Non-constructive interval simulation of dynamic systems

Wei Pang, George M. Coghill, Allan M. Bruce

Technical Report Series **ABDN–CS–12–02**

Department of Computing Science **June 2012**
University of Aberdeen
King's College
Aberdeen AB24 3UE

Non-constructive interval simulation of dynamic systems

Wei Pang, George M. Coghill, Allan M. Bruce

Technical Report ABDN–CS–12–02
Department of Computing Science
University of Aberdeen

June 18, 2012

Abstract: In this report, inspired by non-constructive simulation developed in the qualitative reasoning field, we present a non-constructive interval simulation algorithm for the simulation of dynamic systems. To perform this kind of simulation, we first recast two integration methods, which were originally used in traditional numerical simulation, and made them suitable for performing interval simulation in a non-constructive manner. Then we proposed an iterative interval narrowing algorithm to control the growth of intervals during simulation. To achieve better accuracy and efficiency of the simulation, we designed several simulation modes to meet different requirements of various problems. The proposed simulation algorithm was theoretically studied in terms of its completeness, soundness, convergence, and stability. Finally two classical dynamic systems, as well as an electrical circuit model containing an algebraic loop, were used as test examples to demonstrate the validity of the proposed simulation approach.

Keywords: Qualitative reasoning; Interval simulation; Non-constructive simulation; Interval analysis; Monte-Carlo simulation; Algebraic-loop model

1 Introduction

Numerical simulation of dynamic systems has been widely used in many engineering and scientific fields [1] because of its ability to offer precise estimation of the behaviours of a dynamic system at a quantitative level. Numerical simulation starts from differential equation models, among which the most commonly used are Ordinary Differential Equation (ODE) and Differential Algebraic Equation (DAE) models, and tries to find numerical solutions of the model when it is hard or even impossible to find analytical ones. It is able to predict the exact numerical values of variables of interest at certain time points within the given time period, and thus to generate a trajectory for each variable along the time axis, thereby describing how the dynamic system evolves with time.

However, in many real-world problems, initial values of some variables are not easy to measure and consequently the initial values of these variables are often given in the form of intervals, each of which contains a lower and upper bound. In addition, sometimes some of the parameter values of a model cannot be precisely inferred and are also given as intervals. Simulating differential equation models with interval-valued initial values and/or parameter values necessitated the use of *interval analysis* [38] in the simulation algorithms, which led to the development of the field *interval simulation.* Interval simulation has attracted the interest of two different research communities: Qualitative Reasoning (QR) [31] and Numerical

Simulation (NS). Researchers from these two communities approach interval simulation from different angles, and utilise different methodologies to perform the study based on the existing results from their respective fields. This resulted in the parallel development of interval simulation (it is noted that in QR the corresponding subfield is called semi-quantitative simulation), and there is a gap between these two fields in terms of the study of interval simulation. consequently, there is room for improvement in the development of interval simulation.

In this research we aim to bridge this gap by means of a novel interval simulation algorithm. This simulation algorithm is based on QR research but employs some of the existing research results from the NS field. The proposed simulation algorithm will be theoretically studied and experimentally validated. Furthermore, we have designed the simulation algorithm to be *non-constructive*, a simulation approach originating from QR, details of which are given in Section 3. This enables the proposed approach to straightforwardly and effectively handle models with algebraic loops, which are normally dealt with by numerical simulation algorithms through additional operations [14], which could be complicated and unreliable.

In the rest of the report, we first describe the development of interval simulation within the NS and QR fields in Section 2. Then in Section 3 we determine the motivations of the research. In Section 4 we give an introduction to 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 present the theoretical analysis on the proposed simulation algorithm. This is followed by the report of a series of experiments in Section 7, which experimentally validate the proposed approach. Finally in Section 8 we conclude the report and explore the future work of the proposed approach.

2 Background

2.1 From Numerical Simulation to Interval Simulation

Within the NS community researchers are interested in fully parameterised models with precise initial values, and problems of numerically simulating such systems with initial values are called *Initial Value Problems* (IVPs). Models are often given in the explicit form of autonomous ODEs for ease of algorithm development as well as practical use, as formally described below:

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

$$y(t_0) = y_0. \tag{2}$$

If we use $\mathbb{R}$ to denote the set of real numbers, in the above Equation (1) represents a system of first order ODEs of dimension n, where $y \in \mathbb{R}^n$ is an unknown n-dimensional vector variable; variable $t \in \mathbb{R}$ stands for time; $y'(t)$ is the first derivative of y with respect to t; and $F : \mathbb{R}^n \to \mathbb{R}^n$ is a given function. Equation (2) gives the initial values for this ODE model, where $y_0 \in \mathbb{R}^n$ and $t_0 \in \mathbb{R}$.

Given the model and initial values described by Equations (1) and (2), researchers are interested in answering the following question: If at a given time period $[t_0, t_{end}]$, where $t_{end} \in \mathbb{R}$ and $t_{end} > t_0$, $y(t)$ is continuously differentiable, how can we approximately calculate a series of values $\{y(t_j)\}$, where $t_j \in [t_0, t_{end}]$? This question is termed the IVP for ODEs.

Over the past two centuries there have been many well-established numerical simulation algorithms developed for IVPs, examples of which include Euler methods [3], Runge-Kutta methods [12], and the Adams family [4].

However, some researchers in this community started to realise the importance of the situations when initial values are given as real intervals rather than a real number. A real interval Y is defined as the set of real numbers with given upper and lower bounds:

$$Y = [\underline{Y}, \overline{Y}] = \{y \in \mathbb{R} \mid \underline{Y} \le y \le \overline{Y}\}. \tag{3}$$

The interval IVP for ODEs is represented by the following two sets of equations:

$$Y'(t) = F(Y), \tag{4}$$

$$Y(t_0) = Y_0. \tag{5}$$

In the above equations, $Y \in \mathbb{IR}^n$ is an n-dimensional interval-valued vector variable, where $\mathbb{IR}$ denotes the set of real intervals, each of which has the form described by Formula (3). $Y_0 \in \mathbb{IR}^n$ is the given initial values.

Some researchers went further by considering situations when the parameters are also intervals, which lead to the replacement of Equation (4) by the following equation:

$$Y'(t) = \mathcal{F}(Y, \theta). \tag{6}$$

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.

Based on the existing numerical algorithms for IVPs, researchers tried to recast these algorithms and make them work with interval arithmetic. Many efforts have been made to develop reliable ODE solvers to deal with interval IVPs for ODEs described by Equations (4) and (5). Early work includes the interval methods for ODEs developed by Moore [39] and Markov & Angelov [36]. There are already several publicly available software packages, such as AWA [34], VNODE [48], and COSY [35], for solving interval IVPs for ODEs. More detailed information about existing methods for solving interval IVPs is given by Nedialkov, Jackson, & Corliss [46], and available software packages are summarised by Nedialkov [47]. In addition, there is one system VSPODE [32] trying to deal with problems described by Equations (6) and (5).

The above mentioned interval ODE solver share many similarities: all of them can be considered as interval versions of traditional numerical simulators. To perform the simulation, all of them except COSY largely rely on the successful calculation of interval Taylor coefficient [46], which requires additional automatic differentiation packages, such as FADBAD++ [56]. While COSY employs a Taylor model integrator which enables it to deal with larger interval initial values [35].

Apart from ODEs, some researchers are interested in DAEs (Differential Algebraic Equation) because in many scientific problems DAEs are a natural description of the underlying dynamic systems and consequently easier to understand compared with ODEs. A set of DAEs is represented by the following equation:

$$F(t, y, y', y'', ..., y^{[m]}) = 0 \tag{7}$$

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 $R^{n \cdot m+1} \to R^n$. Note an implicit ODE has the same form as Equation (7), but the difference is that a DAE cannot always be converted to an equivalent form of Equation (1) by introducing intermediate variables.

The initial values at time t_0 is a solution of DAEs in the form of Equation (7) in the following form:

$$F(t_0, y(t_0), y'(t_0), y''(t_0), ..., y^{[m]}(t_0)) = 0 \tag{8}$$

The IVP for DAEs is to numerically simulate DAEs in the form of Equation (7) given the initial values shown in Equation (8). Similar to the way we represent the interval IVPs for ODEs, the interval IVPs for DAEs can be written according to Equations (7) and (8).

The development of interval DAE solvers also received some attention [49, 21] recently. In all of the existing interval DAE solvers, pre-analysis of the structure of the DAE before simulation is required, for instance, the Pryce's structural analysis [52], which has been used by several interval DAE solvers [21, 5, 49]. In addition to the pre-analysis of the DAE, taylor series (as in DAETS [49]) or taylor models (as in COSY [35]) are required, which in turn require the use of FADBAD++ [56] as in interval ODE solvers.

2.2 From Qualitative Simulation to Semi-quantitative Simulation

In the Qualitative Reasoning (QR) community, researchers initially focused on describing complex dynamic systems at a qualitative level by the use of Qualitative Differential Equations (QDEs) [31]. A QDE is the conjunction of a set of qualitative constraints, which link the *qualitative variables* in the model and express the relations among these variables. Qualitative variables can only take values from their associated *quantity spaces*. A quantity space is composed of several qualitative values, for example, in QSIM [29, 30], a well-known qualitative simulator, qualitative values are landmark values and intervals between two landmark values.

As for the qualitative constraints, there are two kinds: those representing algebraic relations, such as *qualitative addition, substraction, multiplication, and division*, and those representing incomplete knowledge about the function relations, such as M^+ and M^- constraints in QSIM, and *function* constraints in *Morven* [17, 7], a fuzzy qualitative reasoning framework. Table 1 lists some commonly used qualitative constraints in QSIM and their corresponding mathematical relations. In this table variables in the right hand column such as *X(t)* are continuous functions of time *t*. f is a function that is continuously differentiable over its domain (the so-called reasonable functions in QSIM), and f' stands for the first derivative of *f*.

Similarly, Table 2 lists some *Morven* constraints and the corresponding mathematical equations. *Morven* constraints are more flexible: each place in a constraint can represent not only the magnitude, but also arbitrary derivative of a variable. As in FuSim [55], a fuzzy qualitative reasoning algorithm, the *func* constraints in *Morven* are more general than M^+ and M^- constraints: it allows any mapping of possible values for one variable to be consistent with another, and it can also be specialised to represent the same relations as M^+ and M^-.

A QDE is an abstraction of a set of ODEs as described by Equation (1) or DAEs as described by Equation (7) because the function relations of a QDE correspond to an infinite number of quantitative mathematical functions, and the qualitative values assigned to variables in a QDE represent various quantitative values.

A qualitative simulation engine such as QSIM or *Morven* is able to simulate a QDE model and predicate possible *qualitative states* and their transitions. A qualitative state is a complete assignment of all qualitative variables in the system.

QR is suitable for modelling dynamic systems with incomplete knowledge and qualitative measurements. However, if there is more precise yet incomplete quantitative information

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

QSIM Constraints	Mathematical Equations
ADD(X,Y,Z)	$Z(t) = X(t) + Y(t)$
MULT(X,Y,Z)	$Z(t) = Y(t) * X(t)$
DERIV(X,Y)	$dX(t)/dt = Y(t)$
MINUS(X,Y)	$Y(t) = -X(t)$
M^+(X,Y)	$Y(t) = f(X(t)),\ f' > 0$
M^-(X,Y)	$Y(t) = -f(X(t)),\ f' > 0$

Table 2: 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)	$dY(t)/dt = f(X(t))$

available, QR engines could make use of this quantitative information to make more precise predictions.

Some researchers started from qualitative simulation engines such as QSIM and tried to incorporate incomplete quantitative information (in the form of intervals), and this led to the development of *semi-quantitative simulation* algorithms which can communicate with or be integrated into existing QR engines. For instance, Q2 [28] and Q3 [6] are two semi-quantitative simulation systems based on the QSIM formalism.

Q2 allows numerical information, including the ranges of some qualitative landmarks and the bounding envelops of monotonic functions, to be used with QSIM. The provided numerical information can augment qualitative descriptions: the range information about a landmark can be propagated across constraints and thus narrow down the range of other landmarks. One major disadvantage of Q2 is that the time step is very coarse because only qualitative landmarks are considered. So a lot of spurious errors are generated within the intervals calculated.

Q3 extends Q2 by introducing step-size refinement. It achieves this by detecting gaps in the behaviour generated by QSIM and Q2, and then inserts new auxiliary states within this gap. The new auxiliary states are interpolated to help refine the ranges of the states. This is continued until the results are sufficiently precise or until no further narrowing of the ranges can occur. Q3 offers a higher resolution simulation than Q2 by providing more information between qualitative landmarks. One disadvantage of Q3 is its computational complexity: it maintains a complex constraint network, in which quantitative information is propagated throughout the network iteratively by the use of the Waltz algorithm [60]. As this constraint network is composed of all constraints instantiated from all states, the computational cost might significantly increase with the increase of the number of newly created states.

NSIM [25] and SQSIM [26] are two later QSIM-based semi-quantitative simulation systems. NSIM aimed to create a simulation engine which could make use of any numerical knowledge in addition to the fully qualitative models. Apart from QDEs, it also used Structural Differential Equations (SDEs) and Semi-Quantitative Differential Equations (SQDEs) to describe models at different levels of abstraction. NSIM evaluates the upper and lower bounds of each constraint to predict the behaviours, which makes it unsuitable for models containing non-monotonic functions. Interval arithmetic is used to propagate these bounds which results in widening of the intervals. SQSIM makes use of the semi-quantitative simulation to refine the behaviour tree of the qualitative simulation thus reducing the number of spurious behaviours generated. It combines the inferences made by QSIM, Q2 and NSIM and thus reduces the imprecision in the predictions made by each.

NIS (Numerical Interval Simulation) [59] is closer to the systems developed by the numerical simulation community. At each time step t_n, NIS calculates the extremal values of all derivatives by interval arithmetic. These extremal values are then used for calculating the variable values for the next time step by an interval version of the Euler or Runge-Kutta method. The authors claimed that NIS produced tighter simulations than early versions of the interval ODE solver such as Moore's approach [39]. However, there is no comparison between NIS and later interval ODE solvers such as AWA [34] and VNODE [48]. In addition, we note that there is a limitation in NIS: in order to generate a complete enclosure, the initial intervals of variables must be limited when these variables are arguments of some non-monotonic functions so that the intervals do not spread across the maximum or minimum peaks of the non-monotonic functions.

3 Motivations of the Research

From Section 2 we can see that interval simulation of differential equations is an active research area and there are many systems being developed to make more robust and precise simulation. We also recognise that researchers from the NS and QR communities take different approaches to the interval simulation, and as mentioned in Section 1 there is a gap between these two communities in terms of the methodology: on one hand, NS researchers do not make use of qualitative knowledge obtained from reasoning. On the other hand, QR researchers tend not to be fully aware of the latest development of interval ODE/DAE solvers and more importantly, do not make better use of existing well-established integration techniques developed within the NS community.

Consequently, as stated in Section 1, from the QR community point of view, there is room for improvement in the development of interval simulators: first, we acknowledge the existence of various integration techniques developed within the NS community. However, as far as the authors are aware, up till now only the Euler and Runge-Kutta methods have been used by QR researchers. So one of the motivations of this report is to explore the potential of other integration techniques, such as the Taylor series [2] and the Adams family [4], applied to QR-based interval simulation.

Second, apart from the integration techniques, there are two approaches to performing the simulation: constructive and non-constructive. The distinction between these two kinds of simulation was first pointed out by Wiegand [63] and also discussed in detail by Coghill and Chantler [16] within the QR community. Constructive simulation requires that derivatives of variables are given in explicit forms. An explicit form of the derivative of a variable, say, y', is a mathematical equation which is consistent with the model and in this equation y' appears only once and it is the only component on the left-hand side of this equation.

The explicit form of a derivative can be obtained directly from the model, for instance, the first derivatives of all variables in the ODEs described by Equations (4) and (6) are explicitly given; the explicit form of a derivative can also be obtained by equation solving if the model is given in an implicit form. The explicit forms of higher derivatives which do not appear in the model can be derived by automatic differentiation packages, such as FADBAD++ [56].

In constructive simulation, at time point t_i, the derivatives are first calculated directly from their explicit forms, and then used for the calculation (estimation) of the magnitudes of variables at the next time point t_{i+1}. Constructive simulation has been largely used in numerical simulation and proved its validity, but one limitation is that when models are given as implicit forms, an additional operation has to be performed to obtain the explicit forms, and the computational cost of this procedure might be expensive. In addition, to solve a DAE constructively, the structure analysis might fail in some ill-structured DAEs [52] and in this situation it is very hard to obtain the explicit forms of the derivatives. Furthermore, it is not straightforward for constructive simulation to deal with models containing algebraic loops, which can occur when modelling some electrical or mechanical systems [15].

Within the QR community there are a number of qualitative simulators performing simulation in a non-constructive manner, such as QSIM and FuSim [55]. For instance, in QSIM the core idea is as follows: at the current simulation step, given a QDE model as described in Section 2.2 and a qualitative state (also described in Section 2.2) at the immediately previous simulation step, the QSIM algorithm first considers each constraint of the QDE model and generates all tuples (pairs or triples) of possible values for the variables of this constraint. Then the tuples which are not consistent with this constraint are eliminated. After this operation, each constraint is associated with a set of tuples of values. Finally, all possible qualitative states at the current simulation time step are created by an exhaustive backtracking search on all combinations of these tuples.

From the QSIM example we see that in general, non-constructive simulation essentially employs a generate-and-eliminate strategy: all possible variable values (e.g., qualitative landmarks or intervals in QSIM) are first generated, then these values are filtered or further refined according to model constraints. One of the advantages of non-constructive simulation is that it is a straightforward approach which does not require the explicit forms of variables or their derivatives. This is particularly effecitive when dealing with some ill-structured DAEs or DAEs containing algebraic loops. Non-constructive simulation does not need pre-analysis or pre-processing for the model, although the generate-and-eliminate strategy sometimes may be more computationally expensive than constructive simulation.

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, each of which corresponds to a mathematical relation of some variables in the model. For instance, as will be shown later, the model described by Equations (9) and (10) can be converted into a *Morven* model listed in Table 3. 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 report, that is, to implement a simulator that can perform interval simulation in a non-constructive manner. We expect that the research into the non-constructive simulation approach will contribute to both the NS and QR communities.

Finally, we intend to develop our interval simulation algorithm within the *Morven* framework [17] (formerly known as the *Mycroft* framework) so that we can make the *Morven* framework able to perform a full spectrum of simulation: qualitative, semi-quantitative, and quantitative. Furthermore, we expect that the interval simulation algorithm can benefit from the features provided by the *Morven* framework, including the vector variable and multiple differential planes which will be described in Section 4.

In summary, the goal of this research is to develop a novel non-constructive interval simulation algorithm that employs appropriate integration techniques and is seamlessly integrated into the *Morven* framework.

4 The *Morven* Framework

In this research we use the *Morven* formalism to represent semi-quantitative models. In fact, *Morven* is implemented as a general simulation framework which is able to represent and simulate models of different levels of abstractions: from qualitative to quantitative.

The *Morven* framework [17] is a constraint-based fuzzy qualitative system, and its development is largely based on FuSim [55], the Predictive Algorithm (PA) [63], and Vector Envisionment (VE) [43].

As in PA, qualitative constraints in a *Morven* model are distributed over multiple *differential planes*. The *0th* differential plane contains the constraints, which can represent a model which has the same form as used for numerical simulation. The constraints in a higher differential plane are obtained by differentiating the corresponding constraints in the preceding differential plane.

As in VE, qualitative variables in *Morven* are in the form of variable length vectors. The first element in the vector is the magnitude of the variable, the *ith* ($i > 1$) element is the *(i-1)th* derivative. The modeller can include as many derivatives as necessary.

As in FuSim, qualitative variables in *Morven* take their values from fuzzy quantity spaces, which are composed of fuzzy numbers in the form of fuzzy four-tuples [55]. *Morven* also employs the same fuzzy arithmetic operations as defined in FuSim to calculate the algebraic constraints using fuzzy quantity spaces. However, in this research as we focus on interval simulation, the fuzzy mechanism is not considered, and consequently fuzzy numbers degenerate into interval numbers in the form described by Equation (3). Accordingly the fuzzy arithmetic operations become interval arithmetic ones.

Any ODE or DAE model can be converted into a *Morven* model, and we use the single tank system shown in Figure 1 as an example to demonstrate how *Morven* is used to represent models. The quantitative model for a linear version of this system is as follows:

$$q_o = kV, \tag{9}$$

$$dV/dt = q_i - q_o, \tag{10}$$

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 determined by the cross sectional area of the tank and the density of the liquid.

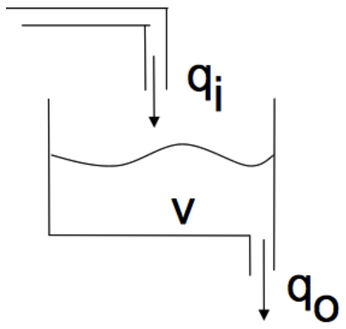


Figure 1: The Single Tank System

Table 3: 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)	($V'' = q_i' - q_o'$)

The above ODE model can be converted into a *Morven* model which is shown in Table 3. This model is composed of four constraints, *C1* to *C4*, and the corresponding quantitative relation for each constraint is shown on the right hand side in the brackets. Here the label *dt* means *derivative*, and the integer immediately following it indicates which derivative of the variable (0 means the magnitude). For variable V, the magnitude, the first and second derivatives are used; for variable q_o and q_i, only the magnitude and the first derivative are used.

The primitive *sub* in Constraints *C2* and *C4* stands for the subtraction relation. The primitive *mul* in Constraint *C1* and *C3* stands for multiplication. It may be noted that in this model the parameter k is treated as an exogenous variable, whose value remains constant.

The *Morven* model given in Table 3 can be quantitative, semi-quantitative, and qualitative depending on how the values of variables are assigned. For instance, if all variables (including their magnitudes and derivatives) have to take values from the signs quantity space, which is shown in Table 4, this *Morven* model is qualitative; if all variables take interval values instead of from quantity spaces, the model becomes semi-quantitative; and if all variables take real numbers, the model is quantitative. The *Morven* framework is designed to reason about its models using different algorithms according to different situations.

The model in Table 3 does not contain *func* constraints as mentioned in Table 2. To represent a more general form of the single tank system, we can replace Constraints *C1* and *C3* with the following two function constraints:

$C1'$: func (dt 0 q_o, dt 0 V),

$C3'$: func (dt 1 q_o, dt 1 V).

After the above replacement, the model can describe not only linear but also non-linear single tank systems through the specification of corresponding mappings for the function

Table 4: The Signs Quantity Space

Quantity	Range
negative(-)	$(-\infty, \quad 0)$
zero(0)	0
positive(+)	$(0, \quad \infty)$

Table 5: Function Mappings Under the Signs Quantity Space

func (A,B)	negative	zero	positive
negative	1	0	0
zero	0	1	0
positive	0	0	1

constraints. For instance, if all variables take the signs quantity space as shown in Table 4, and the mappings of the *func* in the constraints $C1'$ and $C3'$ are given in Table 5, in which "1" stands for the existence of a mapping between variables A and B and "0" otherwise, the model can represent a class of the single tank systems sharing the same qualitative behaviour. It is also pointed out that in this situation this model can only be simulated qualitatively.

5 Non-constructive Interval Simulation

As mentioned in Section 3 we aim to develop non-constructive interval algorithms as a sub-component of the *Morven* framework. More specifically, the resulting simulation algorithms will form part of *JMorven* [7, 8], a Java implementation of the *Morven* framework.

As discussed by Coghill and Chantler [16], in order to perform a non-constructive qualitative simulation, two phases are needed: Transition Analysis (TA) and Qualitative Analysis (QA), the names of which are first coined by Williams [64]. In the TA phase all possible qualitative values of variables are generated. The QA phase is applied to the model after TA, that is, several filters are used to filter out the impossible assignments of values to variables according to model constraints and global consistency.

In the context of interval simulation, the TA phase corresponds to the interval integration and QA corresponds to the further refinement of interval values of variables. This means the interval integration and interval refinement are two most important components of non-constructive interval simulation.

The rest of this section is organised as follows: in Section 5.1, we give the interval arithmetic used in this research. This is followed by the description of interval integration methods in Section 5.2 and interval narrowing algorithm presented in Section 5.3. Finally in Section 5.4 we describe in detail several simulation modes provided by the simulation algorithm.

5.1 Interval Arithmetic Used in the Interval Simulation

The interval arithmetic used in our algorithm is defined in Table 6. In this table we consider two situations: (1) when the two intervals are the same; and (2) when the intervals span zero.

The reason for including these two situations and making a more complicated definition of the interval arithmetic operation is for the ease of implementation and more accurate and efficient calculation during non-constructive simulation. The final two rows of Table 6 define the intersection and union of two intervals.

Table 6: 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 variable 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 for Non-constructive Interval simulation

As discussed in Section 2.1 there are many numerical integration methods available, and some of them have been recast as interval integration methods. However, all of these existing methods are based on constructive simulation. Our first step is to investigate which methods can be used in non-constructive simulation.

It is pointed out that the *Morven* models can represent both ODE and DAE models, as described by Equations (1) and (7), respectively. This means that in a *Morven* model the explicit forms of all derivatives of variables cannot always be obtained straightforwardly because this *Morven* model may represent a DAE model. This fact restricts the use of many existing integration methods for non-constructive simulation.

Firstly, we investigate the Euler methods [3], the simplest integration techniques. There are two streams of Euler methods: the forward and backward ones. The forward Euler method for constructive simulation is given as follows:

$$y_{n+1} = y_n + hf(y_n) \tag{11}$$

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 (11), 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 pervious calculation at time step t_n, and replace $f(y_n)$ in Equation (11), which is shown below:

$$y_{n+1} = y_n + hy'_n \tag{12}$$

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:

$$y_{n+1} = y_n + hf(y_{n+1}), \tag{13}$$

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, the fixed point iteration method [9] can be used: first guess an initial value of y_{n+1}, say, y^0_{n+1}, and use this initial guess to calculate $f(y^0_{n+1})$. The newly obtained $f(y^0_{n+1})$ is substituted into Equation (13) to produce a new value of y_{n+1}, that is, y^1_{n+1}. This process is executed repeatedly until there is no more change in the value of y_{n+1}.

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.

Secondly, we consider the well-established and most commonly used Runge-Kutta methods [12] in NS. The Runge-Kutta methods require interaction between the variables of the model in a similar way to Moore's interacting approach [38]. For instance, consider the common fourth-order Runge-Kutta method as defined by the following equations [50]:

$$\begin{aligned} k_1 &= hf(t_n, y_n) \\ k_2 &= hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 &= hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 &= hf(t_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \end{aligned}$$

In the above equations it is not possible to calculate the values of k_2, k_3, and k_4 at time t_n in non-constructive simulation because function f is not known.

Thirdly, we investigate whether the Taylor Series Expansion [2], of which the forward Euler method is equivalent to a first order, can be used in non-constructive simulation. Taylor Series Expansion uses the current values of derivatives at time t_n to estimate the values at the next time step t_{n+1}, as shown below:

$$y_{n+1} = y_n + y'_n \cdot h + \frac{y''_n}{2!} \cdot h^2 + ... + \frac{(y_n)^m}{n!} \cdot h^m + ... \tag{14}$$

In the above y is a function of time t: $y_t = f(t)$. So we have $y_{n+1} = f(t_{n+1})$ and $y_n = f(t_n)$. h is the step size ($h = y_{n+1} - y_n$), and y'_n, y''_n, ..., $(y_n)^m$ are derivatives of y_n. From Equation (14) we see that if the derivatives of y can be obtained at time step t_n, the value of y at time step t_{n+1} can be estimated, and the more derivatives are specified, the more accurate simulation can be achieved. More importantly, Taylor Series is suited to non-constructive simulation since each variable can be integrated individually. In addition, as we use the *Morven* framework, derivatives of variables can be easily obtained from multiple differential planes.

Fourthly, we consider the linear multi-step methods [10]. This kind of integration method makes use of values of the first derivative calculated at two or more previous time points to estimate the current value. We first start from the Adams-Bashforth (AB) methods [11], and the simplest two-step AB method is shown below:

$$y_{n+2} = y_{n+1} + \frac{3}{2}hy'_{n+1} - \frac{1}{2}hy'_n \tag{15}$$

In the above at time t_{n+2}, the values of y_{n+1}, y'_{n+1} and y'_n are used to estimate the value of y_{n+2}. This is also feasible in non-constructive simulation as the values needed for estimation have already been obtained at previous time points. Similarly, the multi-step AB methods [11] can also be used, for instance, the three-step AB method is as follows:

$$y_{n+3} = y_{n+2} + \frac{23}{12}hy'_{n+2} - \frac{4}{3}hy'_{n+1} + \frac{5}{12}hy'_n \tag{16}$$

The other two major linear multi-step methods: Adams-Moulton and Backward Differentiation Formulas (BDF) [22] require the explicit forms to calculate the values of the first derivative at the current time point, which is similar as in the backward Euler. So these two methods are again not suitable for non-constructive simulation.

Lastly, we consider predictor-corrector methods [51], examples of which include the Euler Trapezoidal method and the Adams-Bathforth-Moulton method [37]. These methods are not suitable for non-constructive simulation because the "corrector" always requires the explicit form of the derivatives. For instance, consider the Euler Trapezoidal method:

$$\textit{Predictor:} \quad y^p_{n+1} = y_n + hy'_n \tag{17}$$

$$\textit{Corrector:} \quad y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n), f(t_{n+1}, y^p_{n+1})). \tag{18}$$

In this method the first equation is the forward Euler and used as a "predictor" to guess an initial value of y_{n+1}, namely y^p_{n+1}. Then the second equation is used as a "corrector". In the corrector equation the explicit form of function f is needed so that the initial guess y^p_{n+1} can be used to further correct the value y_{n+1}.

In summary, after investigating most commonly used integration methods in numerical simulation, we can say that Taylor Series (with the forward Euler method being a special case) and AB methods can be used in non-constructive simulation.

Finally, the Taylor Series and AB methods can be easily recast for interval simulation by using the interval mathematics defined in Section 5.1. In this research these two integration approaches will be used throughout all experiments.

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 should be able to control the growth of intervals during simulation. This can be achieved by iteratively applying the model constraints to intervals.

We first propose the Inverse Constraint Operations, which are detailed as follows: 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 report. For example, consider the following constraint:

$$A = B + C, \tag{19}$$

its inverses will be the following two:

$$B = A - C, \tag{20}$$

$$C = A - B. \tag{21}$$

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 6 to Equation (19), we can determine the range of A by $(B + C)$ as shown below:

$$[5.5,\ 7.5] = [3,\ 4] + [2.5,\ 3.5]. \tag{22}$$

Consider the initial interval $A = [5,\ 6]$ from integration, the intersection of these two intervals will be:

$$A = [5,\ 6] \cap [5.5,\ 7.5] = [5.5,\ 6]. \tag{23}$$

To narrow the rest of the variables the inverse constraints should be used: according to Equation (20), the interval for B should be as follows:

$$[2,\ 3.5] = [5.5,\ 6] - [2.5,\ 3.5].$$

Again from integration, $B = [3,\ 4]$. Taking the intersection of the intervals gives $B = [3,\ 3.5]$. Similarly, using Equation (21), the interval for C is shown below:

$$[2,\ 3] = [5.5,\ 6] - [3,\ 3.5],$$

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 (19) (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 [60] applied to interval values. In particular, the soundness and completeness of the Waltz algorithm applied to interval refinement has been extensively studied by Davis [18]. It is noted that in Q3 [6] the Waltz algorithm was also used to refine interval values, as mentioned in

Section 2.2. However, the difference is that in Q3 the Waltz algorithm was applied 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 (6), 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 [33] in the NS community, which treats the time-invariant interval parameters as independent state variables and thus increases the complexity of the simulation.

5.4 Modes of Simulation

Thus far we have: presented the interval arithmetic in Section 5.1, proposed two integration methods in Section 5.2, and described the iterative interval narrowing algorithm in Section 5.3. Equipped with these three modules, we are able to perform non-constructive interval simulation.

However, a common problem in interval simulation is that the intervals begin to widen and then eventually become uncontrollable during simulation. This is especially the case when initial intervals are set to be very large. (In the NS community, this problem is called the "wrapping effect", as first discussed by Moore [38]). The reason for this is that the interval arithmetic has to bound all the correct solutions in the intervals, which makes it a complete method, but during the interval calculation the spurious behaviours (those values that the system cannot actually achieve) are also bounded in the intervals, which means it is not a sound method. So in this research we offer different approaches to deal with the interval widening problem, and each approach is termed a *simulation mode* in this report. 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.

In this section, we first describe the simplest BIS mode. Then we present other simulation modes, each of which is based on the BIS mode but takes additional process to deal with the initial intervals.

5.4.1 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 addition, as the other more advanced simulation modes are all based on BIS but make some pre-processing for the initial intervals, the description of BIS also forms part of the description of these simulation modes.

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:

$$num = \frac{t_{tot}}{\delta t}.$$

At the initial time step t_0, given an incomplete initial state, which specifies the initial intervals for some variables of the model, 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 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 shown in Equation (14), 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 shown in Equations (15) and (16), 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. For each iteration, after accessing a constraint C (C could be either the original constraint or the inverse of a constraint in the model) a new interval is calculated, an example of which has been shown in Equation (22). Then this new interval is intersected with the relevant interval obtained from integration, an example of which has been given in Equation (23). The resulting interval will then be used for checking against the next constraint.

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.

5.4.2 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 report. 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.

The above idea is formally, and more precisely, described as follows: if we use IR_i (i=0, 1, ..., $m-1$) to represent all initial intervals, where m is the number of intervals, and the sub-interval of each IR_i is represented as $SIR_{i,j}$ (j=0, 1, ..., $n-1$), where n is the number of sub-intervals for each IR_i, then for any two integers $k, p \in (0, 1, ..., n-1)$, $SIR_{m,k} \cap$

$SIR_{m,p} = \emptyset$, and the union of all the sub-intervals will cover exactly the original interval: $\bigcup_{k=0}^{n-1} SIR_{i,k} = IR_i, i \in (0, 1, ..., m-1)$. Each initial state IS will be represented by the following combination of intervals: $< SIR_{0,i}, SIR_{1,j}, SIR_{2,k}, ..., SIR_{m,b} >$, and there will be n^m initial states generated for simulation.

After simulation, the final results will be the union of all these n^m simulations. That is, for each element in the final repository R, $< Var, Der, [a, b] : t_i >$, the interval $[a, b] = \cup_{k=0}^{n^m-1} [l_k, d_k]$, where $[l_k, d_k]$ is the resulting interval for each individual simulation, and the interval union operator "$\cup$" is defined in Table 6.

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, which makes the simulation less expensive. This is illustrated as the following simple example: Suppose in a model there is a constraint $A = B + C$, and the initial condition is A=$[3, 5]$, $B = [1, 2]$, and $C = [2, 3]$. This initial condition is consistent with the constraint, but suppose in the sub-interval simulation each interval is divided into 200 sub-intervals, and the following initial condition A=$[3, 3.01]$, $B = [1.995, 2]$, and $C = [2.995, 3]$ will not be consistent with this constraint.

5.4.3 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 [53]. 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 report.

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.

5.4.4 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. The motivation behind this is that the point method guarantees no unnecessary divergence and using a set of points for each interval should cover most of the possibilities for the final solution. 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.

5.4.5 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. If the boundaries of intervals are critical to the outcome of predictions, such as in the fault diagnosis problems [24], complete methods such as the Basic Interval and Sub-interval Simulation should be used. If we are more interested in revealing the behaviours of the dynamic system under initial states with interval values, for instance, the evolution of intervals along with time, and reasonable approximations of boundaries are acceptable, we can make use of the incomplete methods, which are more computationally efficient.

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.

6.1 Completeness and Soundness

The completeness means that the interval simulation algorithm must bound all real solutions. In another word, suppose at time point n, $\tilde{Y_n} \in \mathbb{IR}$ and $Y_n \in \mathbb{IR}$ are the actual solution and simulated result of the model, respectively, if there exists an interval $Y_o \in \mathbb{IR}$ such that $Y_o \subseteq \tilde{Y_n}$, but $Y_o \nsubseteq Y_n$, we say this algorithm is not complete. The soundness means that the simulation results should be a subset of the actual solution. Using the same notation, if there exists an interval Y_o such that $Y_o \subseteq Y_n$, but $Y_o \nsubseteq \tilde{Y_n}$, we say this algorithm is not sound. As for the completeness, we present the following two theorems:

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

Proof. First, the interval arithmetic shown in Table 6 is reasonably defined, and it is consistent with that defined by Moore [40]. As both the Taylor Series and AB methods use this well-defined interval arithmetic, no actual solution will be excluded from the simulation results after each integration step. Second, the interval narrowing algorithm is complete because of the completeness of the Waltz algorithm as discussed by Davis [18]. Consequently, the whole simulation algorithm under the BIS mode is complete.

□

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

Proof. We give a simplified and illustrative proof for this theorem as follows: suppose in the model there are only two variables taking interval values, and these two intervals are denoted A and B. First we consider A and B as two uncountable sets, each of which is composed of an infinite number of data points. The Cartesian product [61] of A and B is as follows:
$A \times B = \{< a, b > | a \in A, b \in B\}$.

The Basic Interval Simulation considers all the data points included in $A \times B$ as initial conditions. Now we divide sets A and B into two non-overlapping subsets:
$A = A_1 \bigcup A_2$, $A_1 \bigcap A_2 = \emptyset$,
$B = B_1 \bigcup B_2$, $B_1 \bigcap B_2 = \emptyset$.

Note the above $\bigcup$ and $\bigcap$ are set operations rather than interval operations. As the Cartesian product distributes over union [62], which is shown below:

$$X \times (Y \bigcup Z) = (X \times Y) \bigcup (X \times Z),$$

where X, Y, Z are three sets, we have the following equation:

$$\begin{aligned} A \times B &= (A \times B_1) \bigcup (A \times B_2) \\ &= ((A_1 \bigcup A_2) \times B_1) \bigcup ((A_1 \bigcup A_2) \times B_2) \\ &= (A_1 \times B_1) \bigcup (A_1 \times B_2) \bigcup (A_2 \times B_1) \bigcup (A_2 \times B_2) \end{aligned}$$

In the above equation set $A \times B$ includes and only includes all the data points considered by the Basic Interval Simulation and the rightmost expression is the set considered by the Sub-interval Simulation. So the above equation means that the Sub-interval Simulations and the Basic Interval Simulations actually take the same set of data points as initial conditions for simulation, although this set is infinite and uncountable. Similarly, for the situation when there are more than two intervals and each interval is divided into more than two subintervals, we can still reach the same conclusion. Since the Basic Interval Simulation is complete according to Theorem 1, the Sub-interval Simulation is also complete. □

As for the soundness of BIS and SIS, as mentioned in Section 5.4, BIS is not sound because of the wrapping effect, which is intrinsic to interval arithematic. Similarly, SIS is also not sound, but theoretically speaking, SIS is sound if the sub-interval approaches to zero. 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.*

This is because in numerical simulation, all intervals equal zero, which eliminate the wrapping effect of interval arithmetic, and no spurious behaviours will be bounded. This means non-constructive simulation is sound. In addition, numerical simulation is a special case of non-constructive interval simulation when all intervals have zero widths. According to Theorem 1, the simulation is still complete.

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

Proof. We start with a first order differential equation:

$$\mathcal{F}(t, Y, Y') = 0, \tag{24}$$

where $\mathcal{F}$ is an interval valued function, Y is a variable depending on time t, and Y' is its first derivative. Let Y_0 be the interval initial condition: $Y_0 = [\underline{Y_0}, \overline{Y_0}]$, where $\underline{Y_0}$, $\overline{Y_0} \in \mathbb{R}$.

Suppose the corresponding real valued version of Equation (24) is $F(t, y, y') = 0$, and the initial condition is given as $y_0 \in Y_0$. The particular solution of this real-valued differential equation is in the following form: $y = f(y_0, t)$, although the form of real-valued function f might be unknown.

Given any two real number initial conditions y_{l0}, $y_{h0} \in Y_0$, where $y_{l0} < y_{h0}$, we use the BIS mode to perform the simulation. As the non-constructive numerical simulation is complete and sound according to Lemma 1, at any time point t_0, we have the following two equations:

$$\hat{y_l} = f(y_{l0}, t_0) \in \hat{Y}, \tag{25}$$

$$\hat{y_h} = f(y_{h0}, t_0) \in \hat{Y}, \tag{26}$$

where $\hat{y_l}$ and $\hat{y_h}$ are the simulated results, and $\hat{Y}$ is the actual solution of Equation (24) given Y_0 at time t_0. It is noted that in Equations (25) and (26) we use "=", which means we assume that the rounding and truncation errors [57] caused by the numerical simulation are small enough to be ignored.

Let $Y_p = [min(\hat{y_l}, \hat{y_h}), max(\hat{y_l}, \hat{y_h})]$. Apparently Y_p is the interval predicted by the simulation using two points y_{l0} and y_{h0}. Since $\hat{y_l}, \hat{y_h} \in \hat{Y}$, we have $Y_p \subseteq \hat{Y}$. Thus the Point Simulation mode is sound.

Now we will prove that Point Simulation modes are not complete. Given a fixed time point t_0, $f(y_0, t_0)$ in Equations (25) and (26) is a function of real-valued initial condition y_0. However, this function might not be monotonically increasing or decreasing. In the Extreme Point Simulation mode, where $y_{l0} = \underline{Y_0}$ and $y_{h0} = \overline{Y_0}$, Y_p does not always equal to $\hat{Y}$. This means the Extreme Point Simulation model is not complete. Similarly, for Regular-spaced and Monte-Carlo point simulation when $y_{l0} \geq \underline{Y_0}$ and $y_{h0} \leq \overline{Y_0}$, we can reach the same conclusion.

Finally, we can further extend the above conclusion to a system of first order differential equations, in which Y in Equation (24) is a vector variable. Accordingly all other variables and values (such as Y_0, y_0, $\hat{y_l}$, and $\hat{y_h}$) and functions (such as $\mathcal{F}$, F, and f) will also be in the vector form. Furthermore, as any higher order system can be described as a system of first order equations by introducing intermediate variables, the conclusion also applies to higher order systems. □

Based on the above presented theorems, in Table 7 we give a summary of all the simulation modes in terms of their completeness and soundness.

Table 7: Summary of Simulation Approaches

Approach	Sound?	Complete?	Comments
Basic Interval	N	Y	Rapid interval divergence occurs
Sub-Interval	Y*	Y	Less divergence but computationally expensive
Monte-Carlo Interval	N	Y**	Still suffers from divergence
Extreme Point	Y	N	No divergence but doesn't cover all real solutions
Regular-spaced Point	Y	Y**	No divergence, close to covering all real solutions but slow
Monte-Carlo Point	Y	Y**	No divergence, close to covering all real solutions most efficiently

* Soundness is achieved as the interval width tends to zero, i.e., as the number of intervals tend toward infinity.
** Monte-Carlo methods achieve completeness as the number of iterations tend toward infinity. The Regular-spaced method achieves this as the number of points tend toward infinity.
In practice, tending toward infinity is not required; merely tending toward the resolution of the floating point representation is required although this would still result in too many iterations to complete in a reasonable amount of time.

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 [6], who are in turn inspired by Moore [39] and proofs of convergence and stability in numerical simulation [20]. 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 $h \to 0$ stability [6] 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 given by Berleant and Kuipers [6].

Lemma 2. *Consider the first order differential equation described by Equation (24) with a given initial condition Y_0, assume its explicit form is as follows:*

$$Y' = F(Y), \tag{27}$$

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 6. 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*

$$|Y_n| \leq |Y_0| + Kh \tag{28}$$

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.

Proof. Step (1): Let $M \equiv [lo, hi]$, then at any time step i, the following statement is correct:

$$Y_i' \subseteq F(M). \tag{29}$$

This is because given Y_i at time step i, where $i = 0, \cdot\cdot, N$, and N is the maximum integration step, by using the integration narrowing algorithm, $Y_i' = F(Y_i)$.

If $Y_i' \nsubseteq F(M)$, we will have $Y_i \nsubseteq M$ (recall that $F(A) \subseteq F(B)$ if $A \subseteq B$ and F is *inclusion monotonic* [41]).

However, as we know that $Y_i \subseteq M$, which contradicts the above statement. Consequently $Y_i' \subseteq F(M)$.

Step (2): As we use the forward Euler method, we have the following equation:

$$Y_n = Y_{n-1} + hY_n' \tag{30}$$

The above equation combined with Statement (29) gives us:

$$Y_n \subseteq Y_{n-1} + hF(M). \tag{31}$$

Step (3): F is an interval valued function and it satisfies the Lipschitz property for interval functions [42]:

$$|F(M)| \leq L|M|. \tag{32}$$

In the above L is the Lipschitz constant. Apply the operator $|\cdot|$ to (31),

$$|Y_n| \leq |Y_{n-1} + hF(M)| \leq |Y_{n-1}| + h|F(M)|. \tag{33}$$

Substitute (32) into (33), we have

$$|Y_n| \leq |Y_{n-1}| + hL|M|. \tag{34}$$

Let $m = |M|$, and recursively apply (34), we have

$$|Y_n| \leq |Y_0| + nLmh. \tag{35}$$

Apparently, given a fixed n, $K = nLm$ is a constant. Thus we reach the lemma statement (28).

$\square$

Lemma 2 considers the simulation when the Euler method is used. Similarly, if the two-step AB method is used, we have the following lemma:

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 (28).*

Proof. We take the two-step AB method as an example to proof this lemma. The proof is similar to that for Lemma 2. To prove this lemma, Equation (30) will be replaced by the following equation:

$$Y_n = Y_{n-1} + \frac{3}{2}hY_{n-1}' - \frac{1}{2}hY_{n-2}' \tag{36}$$

Apply the operator $|\cdot|$ to (36), we have

$$\begin{aligned} |Y_n| &= |Y_{n-1} + \frac{3}{2}hY'_{n-1} - \frac{1}{2}hY'_{n-2}| \\ &\leq |Y_{n-1}| + |\frac{3}{2}hY'_{n-1}| + |-\frac{1}{2}hY'_{n-2}| \end{aligned} \tag{37}$$

The above equation combined with (29) and (32) gives us

$$\begin{aligned} |Y_n| &\leq |Y_{n-1}| + \frac{1}{2}h(3|F(M)| + |-F(M)|) \\ &= |Y_{n-1}| + 2h|F(M)| \\ &\leq |Y_{n-1}| + 2hL|M| \end{aligned} \tag{38}$$

Let $m = |M|$, and recursively apply (38), we have

$$|Y_n| \leq |Y_0| + 2nLmh. \tag{39}$$

In the above, let $K = 2nLm$, then we reach statement (28). For multistep AB methods the proof can be done in a similar way. $\square$

Lemma 4. *In Lemma 2 if the Taylor method is used, we can still have statement (28).*

Proof. Assume in the model two differential planes are used, which means from Equation (27), we can get

$$Y'' = G(Y'), \tag{40}$$

where $G = F'$. Similar to Formular (32) in Lemma 2, for function G we have

$$|G(F(M))| \leq L_G|F(M)|, \tag{41}$$

where L_G is the Lipschitz constant. According to the Taylor method,

$$Y_n = Y_{n-1} + hY'_{n-1} + \frac{h^2}{2}Y''_{n-1}. \tag{42}$$

Apply $|\cdot|$ to the above,

$$\begin{aligned} |Y_n| &= |Y_{n-1} + hY'_{n-1} + \frac{h^2}{2!}Y''_{n-1}| \\ &\leq |Y_{n-1}| + h|Y'_{n-1}| + \frac{h^2}{2!}|Y''_{n-1}| \\ &\leq |Y_{n-1}| + h|F(M)| + \frac{h^2}{2!}|G(F(M))| \\ &\leq |Y_{n-1}| + hL|M| + \frac{h^2}{2!}L_G|F(M)| \end{aligned} \tag{43}$$

Let $m = |M|$, $N = max(|F(M)|)$, which means N is the maximum value of of $|F(M)|$, we have

$$|Y_n| \leq |Y_{n-1}| + hLm + \frac{h^2}{2!}L_GN \tag{44}$$

Recursively apply (44):

$$\begin{aligned}|Y_n| &\le |Y_0| + nhLm + n\frac{h^2}{2!}L_G N \\ &= |Y_0| + h(nLm + \frac{1}{2}nhL_G N) \end{aligned} \tag{45}$$

Let $K = nLm + \frac{1}{2}nhL_G N$. For any fixed simulation time $b = nh$, K is a constant. Consequently we reach statement (28). Finally for higher order Taylor methods the proof can be done in a similar way.

□

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:*

$$\vec{\mathcal{F}}(t, \mathbf{Y}, \mathbf{Y}') = 0 \tag{46}$$

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 (46) to obtain the explicit form as follows:

$$\mathbf{Y}' = \mathbf{F}(\mathbf{Y}), \tag{47}$$

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 6. 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*

$$||\mathbf{Y_n}|| \le ||\mathbf{Y_0}|| + Kh \tag{48}$$

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.

Proof. This theorem is a vector extension of Lemma 2 $\sim$ 4. In these lemmas, the operator $|\cdot|$ is replaced by $||\cdot||$; all variables are replaced by their corresponding vector forms; and the interval valued functions are replaced by vectors of interval valued functions. In this way we can go through all these lemmas and prove this theorem. □

It is noted that Theorem 4 can be extended to higher order systems as any such systems can be reduced to first order systems by introducing intermediate variables. From statement (48), we see that given a precise initial condition $||\mathbf{Y_0}|| = 0$, for any fixed simulation time $t = nh$, when $h \to 0$, we will have $||\mathbf{Y_n}|| \to 0$. This means that the algorithm is *convergent* when the simulation step approaches zero. According to the $h \to 0$ stability defined by Berleant and Kuipers [6] and shown below:

$$||\mathbf{Y_n}|| \le K||\mathbf{Y_0}||, \tag{49}$$

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

In Theorem 4 we assume that (46) can be solved to obtain its explicit form (47). This actually indicates that the underlying model is a system of ODEs. As we know that (46) could also be a system of DAEs, we given the following theorem:

Theorem 5. *(Convergence and Stability for a system of DAEs) Suppose by solving (46), we can only obtain the following form:*

$$\mathbf{X}' = \mathbf{F}(\mathbf{X}, \mathbf{Z}), \tag{50}$$

$$\mathbf{0} = \mathbf{G}(\mathbf{X}, \mathbf{Z}). \tag{51}$$

In the above $\mathbf{F}$ *and* $\mathbf{G}$ *are two vectors of interval valued functions. Vector* $\mathbf{Y}$ *in (46) 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*

$$||\mathbf{X_n}|| \leq ||\mathbf{X_0}|| + K_1 h \tag{52}$$

$$||\mathbf{Z_n}|| \leq K_2 ||\mathbf{Z_0}|| + K_3 h \tag{53}$$

Proof. Assume the explicit form of (51) is

$$\mathbf{Z} = \mathbf{G_1}(\mathbf{X}). \tag{54}$$

In the above $\mathbf{G_1}$ is a function vector defined when each $Y_j(t) \subseteq [lo, hi]$, and it is noted that $\mathbf{G_1}$ cannot always be explicitly obtained depending on the features of the function vector $\mathbf{G}$ (recall that the conditions for the existence of the explicit form (54) for the implicit equation (51) is described by the *implicit function theorem* [44]). At each time step i, given current value $\mathbf{X_i}$, the value of $\mathbf{Z_i}$ can be determined by using the Waltz algorithm to propagate interval values through the constraint network formed by (51), whose explicit form is (54). Because of the completeness of the Waltz algorithm, we can say that the value of $\mathbf{Z_i}$ calculated through (51) is the same as that calcuated from (54). In this sense we can substitute (54) into (50) and obtain

$$\begin{aligned} \mathbf{X}' &= \mathbf{F}(\mathbf{X}, \mathbf{G_1}(\mathbf{X})), \\ &= \mathbf{F_1}(\mathbf{X}). \end{aligned} \tag{55}$$

As $\mathbf{F_1}$ is also defined when each $Y_j(t) \subseteq [lo, hi]$, according to Theorem 4, we can reach statement (52). Applying $|| \cdot ||$ to (54) and the Lipschitz property of $\mathbf{G_1}$, we have

$$||\mathbf{Z_n}|| = ||\mathbf{G_1}(\mathbf{X_n})|| \leq L_{G1} ||\mathbf{X_n}||, \tag{56}$$

where L_{G1} is the Lipschitz constant determined by function vector $\mathbf{G1}$. The above formula combined with statement (52) gives us

$$||\mathbf{Z_n}|| \leq L_{G1} ||\mathbf{X_n}|| \leq L_{G1} ||\mathbf{X_0}|| + L_{G1} K_1 h \tag{57}$$

Similar to (54), from (51) we can also get

$$\mathbf{X} = \mathbf{G_2}(\mathbf{Z}). \tag{58}$$

Apply the Lipschitz property of function vector $\mathbf{G_2}$ to (57) and assume that L_{G2} is the Lipschitz constant for $\mathbf{G_2}$, we have

$$\begin{aligned} ||\mathbf{Z_n}|| &\leq L_{G1}||\mathbf{X_0}|| + L_{G1}K_1h \\ &= L_{G1}||\mathbf{G_2}(\mathbf{Z_0})|| + L_{G1}K_1h \\ &\leq L_{G1}L_{G2}||\mathbf{Z_0}|| + L_{G1}K_1h \end{aligned} \tag{59}$$

In the above let $K_2 = L_{G1}L_{G2}$, $K_3 = L_{G1}K_1$, and this leads us to statement (53).

□

From Theorem 5 we can conclude that the convergence and stability can still be achieved when dealing with DAEs. Finally, from Theorem 4 and Theorem 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 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 proposed in this report: all these simulation modes try to use a smaller initial interval values or point simulation to reduce the uncertainty of simulation results given the same size of the simulation step.

7 Experiments on Classical Dynamic Systems and Algebraic Loop Models

In this section we will verify the validity of the proposed non-constructive interval simulation approach through a series of experiments on (1) two classical dynamic systems: the spring-mass system and the Van der Pol Oscillator; and (2) an electrical circuit model containing an algebraic loop.

First, experiments on the Basic Interval Simulation are described and the results are reported in Section 7.1. This is followed by the experiments on the Sub-interval Simulation, details of which are presented in Section 7.2. In Section 7.3 the Monte-Carlo Interval Simulation is described. The simulation results from three point simulation approaches are reported in Section 7.4. In Section 7.5, the non-constructive simulation algorithm is tested against an electrical circuit model containing an algebraic loop. Finally in Section 7.6 we give a summary of all experiments.

7.1 Basic Interval Simulation

We use the spring-mass system, a representative of the simple harmonic oscillators [54], to test the performance of Basic Interval Simulation. The dynamics of the spring-mass system is governed by the following differential equation:

$$x'' = F - kx \tag{60}$$

In the above F is the external force which is assumed to be constant, k is a constant parameter related to the mass and the spring, and x is the displacement of the mass with respect to

Table 8: The *Morven* Model for the Spring-Mass System

Differential Plane 0
C1: func (dt 0 d, dt 0 x1)
C2: func (dt 1 x1, dt 0 x2)
C3: sub (dt 1 x2, dt 0 F, dt 0 d)
Differential Plane 1
C4: func (dt 1 d, dt 1 x1)
C5: func (dt 2 x1, dt 1 x2)
C6: sub (dt 2 x2, dt 1 F, dt 1 d)

the equilibrium position. As the behaviour of this spring-mass system is periodic with a frequency that depends on the nature of the mass and the spring, we expect to see such oscillatory behaviour in the simulation.

Similar to the example given in Table 3, Equation (60) is first converted into a *Morven* model as shown in Table 8, and the definitions of relevant constraints appearing in this table can be found in Table 2. In this model, the *func* in Constraints $C1$ and $C4$ is specified as $y = kx$, where y and x are the first and second arguments of the function, and k is the above mentioned constant parameter. For simplicity and without loss of generality, the value of k is set to 1. The *func* in Constraints $C2$ and $C5$ is specified to be an "equal" relation, which means two arguments in the function equal each other. In this model, d stands for the displacement; $x1$ corresponds to x in Equation (60); and $x2$ is the derivative of $x1$.

We first set the initial condition as follows: $x = [0.9, 1, 1]$, $x' = [1, 1]$, $F = [0, 0]$, and the rest of the variables remain unspecified. As two differential planes are given in the model, we choose the Taylor integration method so that we can make best use of the information about higher derivatives. (Unless otherwise specified, in this report models with two differential planes will be simulated using the Taylor method by default). Using the given initial condition, the simulation result for x obtained from Basic Interval Simulation is shown in Figure 2. The simulation is of the first ten seconds from the initial state and it can be seen that even with a fairly precise input interval quite a lot of excessive widening occurs. This output is not very useful as it is not apparent that any of the expected oscillations occur. This demonstrates the problem with Basic Interval Simulation.

7.2 Sub-Inteval Simulation

For the same initial state given in Section 7.1, we split the interval into 10, 100, and 1000 regular-spaced sub-intervals and perform the Sub-interval Simulation, and the trajectories of x are shown in Figure 3. From this set of outputs it can be seen that as the number of sub-intervals is increased the resulting interval suffers less from excessive widening. The major drawback with this method is that it takes a long time to execute since each sub-interval needs to be simulated individually.

7.3 Monte-Carlo Interval Simulation

Figure 4 shows the trajectories of x from the Monte-Carlo Interval Simulation (50 and 1000 samples) using the same initial state given in Section 7.1.

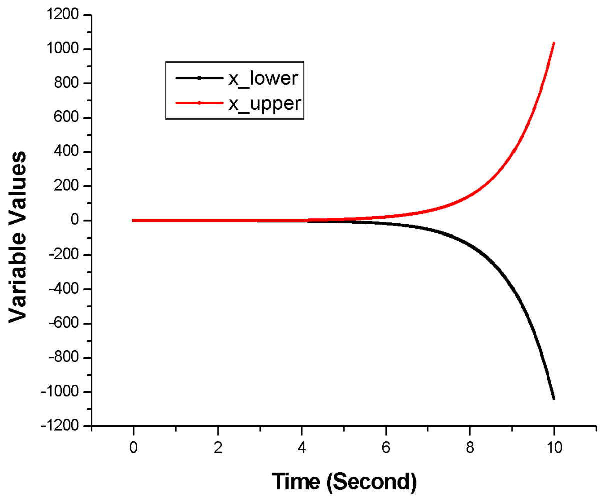
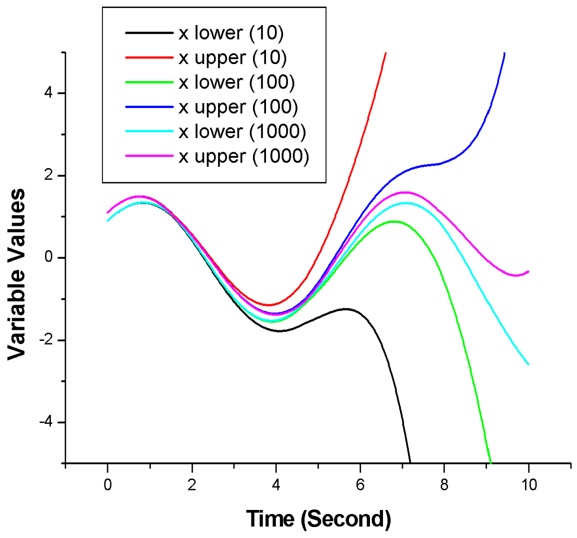
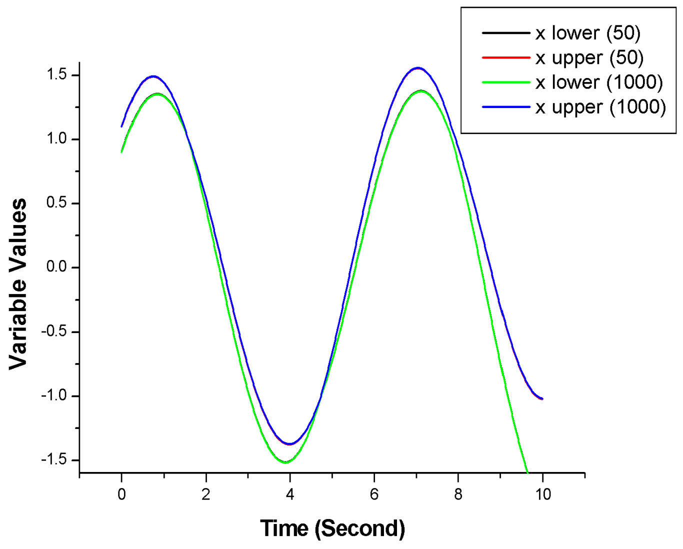


Figure 2: Basic Interval Simulation of the Spring-Mass System



"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 3: Sub-interval Simulation of the Spring-Mass System



"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 4: Monte-Carlo Interval Simulation of the Spring-Mass System

From the results we can see that the interval does not widen as rapidly as with the sub-interval method. This is because each iteration of the Monte-Carlo method is defined as having a very small interval width. In addition, the execution time was far quicker as fewer iterations were required to obtain relatively precise trajectories. This is indicated in Figure 4: we see that the results of 50 runs and 1000 runs are largely overlapped, which indicates that by using only 50 samples the precision of the simulation is already comparable with that using 1000 samples.

One of the disadvantages of this method is that due to the random nature of Monte-Carlo Simulation, there will be subtle differences between simulations of the same model with the same input parameters. In addition, this method is not guaranteed to bound all possible solutions although it may approximate the real solutions very closely.

7.4 Point Simulation Methods

This section discusses the results of the simulation modes that approximate intervals for simulating trajectories of models using groups of points. The model used to test the interval simulations is the Van der Pol oscillator [58], which has been used in several fields, such as Biology [19, 45] and Seismology [13]. The dynamics of the Van der Pol oscillator is described by the following equation:

$$x'' = -P(x^2 - 1)x' - Qx \tag{61}$$

In the above P and Q are two parameters. The corresponding *Morven* model is given in Table 9. In this model only one differential plane is used, and variables $A \sim E$ are auxiliary variables, the function of which is to help break long mathematical equations into *Morven* constraints. "one" in Constraint $C4$ is a constant parameter and set to 1. The function

Table 9: The *Morven* Model for the Van de Pol Oscillator

Differential Plane 0
C1:func (dt 1 x1, dt 0 x2)
C2: mul (dt 0 A, dt 0 Q, dt 0 x1)
C3: mul (dt 0 B, dt 0 x1, dt 0 x1)
C4: sub (dt 0 C, dt 0 one, dt 0 B)
C5: mul (dt 0 D, dt 0 P, dt 0 x2)
C6: mul (dt 0 E, dt 0 D, dt 0 C)
C7: sub (dt 1 x2, dt 0 E, dt 0 A)

constraint in Constraint $C1$ has the same definition as those in Constraints $C2$ and $C5$ given in Table 8. Variables $x1$ and $x2$ correspond to x and x' in Equation (61), respectively.

As in the model shown in Table 9 only one differential plane is available, and the information about the second derivatives is not explicitly given, the Taylor integration method given in Equation (14) is downgraded to the forward Euler Method as shown in Equation (12). For this model to produce more accurate simulation results, we use the Adams-Bashforth integration methods given in Equation (15) instead of the Taylor (Euler) method. Since we know that the truncation error of Adams-Bashforth method is less than that of the forward Euler method [4], one of the aims of the experiments carried out in this section is also to examine the effectiveness of the Adams-Bashforth integration methods. Unless otherwise specifically stated, all experiments reported in Section 7.4 use the two-step Adams-Bashforth method, whose formula is given by Equation (15) in the context of non-constructive simulation.

7.4.1 Extreme Point Simulation

We specify that x, $x' = [0.5, 1.5]$ and $P, Q = [1, 1]$ in the initial state, and other values remain unspecified. Figure 5 shows the output trajectories of the model for x and x' in the first 30 seconds. From the simulation results we see that using the extreme point method provides an output with no widening of the intervals, and as a result not only the upper and lower bounds of variable values are estimated, but also the periodic behaviours of the Van der Pol oscillator are clearly revealed.

Although not all of the solutions are bound by the output, this method still provides a very efficient approximation to the desired output. To demonstrate the accuracy of the Adams-Bashforth method, we perform the same simulation using the Taylor (more precisely, the forward Euler) method. Both results are validated by the traditional numerical simulation in which the fourth order Runge-Kutta method [50] was used, and it shows that the two-step Adams-Bashforth method offers a higher accuracy. Figure 6 shows a snapshot of both simulation results in the period from the third second to the fifth second.

7.4.2 Regular-spaced Point Simulation

Given the same initial state as in the extreme point simulation, Figure 7 shows in the first 30 seconds the output of the Regular-spaced point method using five points to approximate each interval. Since there are two intervals in the initial state of the Van der Pol oscillator for this problem there are 25 unique combinations of points to simulate. The output provides

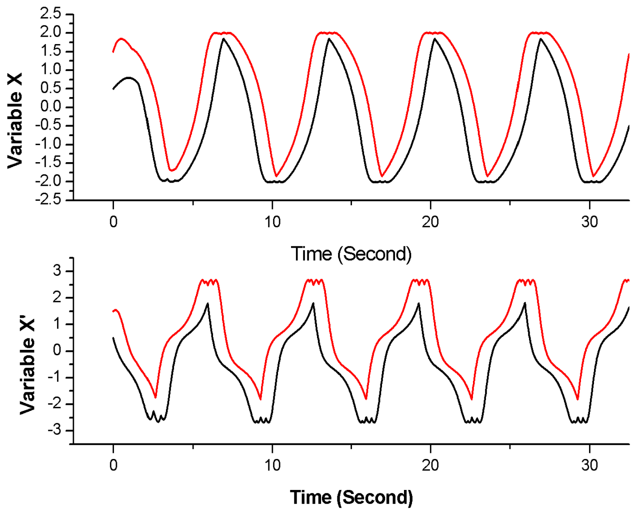
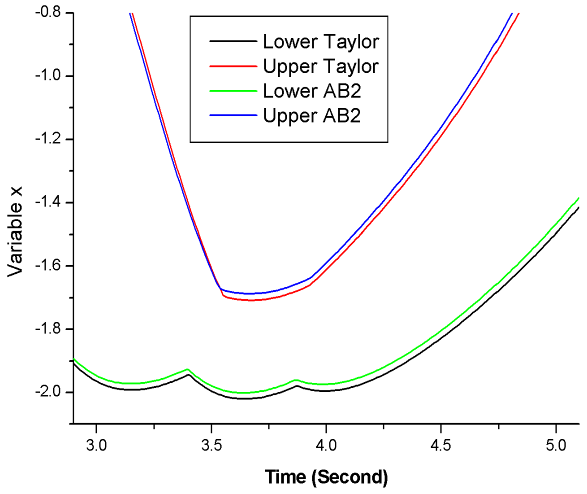


Figure 5: Extreme Points Simulation of the Van der Pol Oscillator



The trajectories in black and red are the lower and upper bounds estimated by the Taylor method, respectively. The trajectories in green and blue are the lower and upper bounds estimated by the two-step Adams-Bashforth method, respectively.

Figure 6: Comparison between the Two-step Adams-Bashforth and Taylor Methods for the Simulation of the Van der Pol Oscillator

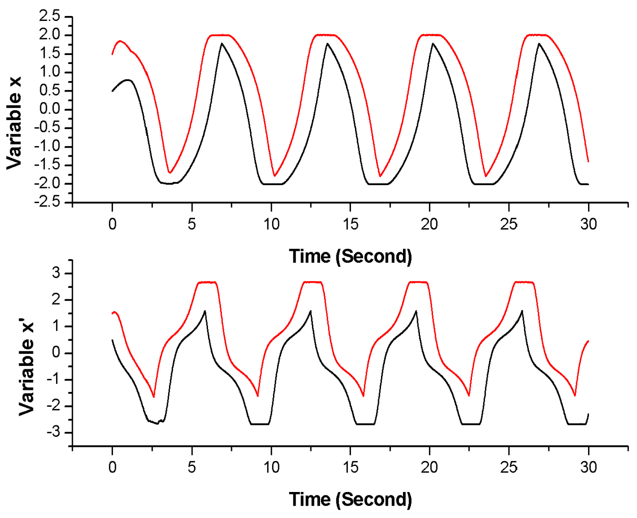


Figure 7: Regular-spaced Point Simulation of the Van der Pol Oscillator (5 Points Per Interval)

a very good trajectory with no widening and again the periodic behaviours of the oscillator are clearly revealed. However, on a close inspection it can be seen that there are a few errors around the peak of each oscillation. Figure 8 shows the same method except using 20 points to approximate the interval. This figure shows that using more points to estimate the intervals gives smoother trajectories.

In particular, there is a noticeable difference between the above two experiments when estimating the upper bounds of the peaks of oscillations in the trajectory for x', as shown in Figure 9, and a greater difference is observed in the first oscillation. There is very little difference between the outputs for x. For simulating this model, approximating the intervals using five to ten points offers a reasonable output trajectory executed in a reasonable length of time. In addition, as the number of points is increased the simulation tends to become sound and complete.

7.4.3 Monte-Carlo Point Simulation

The final simulation mode to be tested is Monte-Carlo Point Simulation. This method offers the benefits of being able to produce simulations with no excessive widening of the intervals.

Figure 10 shows the output of the Monte-Carlo method with 100 initial states. It can be seen that these graphs are very similar to the regular-spaced point method with 20 points. However, the Monte-Carlo version takes far less time to execute. This makes the Monte-Carlo Point Simulation technique a good method to approximate the outcome of the sound and complete simulation of the regular-spaced point technique. The disadvantage with this method is that since it uses random points within the intervals, no two outputs will be identical.

To test the Monte-Carlo method further, the Van der Pol oscillator was simulated again except using the following initial values: $x, x', Q = [1, 1]$ and $P = [0.9, 1.1]$. Having a parameter as an interval causes a different form of output as can be seen in Figure 11. The initial values for x and x' are real numbers, and hence have no width. However, one of the

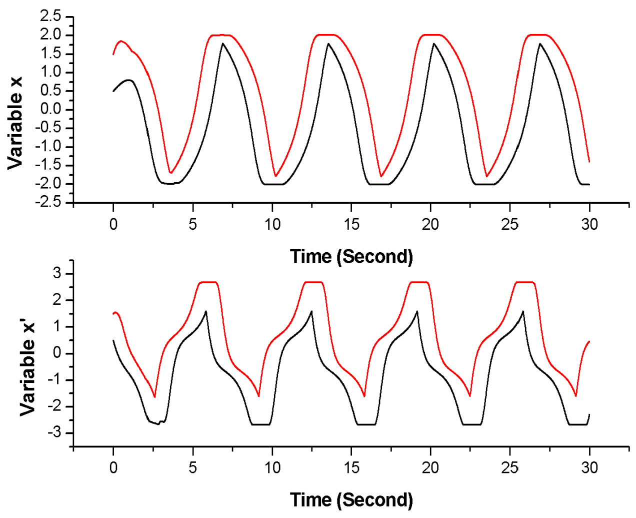


Figure 8: Regular-spaced Point Simulation of the Van der Pol Oscillator (20 Points Per Interval)

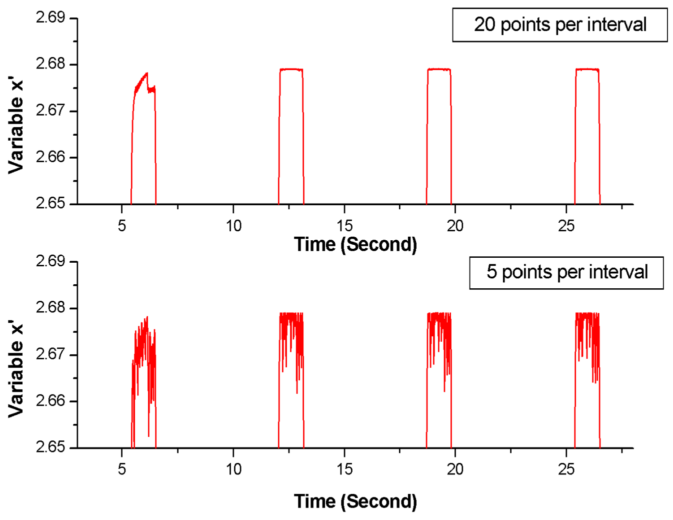


Figure 9: Comparison between the 20-points-per-interval Simulation and 5-points-per-interval Simulation on the Estimation of the Upper Bound of the Peak Area in the Trajectory of x'

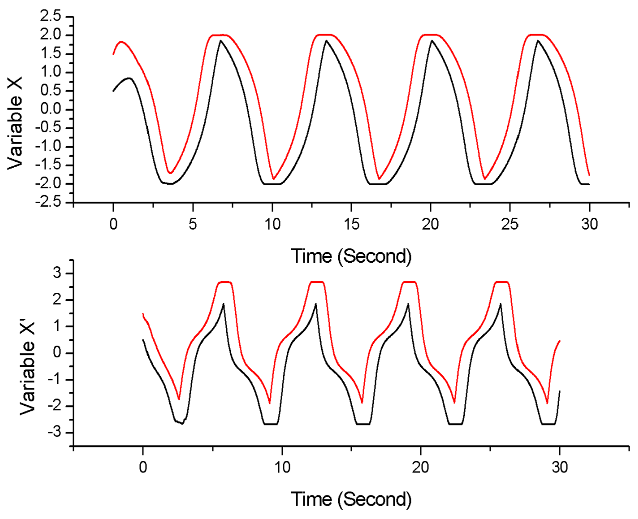


Figure 10: Monte-Carlo Points Simulation of the Van der Pol Oscillator

parameters takes interval values, and this causes the values of x and x' to widen with time. These outputs have been verified with the results reported by Keller [27] showing that the correct results are produced.

The final test for the interval simulation was using the Monte-Carlo Points method on the Van der Pol oscillator with $x, x', P, Q = [0.9, 1.1]$ which is similar to what a real problem might be like, that is, all parameters and initial values taking interval values. The simulated trajectories are shown in Figure 12. The outputs show that the initial interval is very narrow and how it widens with time. This is not due to errors in simulation but due to the parameters being incompletely specified. The simulation still results in a very useful output for determining how the system could behave in reality.

7.5 Experiments on an Algebraic-Loop Model

In this section, we complete our experiments by testing the non-constructive simulation algorithm on an electrical circuit model containing an algebraic loop. This model is given by Cellier [15] as shown in Figure 13 and listed by Equations (62) $\sim$ (70).

$$u_2 = u_3, \tag{62}$$

$$i_3 = \frac{u_3}{R_3}, \tag{63}$$

$$i_2 = \frac{u_2}{R_2}, \tag{64}$$

$$i_1 = i_2 + i_3, \tag{65}$$

$$u_1 = R_1 * i_1, \tag{66}$$

$$u_3 = U_0 - u_1, \tag{67}$$

$$u_L = u_1 + u_2, \tag{68}$$

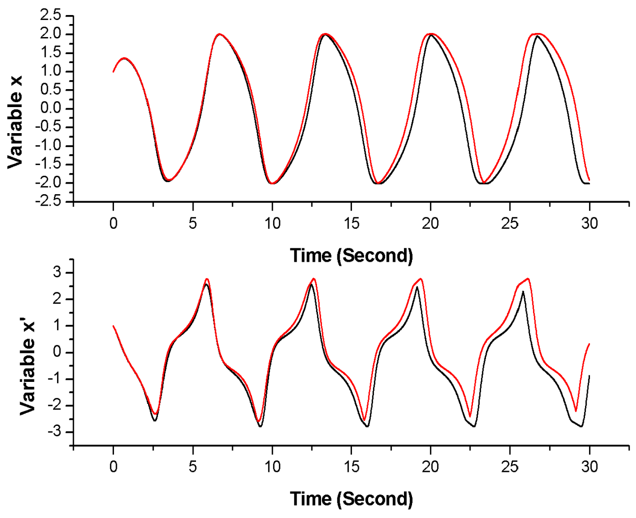


Figure 11: Monte-Carlo Points Simulation of the Van der Pol Oscillator With P Taking Interval Value

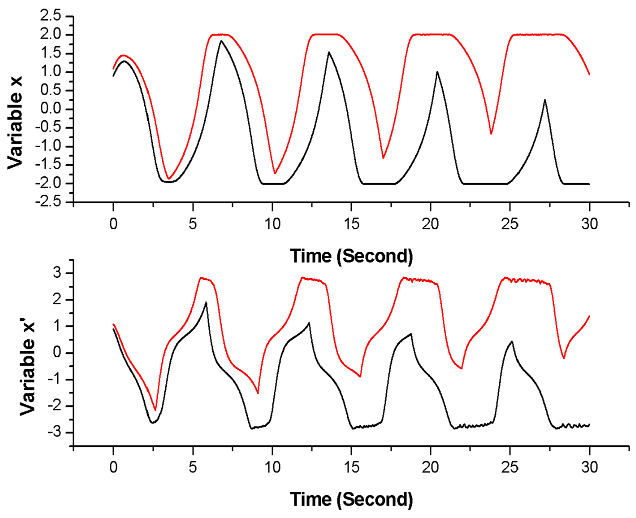


Figure 12: Monte-Carlo Points Simulation of the Van der Pol Oscillator with All Variables taking Interval Values

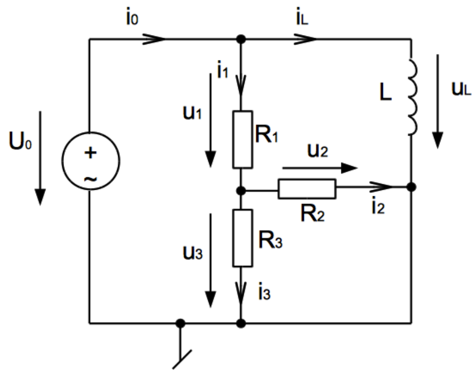


Figure 13: The Electrical Circuit Model Containing an Algebraic Loop (Reproduced from Cellier [15])

$$i_0 = i_1 + i_L, \tag{69}$$

$$i_L' = \frac{u_L}{L}. \tag{70}$$

In the above equations, U_0, u_L, $u_1 \sim u_3$ are voltages; i_L, $i_1 \sim i_3$ are currents; $R_1 \sim R_3$ are resistors; and L is a inductor. Among these variables, U_0 is the ideal voltage source and hence considered as an external variable. $R_1 \sim R_3$ and L are constant variables. A closer investigation into this model tells us that there is an algebraic loop existing among these equations: first, to calculate u_2, we need to know the value of u_3 according to Equation (62); from Equation (67) we see that the calculation of u_3 requires the value of u_1; according to Equation (66) the value of i_1 is needed in order to calculate u_1; from Equation (65) we see we must know the value of i_3 so that we can calculate i_1; finally calculating i_3 requires the value of u_3 according to Equation (63) , which completes the algebraic loop.

The *Morven* form of this electrical circuit model is given in Table 10. In this model the function constraint $C1$ is specified to be an "equal" relation, which means the values of u_2 and u_3 are always the same. To simulate this model, we set the initial condition as follows: $U_0 = [2.9, 3.1]$, R_2=R_3=L=[10, 10], R_1=[1, 1], and i_L=[0, 0]. The initial values of all other variables remain unspecified. The initial condition is chosen in such a way that it is consistent with the actual situation in this electrical circuit. We choose both the Monte-Carlo Interval and Point simulation modes (each with 50 random samples), so that we can test both the interval and point simulation modes against this algebraic-loop model. As only one differential plane is available in this model, we use the two-step AB integration method to make a more precise simulation.

The simulation results of variables i_L, i_L', and u_3 in the first 10 seconds using the Monte-Carlo Interval Simulation mode are shown in Figure 14. (The results using the Monte-Carlo Point Simulation mode are very similar, and thus not shown in this figure.) From the simulation results we see that the initial interval of u_3 has been correctly inferred at the very beginning of the simulation, even though it is within an algebraic loop. This results in the correct calculation of values of other variables, including the state variable i_L and its first derivative i_L'. The simulation results clearly show that non-constructive simulation can effectively handle algebraic loops and produce correct results.

Table 10: The *Morven* Model for the Electrical Circuit Model

Differential Plane 0
C1: func (dt 0 u_2, dt 0 u_3)
C2: div (dt 0 i_3, dt 0 u_3, dt 0 R_3)
C3: div (dt 0 i_2, dt 0 u_2, dt 0 R_2)
C4: add (dt 0 i_1, dt 0 i_2, dt 0 i_3)
C5: mul (dt 0 u_1, dt 0 R_1, dt 0 i_1)
C6: sub (dt 0 u_3, dt 0 U_0, dt 0 u_1)
C7: add (dt 0 u_L, dt 0 u_1, dt 0 u_2)
C8: add (dt 0 i_0, dt 0 i_1, dt 0 i_L)
C9: div (dt 1 i_L, dt 0 u_L, dt 0 L)

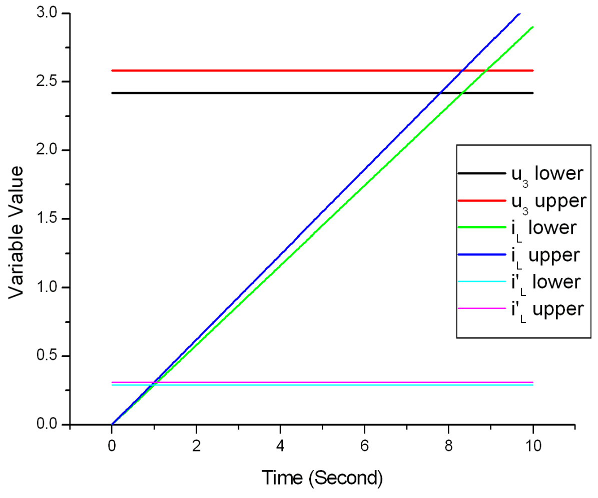


Figure 14: Monte-Carlo Interval Simulation of the Electrical Circuit Model

7.6 Summary of Experiments

In this section, we presented a series of experiments to verify the proposed non-constructive simulation approach. All the proposed simulation modes were tested using two classical dynamic systems with various initial states. Two different integration methods, the Taylor and Adams-Bashforth methods, were both tested in the simulation. In addition, the non-constructive simulation was tested against a model with an algebraic loop, and correct simulation results were obtained. The simulation results reported in this section clearly demonstrate the validity of our simulation approach.

From the experimental results we see that although complete, the Basic Interval Simulation suffers greatly from interval widening, and thus often cannot be used in practice. Instead, the Sub-interval Simulation provides tighter boundaries and is also a complete method, which makes it suitable for many real-world applications. Alternatively, the Monte-Carlo Interval Simulation makes a balance between the accuracy and the computational cost, and can be a very good option for many computationally expensive problems.

Apart from the above three interval simulation methods, point simulation methods are able to achieve a good approximation to the real solutions by sampling a sufficient number of representative points within intervals, instead of sampling sub-intervals. They can be considered as zero-interval versions of the aforementioned three interval simulation methods. One important advantage of point simulation approaches is that they do not suffer interval widening at all. This feature makes it possible for them to perform more complicated problems, such as when both parameters and variables take interval values. In particular, the Monte-Carlo Point Simulation demonstrates great potential for solving real-world problems.

Finally, the successful simulation of the algebraic-loop model demonstrates the advantage of non-constructive simulation: the simulation algorithm deals with all models in the same way, no matter whether they contain algebraic loops or not. This distinguishes itself from constructive simulation approaches, which normally require an individual module to first detect the existence of algebraic loops, and then perform additional operations on models with algebraic loops, such as trying to use an equation solver to remove the loops (One such example was given by Cellier [14], in which a system called DYMOLA was used to detect and remove the algebraic loop in the same electrical circuit model used in this report). The proposed non-constructive simulation offers an alternative and straightforward approach to deal with algebraic loops, which does not require additional operations upon models. This points out a very promising research field, although further improvement is needed, such as improving the efficiency and stability of the simulation.

8 Conclusions and Future Work

In this report we have presented a novel non-constructive interval approach for the simulation of dynamic systems. We first identified the gap between the way NS and QR researchers approach interval simulation. Starting from research results from the QR community, and based on the existing framework *Morven*, 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.

This makes our approach a collection of methods which includes two integration methods, one interval narrowing algorithm, and several simulation modes. The approach was theoretically investigated with regard to its completeness, soundness, convergence, and stability. Then the approach was verified by performing a series of experiments on classical dynamic systems as well as an algebraic-loop model. Experimental results validated our proposed approach, and promising results were obtained.

The features of the approach are summarised as follows:

- Non-constructive simulation is a straightforward method: compared to constructive simulation, it does not require any additional pre-analysis on DAEs or operations upon models with algebraic loops before the simulation starts. This makes it easy to deal with DAEs, models with interval-valued parameters, and models with algebraic loops, because it employs the same strategy (generate-and-eliminate) to simulate all these kinds of models.

- Non-constructive interval simulation is theoretically correct. Its completeness, soundness, convergence, and stability have been investigated.

- Two integration methods are provided for solving problems with different initial conditions and available information about derivatives.

- The interval narrowing algorithm reduces the spurious behaviours generated from the interval arithmetic operation, and improves the quality of solutions.

- Several simulation modes were implemented. The availability of different simulation modes makes the proposed simulation approach more flexible, efficient, and applicable to real-world problems: one may choose different simulation modes to meet different requirements.

- The simulation algorithm is implemented within the existing *Morven* framework, and thus benefits from features provided by *Morven*, such as the use of multiple differential planes and vector variables.

As a novel simulation algorithm, we acknowledge that there is still room for improvement in the design and theoretical analysis of the approach. From a theoretical point of view, the Monte-Carlo Interval Simulation should be investigated for its completeness, and a quantitative description for this should be provided. That is, given the size of the sub-intervals, how many simulations should be performed to achieve the given probability of the simulation being complete.

From the perspective of algorithm design, at the current stage our algorithm offers both deterministic and Monte-Carlo simulation modes to achieve good simulation results. 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. Through this a better balance between accuracy and efficiency can be achieved. For instance, it is worth investigating the potential of the Latin Hypercube Sampling approach [23], an advanced sampling approach suitable for dealing with systems with a large number of uncertain variables and parameters, to be implemented as a simulation mode in order to handle dynamic systems with many initial intervals.

Another design issue is the parallelisation of non-constructive simulation. The structure of our non-constructive simulation algorithm makes it convenient to parallelise some of the components of the proposed algorithm, and Bruce [8] has done some initial work to investigate this issue. To start with, the parallelisation of the simulation modes will be first investigated: apart from the Basic Interval Simulation, all the other simulation modes can be easily parallelised by assigning simulations with different sub-intervals or points to different CPUs, and maintaining a central repository to store simulation results from all CPUs. In the future, we will further investigate the parallelisation of our simulation algorithm, perform more experiments, and carry out relevant analysis for the obtained parallelisation benefit.

Finally, we expect that the research results presented in this report will contribute to both the NS and QR communities by inspiring the development of new non-constructive numerical and interval simulation algorithms, and we foresee the non-constructive approach as a fruitful research direction for simulation at both quantitative and semi-quantitative levels. The proposed approach is also a good example of combining established methodologies from both AI and scientific computation areas to develop a novel hybrid approach. This shows that in general the AI research can benefit from bringing in knowledge and methods from other areas, especially long-established research areas where there are many research results available. Furthermore, we hope the presented algorithm can be applied to more real-world problems as either an independent tool or a complementary approach to both qualitative and quantitative simulation.

References

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

[2] 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.

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

[4] Uri M. Ascher and Linda Ruth Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, chapter 5, pages 124–129. SIAM, 1998.

[5] R. Barrio. Performance of the Taylor series method for ODEs/DAEs. *Applied Mathematics and Computation*, 163:525–545, 2005.

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

[7] 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.

[8] Allan M. Bruce. *JMorven: A Framework for parallel non-constructive qualitative reasoning and fuzzy interval simulation.* PhD thesis, Department of Computing Science, Univeristy of Aberdeen, October 2007.

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

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

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

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

[13] J. Cartwright, V. Eguiluz, E. Hernandez-Garcia, and O. Piro. Dynamics of elastic excitable media. *International Journal of Bifurcation and Chaos*, 9:2197–2202, 1999.

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

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

[16] 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.

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

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

[19] R FitzHugh. Impulses and physiological states in theoretical models of nerve membranes. *Biophysics Journal*, 1(6):445–466, 1961.

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

[21] Andreas Griewank and Andrea Walther. On the efficient generation of Taylor expansions for DAE solutions by automatic differentiation. In *PARA′04, State-of-the-art in scientific computing*, volume 3732 of *LNCS*, pages 1103–1111. Springer-Verlag, 2006.

[22] 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.

[23] R.L. Iman, J.C. Helton, and J.E. Campbell. An approach to sensitivity analysis of computer models, part 1. introduction, input variable selection and preliminary variable assessment. *Journal of Quality Technology*, 13(3):174–183, 1981.

[24] R. Isermann and P. Ball. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5):709–719, May 1997.

[25] H. Kay and B. Kuipers. Numerical behavior envelopes for qualitative models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-93)*, pages 606–613, Cambridge, MA, 1993. AAAI/MIT Press.

[26] Herbert Kay. SQSIM: a simulator for imprecise ODE models. *Computers and Chemical Engineering*, 23(1):27–46, November 1998.

[27] U. E. Keller. *Qualitative Model Reference Adaptive Control.* PhD thesis, Heriott-Watt University, September 1999.

[28] 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.

[29] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[30] Benjamin Kuipers. Modeling and simulation with incomplete knowledge. *Automatica*, 25(4):571–585, 1989.

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

[32] Youdong Lin and Mark A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Applied Numerical Mathematics*, 57(10):1145–1162, October 2007.

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

[34] Rudolf J. Lohner. Computations of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In J. R. Cash and I. Gladwell, editors, *Computational ordinary differential equations*, pages 425–435, Oxford, 1992. Clarendon Press.

[35] Kyoko Makino and Martin Berz. COSY INFINITY version 9. *Nuclear Instruments and Methods in Physics Research A*, 558:346–350, 2006.

[36] S Markov and R Angelov. An interval method for systems of ODE. In *Proceedings of the International Symposium on Interval Mathematics on Interval Mathematics*, pages 103–108, London, UK, 1985. Springer-Verlag.

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

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

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

[40] Ramn E. Moore. *Methods and applications of interval analysis*, chapter 2, pages 9–17. SIAM, 1979.

[41] Ramn E. Moore. *Methods and applications of interval analysis*, chapter 3.2, pages 20–21. SIAM, 1979.

[42] Ramn E. Moore. *Methods and applications of interval analysis*, page 34. SIAM, 1979.

[43] A.J. Morgan. *Qualitative behaviour of Dynamic Physical Systems.* PhD thesis, University of Cambridge, 1988.

[44] James R. Munkres. *Analysis on Manifolds.* Perseus Books (Sd), 1990.

[45] J. Nagumo, S. Arimoto, and S Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IEEE*, 50:2061–2070, 1962.

[46] 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.

[47] Nedialko S. Nedialkov. Interval tools for ODEs and DAEs. In *Proceedings of the 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*, pages 4–4. IEEE Computer Society Washington, DC, USA, 2006.

[48] Nedialko S. Nedialkov and K. R. Jackson. The design and implementation of a validated object-oriented solver for IVPs for ODEs. Technical report, Software Quality Research Laboratory, Department of Computing and Software, McMaster University, Hamilton, Canada, 2002.

[49] 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.

[50] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing (2nd Edition)*, chapter 16, pages 710–722. Cambridge University Press, 1992.

[51] 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.

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

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

[54] Raymond A. Serway and John W. Jewett. *Physics for Scientists and Engineers (7nd Edition)*, volume 1, chapter 15.2, pages 420–426. Thomson-Brooks/Cole, 2003.

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

[56] Ole Stauning and Claus Bendtsen. Fadbad++. `http://www.fadbad.com/fadbad.html`, 2007.

[57] Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*, chapter 2.4 and 12.2. Cambridge University Press, 2003.

[58] B. Van der Pol and J. Van der Mark. Frequency demultiplication. *Nature*, 120:363–364, 1927.

[59] Marcos Vescovi, Adam Farquhar, and Yumi Iwasaki. Numerical interval simulation: combined qualitative and quantitative simulation to bound behaviors of non-monotonic systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1806–1812. AAAI Press, 1995.

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

[61] Seth Warner. *Modern Algebra*, chapter 1, page 6. Courier Dover Publications, 1990.

[62] Thomas A. Whitelaw. *Introduction to Abstract Algebra*, chapter 1.8, pages 18–21. Springer, 1994.

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

[64] Brian C. Williams. The use of continuity in a qualitative physics. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-84)*, pages 350–354, Austin, Texas, 1984. AAAI Press.