *Morven* formalism used in our approach. In Section 5 the proposed non-constructive interval simulation approach is described in detail. In Section 6 we provide the theoretical analysis on the proposed simulation algorithm. This is followed by the report of a series of experiments in Section 7. Finally Section 8 concludes the paper and explore the future work.

2 Background

2.1 From Numerical Simulation to Interval Simulation

Researchers in NS extended IVP (Initial Value Problem) for ODEs to the interval IVP for ODEs as follows:

$$Y'(t) = F(Y), \tag{1}$$

$$Y(t_0) = Y_0. \tag{2}$$

In the above, $Y \in \mathbb{IR}^n$ is an unknown n-dimensional interval-valued vector variable, where $\mathbb{IR}$ denotes the set of real intervals. $Y_0 \in \mathbb{IR}^n$ is the given initial values. $Y'(t)$ is the first derivative of Y with respect to time t, and $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a given function.

Some researchers further consider situations when the parameters are also intervals, which lead to the replacement of Equation (1) by the following equation:

$$Y'(t) = \mathcal{F}(Y, \theta). \tag{3}$$

In the above, $\mathcal{F}$ is a vector function containing interval-valued parameters, and $\theta \in \mathbb{IR}^m$ is the parameter vector with m being the number of parameters in the model.

Many efforts in NS have been made to recast numerical ODE solvers to deal with interval IVPs for ODEs described by Equations (1) $\sim$ (2) [Nedialkov *et al.*, 1999].

Apart from ODEs, DAEs (Differential Algebraic Equation) are often used in practice. A set of DAEs is represented as follows:

$$F(t, y, y', y'', ..., y^{[m]}) = 0 \tag{4}$$

In the above y is an unknown n-dimensional vector variable, and y', y'',..., $y^{[m]}$ are derivatives of y with respect to time t. Function F is a mapping $R^{n \cdot m+1} \rightarrow R^n$.

The initial values at time t_0 is a solution of DAEs in the

Table 1: Some qualitative constraints in *Morven* and their corresponding mathematical equations

Morven Constraints	Mathematical Equations
sub (dt 0 Z, dt 0 X, dt 0 Y)	$Z(t) = X(t) - Y(t)$
mul (dt 0 Z, dt 0 X, dt 0 Y)	$Z(t) = Y(t) * X(t)$
div (dt 0 Z, dt 0 X, dt 0 Y)	$Z(t) = X(t)/Y(t)$
func (dt 0 Y, dt 0 X)	$Y(t) = f(X(t))$
sub (dt 1 Z, dt 0 X, dt 0 Y)	$dZ(t)/dt = X(t) - Y(t)$
func (dt 1 Y, dt 0 X)	$dY(t)/dt = f(X(t))$

form of Equation (4) in the following form:

$$F(t_0, y(t_0), y'(t_0), y''(t_0), ..., y^{[m]}(t_0)) = 0 \qquad (5)$$

The IVP for DAEs is to numerically simulate DAEs in the form of Equation (4) given the initial values in Equation (5). Similar to the way we represent the interval IVPs for ODEs, the interval IVPs for DAEs can be formulated according to Equations (4) and (5). The development of interval DAE solvers also received some attention [Nedialkov and Pryce, 2005] recently.

2.2 Semi-quantitative Simulation

In the Qualitative Reasoning (QR) community, qualitative simulation uses Qualitative Differential Equations (QDEs) [Kuipers, 1994; Coghill, 1996; Bruce and Coghill, 2005]. Table 1 lists some *Morven* constraints and the corresponding mathematical equations.

Some researchers started from qualitative simulation engines such as QSIM and incorporated incomplete quantitative information (in the form of intervals), and this led to the development of *semi-quantitative simulation* algorithms, for instance, Q2 [Kuipers and Berleant, 1988] and Q3 [Berleant and Kuipers, 1997].

3 Motivations of the Research

From Section 2 we can see that the researchers from the NS and QR communities took different approaches to interval simulation, and there is a gap between these two communities.

From the QR perspective, there is room for improvement for interval simulators: first, as far as the authors are aware, up till now only the Euler and Runge-Kutta integration methods have been used in QR. So one motivation of our research is to explore the potential of a wider range of integration techniques as applied to QR-based interval simulation.

Second, there are both constructive and non-constructive simulation [Wiegand, 1991; Coghill and Chantler, 1999] in QR. Constructive simulation requires the explicit forms of the derivatives of variables. In constructive simulation, at time point t_i, the derivatives are first calculated directly from their explicit forms, and then used for the estimation of the magnitudes of variables at the next time point t_{i+1}. Constructive simulation has been extensively used in NS, but one limitation is that when models are given as implicit forms (i.e., DAEs), additional procedures have to be performed to obtain the explicit forms, and such procedures may be computationally expensive. In addition, to solve a DAE constructively, the structure analysis may fail in some ill-structured DAEs [Pryce, 2001], which makes it impossible to obtain the explicit forms. Furthermore, it is not straightforward for constructive simulation to deal with models containing algebraic loops, which are sometimes used when modelling some real-world systems [Cellier, 1991b].

Table 2: The *Morven* Model for the Single Tank System

Differential Plane 0	
C1: mul (dt 0 q_o, dt 0 k, dt 0 V)	$(q_o = kV)$
C2: sub (dt 1 V, dt 0 q_i, dt 0 q_o)	$(V' = q_i - q_o)$
Differential Plane 1	
C3: mul (dt 1 q_o, dt 1 k, dt1 V)	$(q_o' = kV')$
C4: sub (dt 2 V, dt 1 q_i, dt1 q_o)	$(V'' = q_i' - q_o')$

Within QR there are a number of non-constructive qualitative simulators, such as QSIM and FuSim [Shen and Leitch, 1993]. Non-constructive simulation essentially employs a generate-and-eliminate strategy, and it does not require the explicit forms of variables or their derivatives. This is particularly effective when dealing with some ill-structured DAEs or DAEs containing algebraic loops.

In interval simulation, variables take interval values instead of qualitative values. However, under the QR framework a differential equation model with interval initial conditions and parameter values can be converted into a semi-quantitative model composed of several constraints. So similar to qualitative simulation, in interval simulation we can use each constraint to determine the ranges of interval values for all variables of this constraint (generate). Then the resulting interval value for a variable will be the intersection of all intervals obtained from each individual constraint (eliminate). This means we can also perform the interval simulation in a non-constructive manner.

The above consideration leads to another motivation of this research: implement a non-constructive interval simulator. We expect such research will contribute to both the NS and QR communities.

4 The *Morven* Framework

In this research we use the *Morven* [Coghill, 1996] formalism to represent semi-quantitative models. The *Morven* framework is a constraint-based fuzzy qualitative system. Qualitative constraints in a *Morven* model are distributed over multiple *differential planes*. Qualitative variables in *Morven* are in the form of variable length vectors.

Consider the ODE model of the single tank system:

$$q_o = kV, \qquad (6)$$

$$dV/dt = q_i - q_o, \qquad (7)$$

where V is the volume of the liquid in the tank, q_i is the inflow, q_o is the outflow, and k is a positive constant coefficient.

The above ODE model can be converted into a *Morven* model shown in Table 2. It is noted that *Morven* can use function constraints ("func") to represent qualitative models.

5 Non-constructive Interval Simulation

There are two phases in non-constructive qualitative simulation [Coghill and Chantler, 1999]: Transition Analysis (TA) and Qualitative Analysis (QA), which correspond to interval integration and interval refinement in interval simulation, respectively, and they are two most important components of non-constructive interval simulation.

5.1 Interval Arithmetic

The interval arithmetic used in our algorithm is defined in Table 3.

Table 3: Interval Arithmetic Operations

Let:	$m = [a \quad b], n = [c \quad d]$	
Operation	Result	Conditions
$-n$	$[d \quad c]$	all n
$\frac{1}{n}$	$[\frac{1}{d} \quad \frac{1}{c}]$	$c, d > 0$ or $c, d < 0$
	$[-\infty \quad \infty]$	$c \leq 0$ and $d \geq 0$
$m + n$	$[a + c \quad b + d]$	all m, n
$m - n$	$[a - d \quad b - c]$	$m \neq n$
	$[0 \quad 0]$	$m = n$
$m \times n$	$[ac \quad bd]$	$m = n$ and ($c, d > 0$ or $c, d < 0$)
	$[0 \quad bd]$	$m = n$ and $c \leq 0$ and $d \geq 0$
	$[ac \quad bd]$	$m \neq n$ and $a, c > 0$
	$[bc \quad ad]$	$m \neq n$ and $a > 0$ and $d < 0$
	$[bc \quad bd]$	$m \neq n$ and $a > 0$ and $c \leq 0$ and $d \geq 0$
	$[ad \quad bc]$	$m \neq n$ and $b < 0$ and $c > 0$
	$[bd \quad ac]$	$m \neq n$ and $b < 0$ and $d < 0$
	$[ad \quad ac]$	$m \neq n$ and $b < 0$ and $c \leq 0$ and $d \geq 0$
	$[ad \quad bd]$	$m \neq n$ and $a \leq 0$ and $b \geq 0$ and $c > 0$
	$[bc \quad ac]$	$m \neq n$ and $a \leq 0$ and $b \geq 0$ and $d < 0$
	$[min(ad, bc) \quad max(ac, bd)]$	$m \neq n$ and $a \leq 0$ and $b \geq 0$ and $c \leq 0$ and $d \geq 0$
$\frac{m}{n}$	$[1 \quad 1]$	$m = n$
	$m \times \frac{1}{n}$	$m \neq n$
$m \cap n$	$[max(a, c), min(b, d)]$	$a \leq c \leq b$ or $c \leq a \leq d$
	$\emptyset$	$b > c$ or $d > a$
$m \cup n$	[max(a, c), min(b, d)]	

$m \neq n$ denotes that the intervals do not correspond to the same interval whereas $m = n$ indicates that the intervals do correspond to the same interval. $a, c > 0$ indicates that both intervals are positive whereas $b, d < 0$ dictates that both intervals are negative. $c \leq 0$ and $d \geq 0$ (as well as $a \leq 0$ and $b \geq 0$) governs that the interval spans zero. It is possible to define $m \times n$ for when both intervals span zero however it has been left out in this table for simplicity.

5.2 Integration Methods

In this subsection, we investigate which integration methods can be used in non-constructive simulation as $Morven$ models can represent those ill-structured ODE and DAE models which are impossible to simulate constructively.

We have explored several common integration methods: (1) Euler methods [Ascher and Petzold, 1998] (including both forward and backward Euler methods), (2) Runge-Kutta methods [Butcher, 2008], (3) Taylor Series Expansion [Arfken *et al.*, 2005], (4) the linear multi-step methods [Butcher, 2003a], including the Adams-Bashforth (AB) methods [Butcher, 2003b], Adams-Moulton and Backward Differentiation Formulas (BDF) [Hairer *et al.*, 1993], (5) the predictor-corrector methods [Press *et al.*, 1992], including the Euler Trapezoidal method and the Adams-Bathforth-Moulton method [Mathews and Fink, 2004].

Among all the above mentioned methods, we identified that Taylor Series (with the forward Euler method being a special case) and AB methods can be used in non-constructive simulation. They can also be easily recast for interval simulation by using the interval mathematics defined in Section 5.1. Therefore, in this research these two integration approaches will be investigated.

We take the Euler methods as an example to demonstrate how we determine the suitability of an integration method for non-constructive simulation, and due to page limit, we will not present the investigation on other integration methods.

We study two streams of Euler methods: forward and backward. The forward Euler method for constructive simulation is given as follows:

$$y_{n+1} = y_n + hf(y_n) \tag{8}$$

In the above y_n is the magnitude of y at time step t_n; $f(y_n)$ is the explicit form of y'_n given by the model: $y'_n = f(y_n)$; y_{n+1} is the magnitude of y at the next time step t_{n+1}, and h is the step size ($t_{n+1} - t_n$). In constructive simulation, at time step t_{n+1}, first we calculate the value of y_{n+1} according to Equation (8), then this value will be used to calculate the value of y'_{n+1} in the next simulation step t_{n+1} by evaluating $f(y_{n+1})$.

In the context of non-constructive simulation, at time step t_{n+1}, the value of $f(y_n)$ cannot be calculated directly as the form of $f(y_n)$ is not explicitly given. However, the value of y'_n can be taken from previous calculation at time step t_n, and replace $f(y_n)$ in Equation (8), which is shown below:

$$y_{n+1} = y_n + hy'_n \tag{9}$$

In addition, the initial value of y'_0 is either given or calculated by the interval narrowing algorithm which will be described in Section 5.3. This means that the forward Euler method can be used in non-constructive simulation.

The backward Euler method for constructive simulation is given as follows:

$$y_{n+1} = y_n + hf(y_{n+1}), \tag{10}$$

where $f(y_{n+1})$ is the explicit form of y'_{n+1}: $y'_{n+1} = f(y_{n+1})$. In constructive simulation the above equation has to be solved to obtain the precise value of y_{n+1}, for instance, by the fixed point iteration method [Burden and Faires, 2000].

On the other hand, in non-constructive simulation at the time step t_{n+1} the explicit form $f(y_{n+1})$ cannot be obtained straightforwardly, and the value of y'_{n+1} is not calculated yet. This means the backward Euler is not suitable for non-constructive simulation.

5.3 Interval Narrowing Algorithm

Having defined the interval mathematics and chosen the integration methods, the next component to be developed is the interval narrowing algorithm. The interval narrowing algorithm is to control the growth of intervals during simulation, which is achieved by iteratively applying the model constraints to intervals.

We first propose the Inverse Constraint Operations, as detailed below: for each constraint in the model, we obtain all of its mathematically equivalent forms, each of which is called an inverse of this constraint in this paper. For example, consider the following constraint:

$$A = B + C, \tag{11}$$

its inverses will be the following two:

$$B = A - C, \tag{12}$$

$$C = A - B. \quad (13)$$

Then this constraint together with its inverse(s) are used to narrow down the intervals of relavent variables after an integration step. For example, suppose after an integration step, the intervals for variables A, B, and C are A = [5, 6], B = [3, 4], and C = [2.5, 3.5].

Applying interval arithmetic defined in Table 3 to Equation (11), we can determine the range of A by $(B + C)$ as follows: [5.5, 7.5] = [3, 4] + [2.5, 3.5]. Consider the initial interval $A = [5,\ 6]$ from integration, the intersection of these two intervals will be: A= [5, 6] ∩ [5.5, 7.5] = [5.5, 6].

To narrow the rest of the variables the inverse constraints should be used: according to Equation (12), the interval for B should be: [2, 3.5] = [5.5, 6] - [2.5, 3.5]. Again from integration, $B = [3,\ 4]$. Taking the intersection of the intervals gives $B = [3,\ 3.5]$. Similarly, using Equation (13), the interval for C is: [2 , 3] = [5.5, 6] - [3, 3.5], but $C = [2.5,\ 3.5]$ from integration therefore taking the intersection $C = [2.5,\ 3]$.

Finally, intervals for all variables narrowed as much as possible are: [2.5, 3] = [5.5, 6] - [3, 3.5].

After this process the intervals for A, B and C are narrowed as much as possible by reasoning over Constraint (11) (and its two inverses) alone. However, these updated values may result in further narrowing in other constraints; hence the process is repeated until no more changes are made in the whole model or the change is within a given threshold, which is a very small value and determines the simulation precision. Due to the narrowing of intervals using this Inverse Constraint Operations, not much looping of the whole model is required.

The interval narrowing algorithm described above is essentially a Waltz algorithm [Waltz, 1975] applied to interval values. In particular, the soundness and completeness of the Waltz algorithm applied to interval refinement has been extensively studied by Davis [Davis, 1987]. It is noted that in Q3 [Berleant and Kuipers, 1997] the Waltz algorithm was also used to refine interval values. However, Q3 applied the Waltz algorithm to the constraint network composed of constraints instantiated from the model across different time points, one of the reasons for which is to propagate the quantitative information (in the form of intervals) annotated on the given and newly interpolated states throughout the network. While in our interval simulation algorithm the Waltz algorithm is used only at the current time point to ensure the generation of tight interval values for precise estimation of variable values as well as for the integration towards the next time point.

We finally point out that this interval narrowing algorithm is suitable for the situation when there exist time-invariant interval parameters in the model, as described in Equation (3), because the constant intervals for these parameters can be directly processed by the interval arithmetic during interval narrowing. This makes our simulation different from the widely used approach suggested by Lohner [Lohner, 1988] in the NS community, which treats the time-invariant interval parameters as independent state variables and thus increases the complexity of simulation.

5.4 Modes of Simulation

A common problem in interval simulation is that the intervals begin to widen and then eventually become uncontrollable during simulation. This problem is called the "wrapping effect" [Moore, 1966] in NS. In this research we offer different approaches to deal with the interval widening problem, and each approach is termed a *simulation mode* in this paper. For ease of description, the simulation which does not take any additional approach to reduce the spurious behaviours is also considered as a simulation mode: the *Basic Interval Simulation* (BIS) mode.

Basic Interval Simulation

In the *Basic Interval Simulation* (BIS) mode, the previously mentioned three modules are straightforwardly employed to simulate a model. In this sense BIS will demonstrate how the basic non-constructive simulation algorithm is performed.

In the BIS mode, at the beginning of the simulation, users are asked to give the size of the integration step δt and the total simulation time t_{tot}, and therefore the number of simulation steps is given by: $num = t_{tot}/\delta t$.

At the initial time step t_0, given an incomplete initial state, which specifies the initial intervals for some model variables, the interval narrowing algorithm is first used to narrow down as much as possible these initial intervals. Meanwhile, based on the known initial intervals, the interval narrowing algorithm also tries to infer the initial intervals of those variables whose values are not specified. Another function of the interval narrowing algorithm is to check whether the initial state is consistent with the model, and if the initial state is inconsistent, the simulation will not proceed.

During simulation a repository R is maintained to record the history of simulation data, and each element in R is a four-tuple $< Var, Der, [a, b] : t_i >$, where Var is the name of the variable; Der is a non-negative integer which specifies the order of derivative of Var (0 means the magnitude); the third and fourth elements represent the interval value and the time step, respectively.

At each new time step t_i all derivatives of all variables are first integrated, and the relevant intervals used for integration can be retrieved from the repository R. For the Taylor method, only the data at time step t_{i-1} will be retrieved, and the data may include intervals for magnitudes and derivatives of variables. For the Adams-Bashforth method, apart from data at t_{i-1}, data before t_{i-1} will also be retrieved, but only intervals for the first derivatives of relevant variables will be used.

Then in the interval narrowing process, all the intervals for all derivatives are iteratively checked against each constraint of the model. After the interval narrowing process, all the updated intervals are stored in the repository R, which are ready for the simulation at the next time step t_{i+1}. Finally we point out that BIS is complete, as will be discussed in Section 6.1.

Sub-interval Simulation

One way to handle the uncontrollable growth of intervals during simulation is to simulate the model many times but using a smaller sub-interval each time. This approach is called Sub-interval Simulation (SIS) in this paper. The motivation of SIS

is that when the initial intervals are very small the simulation will suffer less from the interval divergence over time.

In the SIS mode, the initial interval for each variable/derivative is divided into n equal and non-overlapping sub-intervals. Then for each combination which takes one sub-interval from each initial interval, an initial state is generated and the BIS is used to simulate the model with this initial state. After all possible combinations of sub-intervals have been used for simulation, the final simulation results will be the union of all individual simulations.

As with the Basic Interval Simulation, the Sub-interval Simulation is complete in the sense that it can guaranteed to bound all real solutions, as will be discussed in Section 6.1. Although complete, the Sub-interval Simulation requires a large number of initial states generated for simulation, which becomes a major concern in terms of the computational efficiency. However, it is often the case that not all the generated initial states are consistent with the model, and these states will be discarded during the interval narrowing process, which makes the simulation less expensive.

Monte-Carlo Interval Simulation

Although the sub-interval simulation is complete, its computational cost may increase exponentially with the increase of the number of intervals. Another simulation mode is to use the Monte-Carlo method [Rubinstein, 1981]. That is: instead of exhaustively using all combinations of sub-intervals as in SIS, we only randomly sample a specified number of combinations. This simulation mode is called Monte-Carlo Interval Simulation (MCIS) in this paper.

The sub-intervals used by MCIS can be smaller than those in SIS, which means it can generate a tighter enclosure of the simulation trajectories. Theoretically speaking, MCIS has to sample an infinite number of combinations to bound all real solutions, but as the sample space of MCIS is finite (each interval is divided into a finite number of sub-intervals), compared with the MC method applying to infinite sample space, it is more likely to cover all solutions if the samples are sufficient enough.

Point Simulation

The point simulation samples points (intervals with zero width) rather than sub-intervals from given initial intervals to approximate the solutions. The point simulation is similar to traditional numerical simulation, except that non-constructive simulation is used. The motivation for using points for simulation is to reduce the spurious behaviours, because the trajectories obtained from simulation with point initial conditions are zero width.

Several point simulation methods are developed in this research: (1) the Extreme Point Simulation, in which each initial state is formed by taking the upper or lower bound from each initial interval. Taking the extreme points of each interval gives an approximate range of possible values whilst maintaining an efficient method. This method is sound but incomplete in the sense that the solutions found contains no spurious ones but it may not cover every possible solution.

(2) Regular-spaced Point Simulation: As with the Sub-interval Simulation, the Regular-spaced Point Simulation method takes each interval in the initial state and splits it into a number of states. These states contain a number of regular-spaced points which approximate the interval. This method is theoretically sound and complete when the number of points in each interval tends to infinity. As with the Sub-interval Simulation, this method is exponential in the number of intervals. However, similar to Sub-interval Simulation, it also has the benefit that not every state has to be simulated because some of them will be inconsistent with the model.

(3) Monte-Carlo Point Simulation: For each initial interval, we randomly choose a point within it and thus form an initial state. Then we generate a specified number of such initial states to perform simulation. The advantages of this approach are: using points guarantees that the solution is sound and using Monte-Carlo methods makes the solution tend to be complete and more efficient than Regular-spaced Point Simulation.

A Summary of All Simulation Modes

In this subsection, we proposed several simulation modes to improve the simulation. These simulation modes are classified by two categories: simulation using real intervals and simulation using group of points to approximate the real solutions. We also offer both deterministic and Monte-Carlo approaches to perform the simulation.

In practice, the choices of simulation modes mainly depend on two factors: the requirements of different problems and the computational cost.

6 Theoretical Analysis of the Simulation Algorithm

In this section, we present some theoretical analysis on the completeness, soundness, convergence, and stability of the proposed algorithm. As the algorithm is a collection of integration methods and simulation modes, we have to analyse every combination of simulation modes and integration methods. We first study the completeness and soundness of the algorithm under different simulation modes, then we further analyse the convergence and stability of the algorithm. All proofs of the lemmas and theorems are not presented in this paper due to page limit.

6.1 Completeness and Soundness

The completeness means that the interval simulation algorithm must bound all real solutions. The soundness means that the simulation results should be a subset of the actual solution. As for the completeness, we present the following two theorems:

Theorem 1. *Non-constructive Interval Simulation under the Basic Interval Simulation mode is complete.*

Theorem 2. *Non-constructive Interval Simulation under the Sub-interval Simulation mode is complete.*

BIS is not sound because of the wrapping effect, which is intrinsic to interval arithmetic. Similarly, SIS is also not sound. For the point simulation modes, we give the following lemma and theorem:

Lemma 1. *Non-constructive numerical simulation using the BIS mode is complete and sound.*

Theorem 3. *All Point Simulation modes are sound, but not complete.*

6.2 Convergence and Stability

In this section we discuss the convergence and stability of the algorithm. The convergence and stability analysis presented in this section is influenced by Berleant and Kuipers [Berleant and Kuipers, 1997], who are in turn inspired by Moore [Moore, 1979] and proofs of convergence and stability in numerical simulation [Gear, 1971]. For interval simulation, a simulation algorithm is convergent if at any time point the uncertainty of any variable values is eliminated when the integration step approaches zero and the uncertainty of initial conditions does not exist. Here the uncertainty of a variable value in the context of interval simulation means the width of intervals. For interval simulation, we say a simulation algorithm can achieve $h \rightarrow 0$ stability [Berleant and Kuipers, 1997] if at any time point the uncertainty of variable values is bounded by the uncertainty of initial conditions when the integration step h approaches zero.

We first give the following lemma, which shares some similarities with Lemma 1 in [Berleant and Kuipers, 1997].

Lemma 2. *Consider the first order differential equation with a given initial condition Y_0, assume its explicit form is as follows:*

$$Y' = F(Y), \tag{14}$$

where F is an interval valued function of Y, although this explicit form cannot always be obtained as function $\mathcal{F}$ is not always solvable. Suppose $Y(t) \subseteq [lo, hi]$, where $lo, hi \in \mathbb{R}$. We further assume that $F(Y)$ is defined when $Y(t) \subseteq [lo, hi]$, and $F(Y)$ is calculated by using the interval arithmetic defined in Table 3. Suppose the corresponding real rational function of $F(Y)$ is $f(y)$.

Let Y_n be the simulated value for Y at time step n. If the given initial condition $Y_0 \subseteq [lo, hi]$, and the **forward Euler method** *is used, there exists a constant K such that*

$$|Y_n| \leq |Y_0| + Kh \tag{15}$$

In the above operator $|\cdot|$ denotes the width of an interval; h is the integration step size; Y_n is the simulated interval at time step n.

Similarly, if the two-step AB method and Taylor method are used, we have the following two lemmas:

Lemma 3. *If all the assumptions are the same as those made in Lemma 2 except that* **the AB method** *is used, we still have statement (15).*

Lemma 4. *In Lemma 2 if the Taylor method is used, we can still have statement (15).*

Equipped with the above lemmas, we give the theorem of convergence and stability:

Theorem 4. *(Convergence and Stability for a system of ODEs) Consider the following system of first order differential equations:*

$$\vec{\mathcal{F}}(t, \mathbf{Y}, \mathbf{Y}') = 0 \tag{16}$$

where $\mathbf{Y}$ is an interval valued vector, and $\vec{\mathcal{F}}$ is a vector of interval valued functions of $\mathbf{Y}$. Assume for each element of vector $\mathbf{Y}$, $Y_{(i)}(t) \subseteq [lo, hi]$. We further assume that we can solve $\vec{\mathcal{F}}$ in (16) to obtain the explicit form as follows:

$$\mathbf{Y}' = \mathbf{F}(\mathbf{Y}), \tag{17}$$

where $\mathbf{F}$ is a vector of interval valued functions of $\mathbf{Y}$. Suppose for each element of $\mathbf{Y}$, $Y_{(j)}(t) \subseteq [lo, hi]$, where $lo, hi \in \mathbb{R}$. We further assume that $\mathbf{F}(\mathbf{Y})$ is defined when each $Y_j(t) \subseteq [lo, hi]$, and for each element of $\mathbf{F}(\mathbf{Y})$, $F_{(j)}$ is calculated by using the interval arithmetic defined in Table 3. Suppose the corresponding real rational function of $F(Y)$ is $f(y)$.

Let $\mathbf{Y_n}$ be the simulated value for $\mathbf{Y}$ at time step n. If for each element of the given initial condition $\mathbf{Y_0}$, $Y_{0j} \subseteq [lo, hi]$, and the **Tayor or AB integration methods** *are used, there exists a constant K such that*

$$||\mathbf{Y_n}|| \leq ||\mathbf{Y_0}|| + Kh \tag{18}$$

In the above, operator $||\cdot||$ denotes the norm of an interval valued vector such that for an interval valued vector $\mathbf{A} = \{A_1, A_2, \cdot\cdot, A_n\}$, $||\mathbf{A}|| = max(|A_1|, |A_2|, \cdot\cdot, |A_n|)$, where $|\cdot|$ denotes the width of an interval; h is the integration step size; $||\mathbf{Y_n}||$ is the simulated interval vector at time step n.

It is noted that Theorem 4 can be extended to higher order systems as any such systems can be reduced to first order systems. From statement (18), we see that given a precise initial condition $||\mathbf{Y_0}|| = 0$, for any fixed simulation time $t = nh$, when $h \rightarrow 0$, we will have $||\mathbf{Y_n}|| \rightarrow 0$. This means that the algorithm is *convergent* when the simulation step approaches zero. According to the $h \rightarrow 0$ stability defined in [Berleant and Kuipers, 1997] and shown below:

$$||\mathbf{Y_n}|| \leq K||\mathbf{Y_0}||, \tag{19}$$

where K is a constant, we can see that our proposed simulation algorithm possesses the $h \rightarrow 0$ stability from Theorem 4.

In Theorem 4 we assume that (16) can be solved to obtain its explicit form (17). This actually indicates that the underlying model is a system of ODEs. As we know that (16) could also be a system of DAEs, we give the following theorem:

Theorem 5. *(Convergence and Stability for a system of DAEs) Suppose by solving (16), we can only obtain the following form:*

$$\mathbf{X}' = \mathbf{F}(\mathbf{X}, \mathbf{Z}), \tag{20}$$

$$\mathbf{0} = \mathbf{G}(\mathbf{X}, \mathbf{Z}). \tag{21}$$

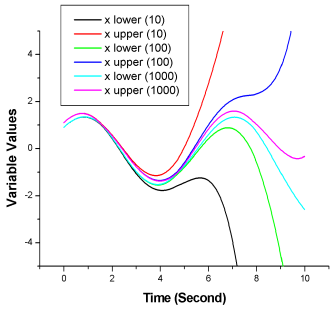
In the above $\mathbf{F}$ and $\mathbf{G}$ are two vectors of interval valued functions. Vector $\mathbf{Y}$ in (16) can be formed by combining vectors $\mathbf{X}$ and $\mathbf{Z}$ together. (This means that the underlying model is a system of DAEs.)

If we assume that $\mathbf{F}$ and $\mathbf{G}$ are defined when each $Y_j(t) \subseteq [lo, hi]$, and the other assumptions are the same as those in Theorem 4, there exist three constants K_1, K_2, and K_2 such that

$$||\mathbf{X_n}|| \leq ||\mathbf{X_0}|| + K_1 h \tag{22}$$

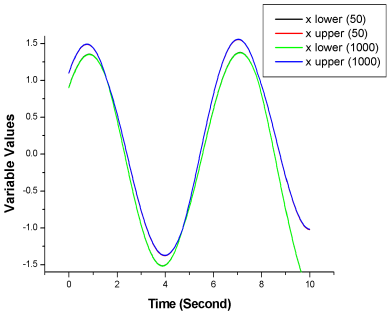
$$||\mathbf{Z_n}|| \leq K_2||\mathbf{Z_0}|| + K_3 h \tag{23}$$

From this theorem we can conclude that the convergence and stability can still be achieved when dealing with DAEs. Finally, from Theorems 4 and 5 we see that in non-constructive interval simulation the uncertainty of simulation results measured by the norm of vectors at each simulation step is determined by two factors: the uncertainty of initial



"x lower" and "x upper" mean the lower and upper bounds of x, respectively. Numbers in the brackets indicate the numbers of sub-intervals being used.

Figure 1: Sub-interval Simulation



"x lower" and "x upper" mean the lower and upper bounds of x, respectively. Numbers in the bracket indicate the numbers of Monte-Carlo Sample Intervals being used. In this figure as the results of simulation using 50 samples and those using 1000 samples largely overlapped, only two curves can be clearly seen.

Figure 2: Monte-Carlo Interval Simulation

conditions and the size of the simulation step. To reduce the uncertainty, we can either use a smaller simulation step or split initial intervals into several subintervals. This justifies the use of all the simulation modes in addition to BIS.

7 Experiments

In this section we only report part of the experimental results due to page limit, and readers are referred to [Pang *et al.*, 2012] or the supplementary material for a full experiment report.

The model for the spring-mass system is given as: $x'' = F - kx$, where F is the constant external force, k is a constant parameter, and x is the displacement of the mass with respect to the equilibrium position. The initial condition is set as follows: $x = [0.9, 1.1]$, $x' = [1, 1]$, $F = [0, 0]$. The sub-interval simulation and Monte-Carlo Interval Simulation results are shown in Figure 1 and Figure 2, respectively.

The Van der Pol oscillator model is given as $x'' = -P(x^2 - 1)x' - Qx$, where P and Q are two parameters. The Regular-spaced Point simulation for this model is given in Figure 3, where the initial condition is x, $x' = [0.5, 1.5]$ and $P, Q = [1, 1]$. The Monte-Carlo Points simulation on this model is given in Figure 4, where the initial condition is

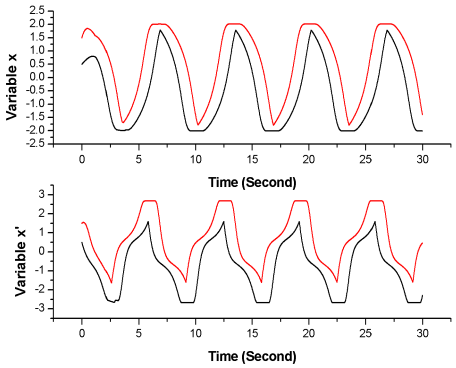


Figure 3: Regular-spaced Point Simulation of the Van der Pol Oscillator (20 Points Per Interval)

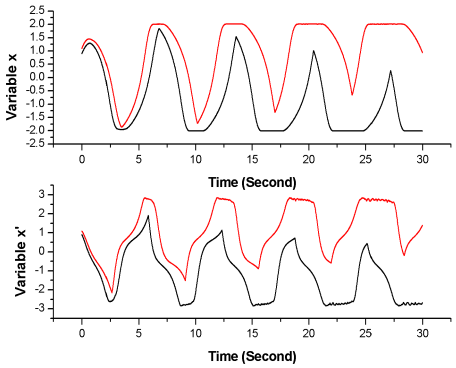


Figure 4: Monte-Carlo Points Simulation of the Van der Pol Oscillator with All Variables taking Interval Values

$x, x', P, Q = [0.9, 1.1]$ (all initial values and parameter values are intervals).

8 Conclusions and Future Work

In this paper we have presented a novel non-constructive interval approach for the simulation of dynamic systems. We established our novel non-constructive interval simulation approach by (1) recasting existing NS integration methods which are feasible for non-constructive simulation, (2) providing an iterative interval narrowing algorithm to deal with the interval widening effect, and (3) offering several simulation modes to meet different requirements.

In the future more simulation modes will be investigated and implemented so that we can better sample the sub-intervals or points from initial intervals of variables and parameters. Finally, we expect that the research results presented in this paper will contribute to both the NS and QR communities, and we foresee the non-constructive approach as a fruitful research direction for simulation at both quantitative and semi-quantitative levels.

References

[Angermann, 2011] Lutz Angermann, editor. *Numerical Simulations - Applications, Examples and Theory*. InTech, 2011.

[Arfken *et al.*, 2005] George B. Arfken, Hans J. Weber, and Frank E. Harris. *Mathematical Methods for Physicists, Sixth Edition: A Comprehensive Guide*, chapter 5.6, pages 352–362. Academic Press, 2005.

[Ascher and Petzold, 1998] Uri M. Ascher and Linda Ruth Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, chapter 3, pages 37–67. SIAM, 1998.

[Berleant and Kuipers, 1997] Daniel Berleant and Benjamin Kuipers. Qualitative and quantitative simulation: bridging the gap. *Artificial Intelligence*, 95(2):215–255, 1997.

[Bruce and Coghill, 2005] A. M. Bruce and G. M. Coghill. Parallel fuzzy qualitative reasoning. In *Proceedings of the 19th International Workshop on Qualitative Reasoning*, pages 110–116, Graz, Austria, 2005.

[Burden and Faires, 2000] Richard L. Burden and J. Douglas Faires. *Numerical Analysis (7th Edition)*, chapter 2, pages 56–66. Brooks Cole, 2000, December 2000.

[Butcher, 2003a] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*, pages 97–106. John Wiley, 2003.

[Butcher, 2003b] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*, page 103. John Wiley, 2003.

[Butcher, 2008] John Charles Butcher. *Numerical methods for ordinary differential equations (2nd Edition)*, chapter 2, pages 93–103. John Wiley and Sons, 2008.

[Cellier, 1991a] Francois E. Cellier. *Continuous System Modeling*, chapter 5, pages 167–169. Springer-Verlag, 1991.

[Cellier, 1991b] Francois E. Cellier. *Continuous System Modeling*, chapter 3, pages 66–68. Springer-Verlag, 1991.

[Coghill and Chantler, 1999] G. M. Coghill and M. J. Chantler. Constructive and non-constructive asynchronous qualitative simulation. In *Proceedings of the 13th Inernational Workshop on Qualitative Reasoning*, pages 51–61, Loch Awe, Scotland, June 1999.

[Coghill, 1996] George M. Coghill. *Mycroft: A Framework for Constraint based Fuzzy Qualitative Reasoning*. PhD thesis, Heriot-Watt University, September 1996.

[Davis, 1987] Ernest Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, July 1987.

[Gear, 1971] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall, 1971.

[Hairer *et al.*, 1993] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics*, chapter III.1, pages 356–366. Springer, 1993.

[Kuipers and Berleant, 1988] B. J. Kuipers and D. Berleant. Using incomplete quantitative information in qualitative reasoning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-88)*, pages 324–329, Los Altos, CA, 1988. Morgan Kaufman.

[Kuipers, 1994] Benjamin Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, MA, 1994.

[Lohner, 1988] Rudolf Lohner. *Enclosing the solutions of ordinary initial and boundary value problems (in German)*. Ph.D. thesis, Universität Fridericiana Karlsruhe, Karlsruhe, Germany, 1988.

[Mathews and Fink, 2004] John H. Mathews and Kurtis K. Fink. *Numerical Methods Using MATLAB (4th Edition)*, chapter 9.6, pages 505–508. Prentice-Hall Inc., 2004.

[Moore, 1966] Ramon E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliff, New Jersey, 1966.

[Moore, 1979] Ramn E. Moore. *Methods and applications of interval analysis*, chapter 8, pages 93–99. SIAM, 1979.

[Nedialkov and Pryce, 2005] Nedialko S. Nedialkov and John D. Pryce. Solving differential algebraic equations by Taylor series (I): Computing Taylor coefficients. *BIT Numerical Mathematics*, 45(3):561–591, 2005.

[Nedialkov *et al.*, 1999] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, October 1999.

[Pang *et al.*, 2012] Wei Pang, G M. Coghill, and Alan M. Bruce. Non-constructive interval simulation of dynamic systems. Technique Report ABDNCS1202, Department of Computing Science, University of Aberdeen, 2012.

[Press *et al.*, 1992] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing (2nd Edition)*, chapter 16, pages 740–744. Cambridge University Press, 1992.

[Pryce, 2001] J. D. Pryce. A simple structural analysis method for DAEs. *BIT Numerical Mathematics*, 41(2):364–394, 2001.

[Rubinstein, 1981] Reuven Rubinstein. *Simulation and the Monte Carlo Method*. Wiley, 1981.

[Shen and Leitch, 1993] Qiang Shen and Roy Leitch. Fuzzy qualitative simulation. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4):1038–1061, 1993.

[Waltz, 1975] David Waltz. Understanding line drawings of scenes with shadows. In Patrick Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975.

[Wiegand, 1991] M. Wiegand. *Constructive Qualitative Simulation of Continuous Dynamic Systems*. PhD thesis, Heriot-Watt university, May 1991.