Validated Enclosure of Uncertain Nonlinear Equations using SIVIA Monte-Carlo

Nisha Rani Mahato[1], Luc Jaulin[2](0000 − 0002 − 0938 − 0615), Snehashish Chakraverty[1](0000 − 0003 − 4857 − 644X), and Jean Dezert[3]

[1] Department of Mathematics, National Institute of Technology Rourkela, Rourkela-769008, Odisha, India
[2] ENSTA-Bretagne, LabSTICC, CNRS 6285, 2 rue François Verny, 29806 Brest, France
[3] The French Aerospace Lab, 91120 Palaiseau, France
nisha.mahato1@gmail.com, lucjaulin@gmail.com, sne_chak@yahoo.com, jdezert@gmail.com

Abstract. The dynamical systems in various science and engineering problems are often governed by nonlinear equations (differential equations). Due to insufficiency and incompleteness of system information, the parameters in such equations may have uncertainty. Interval analysis serves as an efficient tool for handling uncertainties in terms of closed intervals. One of the major problem with interval analysis is handling 'dependency problems' for computation of tightest range of solution enclosure or exact enclosure. Such dependency problems are often observed while dealing with complex nonlinear equations. In this regard, initially two test problems comprising of interval nonlinear equations are considered. The Set Inversion via Interval Analysis (SIVIA) along with Monte-Carlo approach is used to compute the exact enclosure of the test problems. Further, the efficiency of the proposed approach has also been verified for solving nonlinear differential equations (Van der Pol oscillator) subject to interval initial conditions.

Keywords: Uncertain nonlinear equations · Nonlinear oscillator · Dependency problem · SIVIA Monte-Carlo · Contractor

1 Introduction

Various vibration problems in science and engineering disciplines viz. structural mechanics, control theory, seismology, physics, biology etc. may be expressed in terms of nonlinear equations, system of nonlinear equations and nonlinear differential equations. Generally, the parameters in such equations deal with precise variables. But, the insufficiency and incompleteness of the system information often led to parameters or variables with imprecision or uncertainty. For instance, let us consider a nonlinear damped spring-mass system as given in Fig. 1 governed by the equation,

$$m\ddot{x} + c\dot{x} + \alpha\dot{x}^2 + kx + \beta x^3 = f(t) \tag{1}$$

where, m, c and k are respectively mass, damping and stiffness of the nonlinear system. Here, the external force applied on the system is $f(t)$ with damping force $f_d = c\dot{x} + \alpha\dot{x}^2$ and spring force $f_s = kx + \beta x^3$.

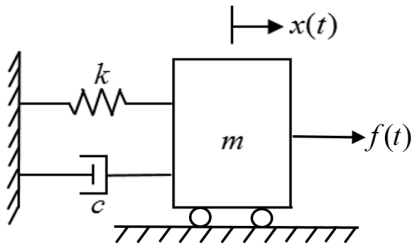

Fig. 1. Damped spring-mass system

The uncertainty of the material properties in Eq. (1) led to uncertain nonlinear differential equation. Such uncertainties may be modeled either using probabilistic approach, interval computation or fuzzy set theory. In case of non-availability of sufficient experimental data, probabilistic methods may not be able to deliver reliable results. Moreover, in fuzzy set theory a fuzzy number is expressed in terms of closed intervals through α-cut approach. As such, interval analysis have emerged as a powerful tool for various practical problems in handling the uncertainties.

In early 1960s the pioneer concept related to interval computations, functions, matrices, integral and differential equations has been started by R. E. Moore [12–14]. System of equations, algebraic eigenvalue problems, second order initial and boundary value problems has been discussed by Alefeld and Herzberger [3]. Guaranteed interval computations with respect to set approximations, parameter and state estimation with applications in robust control and robotics are addressed by Jaulin et al. [10]. While dealing with interval computations, one of the major obstacle is to handle the 'dependency problems' effectively such that the tightest enclosure of solution bound may be obtained. Such dependency problems often occur in dealing with systems governed by complex nonlinear equations which often lead to over-estimation of solution bound. The dependency problem due to overestimation (*wrapping effect*) has been studied by Krämer [11] with respect to generalized interval arithmetic proposed by Hansen [9]. Other approach for reduction of overestimation while handling dependency problem may be performed using contractors [10], affine arithmetic [16] and/or parametric forms. As such, the present work proceeds with the introduction section. The preliminaries of classical arithmetic of Interval Analysis (IA) along with its application for two complex nonlinear equations comprising of imprecise variables are considered in Section 2. The Set Inversion via Interval Analysis (SIVIA) along with Monte-Carlo approach is then used to compute the exact

enclosure of the two test problems in Section 3. Further, the proposed approach has also been verified for computing validated enclosure of nonlinear differential equations (Van der Pol oscillator) subject to interval initial conditions in Section 4.

2 Classical Interval Computations

Interval analysis deals with interval computations on a set of closed intervals $\mathbb{IR}$ of real line $\mathbb{R}$, in order to obtain the tightest bound or enclosure for uncertain systems. A closed interval $[x] \subset \mathbb{IR}$ is denoted by $[x] = [\underline{x}, \overline{x}]$ such that

$$[x] = [\underline{x}, \overline{x}] = \{t \mid \underline{x} \leq t \leq \overline{x}, \text{ where } \underline{x}, \ \overline{x} \in \mathbb{R}\}.$$

Here, $\underline{x} = \inf[x]$ is the inifimum or lower bound of $[x]$ and $\overline{x} = \sup[x]$ is the supremum or upper bound of $[x]$. The width and center of $[x]$ may be referred as $[x]^w = \overline{x} - \underline{x}$ and $[x]^c = \frac{\underline{x}+\overline{x}}{2}$ respectively.

Basic operations using classical interval arithmetic given in Moore et al. [14] are illustrated as follows:

- **Addition**: $[x] + [y] = [\underline{x} + \underline{y}, \ \overline{x} + \overline{y}]$
- **Subtraction**: $[x] - [y] = [\underline{x} - \overline{y}, \ \overline{x} - \underline{y}]$
- **Multiplication**: $[x] \cdot [y] = [\min\{S.\,([\underline{x}], [y])\}, \max\{S.\,([x], [y])\}]$, where $S.\,([x], [y]) = \{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}$
- **Division**: $[x]/[y] = \begin{cases} \left[\frac{\underline{x}}{\overline{y}}, \frac{\overline{x}}{\underline{y}}\right], & 0 \notin [\underline{y}, \ \overline{y}], \\ (-\infty, \infty), & 0 \in [\underline{y}, \ \overline{y}] \end{cases}$
- **Power**:
  - If $n > 0$ is an odd number, then $[x]^n = [\underline{x}^n, \overline{x}^n]$
  - If $n > 0$ is an even number, then $[x]^n = \begin{cases} [\underline{x}^n, \overline{x}^n], & [x] > 0 \\ [\overline{x}^n, \underline{x}^n], & [x] < 0 \\ [0, \max\{\underline{x}^n, \overline{x}^n\}], & 0 \in [x] \end{cases}$

Then, we have illustrated two test examples for the implementation of basic interval arithmetic in Examples 1 and 2.

Example 1: Compute the bound $[z_1]$ satisfying constraint

$$z_1 = x_1 y_1 + x_1 y_3 + x_3 y_1 \tag{2}$$

such that $x_1 + x_2 + x_3 = 1$ and $y_1 + y_2 + y_3 = 1$. Here, $x_1 \in [x_1] = [0.2, 0.3]$, $x_2 \in [x_2] = [0.1, 0.2]$, $y_1 \in [y_1] = [0.4, 0.6]$ and $y_2 \in [y_2] = [0.2, 0.3]$.

Using classical IA, the bounds $[x_3]$ and $[y_3]$ are initially estimated as

$$[x_3] \sim 1 - [x_1] - [x_2] = [0.5, 0.7] \text{ and } [y_3] \sim 1 - [y_1] - [y_2] = [0.1, 0.4]$$

respectively with respect to the constraints $x_1 + x_2 + x_3 = 1$ and $y_1 + y_2 + y_3 = 1$. Then, the bound $[z_1]$ is obtained as

$$[z_1]^{\mathbf{IA}} \sim [x_1] \cdot [y_1] + [x_1] \cdot [y_3] + [x_3] \cdot [y_1] = [0.30, 0.72]. \tag{3}$$

Further, we have considered a more complicated nonlinear constraint in Example 2, related to problems of multi-criteria decision-making under imprecise scores given in Dezert et al. [7].

Example 2: [7] Compute the bound $[z_2]$ satisfying constraint

$$z_2 = z_1 + \frac{x_1^2 y_2}{x_1 + y_2} + \frac{y_1^2 x_2}{y_1 + x_2} \tag{4}$$

such that $x_1 \in [0.2, 0.3]$, $x_2 \in [0.1, 0.2]$, $y_1 \in [0.4, 0.6]$ and $y_2 \in [0.2, 0.3]$.

Here, the bound of $[z_2]$ is obtained as

$$[z_2]^{\mathbf{IA}} \sim [z_1]^{\mathbf{IA}} + \frac{[x_1]^2[y_2]}{[x_1] + [y_2]} + \frac{[y_1]^2[x_2]}{[y_1] + [x_2]} = [0.3333, 0.9315]. \tag{5}$$

The enclosures obtained in Eqs. (3) and (5) have been compared with enclosures obtained using Monte-Carlo simulation in Table 1.

Table 1. Interval bounds of z_1 and z_2

i	Interval bounds	
	$[z_i]^{\mathbf{IA}}$	$[z_i]^{\mathbf{MC}}$
1	$[0.30, 0.72]$	$[0.3850, 0.5935]$
2	$[0.3333, 0.9315]$	$[0.4617, 0.6825]$

Here, the Monte-Carlo simulation approach using uniformly distributed 100000 independent random sample values of variables x_1, x_2, y_1 and y_2 have been considered, where $x_1 \sim U([x_1])$, $x_2 \sim U([x_2])$, $y_1 \sim U([y_1])$ and $y_2 \sim U([y_2])$. From Table 1, it is worth mentioning that the bounds for $i = 1, 2$ satisfy

$$[z_i]^{\mathbf{MC}} \subset [z_i]^{\mathbf{IA}}.$$

In case of more sample values, the Monte-Carlo simulation may yield better interval enclosure with respect to the constraints (2) and (4), but such approach is inefficient with respect to computational time. So, we may consider the problem in handling interval computations as to interpret the tightest or the exact enclosure $[z_i]$ of z_i that satisfies

$$[z_i]^{\mathbf{MC}} \subset [z_i] \subset [z_i]^{\mathbf{IA}} \tag{6}$$

such that

$$\inf [z_i]^{\mathbf{IA}} \leq \inf [z_i] \leq \inf [z_i]^{\mathbf{MC}} \text{ and } \sup [z_i]^{\mathbf{MC}} \leq \sup [z_i] \leq \sup [z_i]^{\mathbf{IA}} \tag{7}$$

$$\text{or } \underline{z}_i^{\mathbf{IA}} \leq \underline{z}_i \leq \underline{z}_i^{\mathbf{MC}} \text{ and } \overline{z}_i^{\mathbf{MC}} \leq \overline{z}_i \leq \overline{z}_i^{\mathbf{IA}}. \tag{8}$$

Although in the above computations, interval arithmetic looks simple for basic operations with intervals and seems appealing. But, the 'dependency problem' is a major obstacle when complicated expressions have to be computed in order to find tightest enclosure. In this regard, the dependency effect has been discussed in detail in next section.

2.1 Dependency Problem in IA

Variable or parameter dependency problem in IA is generally exhibited when we have more than one occurrence of imprecise parameter in the governing constraint. For instance, in case of the nonlinear constraint

$$z = x^2 + y^2 \text{ for } x \in [0.1, 0.5] \text{ and } y \in [-0.6, 0.1],$$

the occurrence of each imprecise variable x and y is once. The computation of enclosure with respect to constraint $z = x^2 + y^2$ using classical IA results to $[z]^{\mathbf{IA}} = [0.01, 0.61]$ which is found equivalent to the Monte-Carlo simulation of $x \sim U([0.1, 0.5])$, $y \sim U([-0.6, 0.1])$ for 100000 sample values yield $[z]^{\mathbf{MC}} = [0.01, 0.61]$. But, the complexity occurs while dealing with complex nonlinear constraints as given in Examples 1 and 2, where the dependency effect is exhibited due to multiple occurrence on imprecise variables.

The dependency effect may be reduced by replacing the constraint given in Eq. (2) with an equivalent simpler constraint having less (or none) redundant variables. For instance, the equivalent constraint

$$z_1 = (1 - x_2)y_2 + x_3 y_1 \tag{9}$$

results to a better enclosure approximation $[z_1]^{\mathbf{IA}} = [0.34, 0.66]$. Here, the interval bound $[0.34, 0.66]$ is contained in the bound $[0.30, 0.72]$ obtained using the equivalent constraint given in Eq. (2). But, on the other-hand an equivalent constraint

$$z_1 = (1 - y_2)x_1 + (1 - x_2)y_1 - x_1 y_1 \tag{10}$$

results to an overestimated bound $[z_1]^{\mathbf{IA}} = [0.28, 0.70]$. Due to such dependency, the interval bounds often yield overestimation of the tightest enclosure. Similar, dependency effect is exhibited while computing $[z_2]^{\mathbf{IA}}$ for constraints $z_2 = z_1 + \left(\frac{1}{x_1 y_2} + \frac{1}{x_1^2}\right)^{-1} + \frac{y_1^2 x_2}{y_1 + x_2}$ and $z_2 = z_1 + \frac{x_1^2 y_2}{x_1 + y_2} + \left(\frac{1}{y_1 x_2} + \frac{1}{y_1^2}\right)^{-1}$ with respect to (4). As such, identification of constraint yielding tightest enclosure is cumbersome. In this regard, the problem formulation for reduction of dependency effect has been carried out in the next section.

Problem Formulation The main aim in the present work is to compute tightest enclosure $[\underline{z}_i, \overline{z}_i]$ or exact enclosure such that $[z_i]^{\mathbf{MC}} \sim [z_i]^{\mathbf{IA}}$ or

$$\underline{z}_i^{\mathbf{IA}} = \underline{z}_i = \underline{z}_i^{\mathbf{MC}} \text{ and } \overline{z}_i^{\mathbf{MC}} = \overline{z}_i = \overline{z}_i^{\mathbf{IA}}. \tag{11}$$

associated with some nonlinear constraint $z_i = f(x_1, x_2, y_1, y_2)$, where $x_i \in [x_i]$ and $y_i \in [y_i]$ for $i = 1, 2$. In this regard, SIVIA Monte-Carlo approach based on set inversion via interval computations and Monte-Carlo simulation have been proposed to estimate exact bounds in next section.

3 SIVIA Monte-Carlo Approach

Initially, the general procedure of SIVIA has been incorporated in Section 3.1 followed by contractors in Section 3.2. Finally, the combination of SIVIA with Monte-Carlo approach has been performed in Section 3.3.

3.1 SIVIA

Set inversion of a typical set $\mathbb{X} \subset \mathbb{R}^m$ with respect to function $f : \mathbb{R}^m \to \mathbb{R}^n$ is expressed as

$$\mathbb{X} = f^{-1}(\mathbb{Y}) = \{\mathbf{x} \in \mathbb{R}^m \mid f(\mathbf{x}) \in \mathbb{Y}\}$$

where, $\mathbb{Y} \subset \mathbb{R}^n$. In case of SIVIA [10], an initial search set $[\mathbf{x}_0]$ is assumed containing the required set $\mathbb{X}$. Then, using sub-pavings as given in Fig. 2, the desired enclosure of solution set $\mathbb{X}$ is obtained based on the inclusion properties:

1. *Case I*: $[f]([\mathbf{x}]) \subset \mathbb{Y} \implies [\mathbf{x}] \subset \mathbb{X}$, then $[\mathbf{x}]$ is a solution,
2. *Case II*: $[f]([\mathbf{x}]) \cap \mathbb{Y} = \phi \implies [\mathbf{x}] \cap \mathbb{X} = \phi$, then $[\mathbf{x}]$ is not a solution,
3. *Case III*: $[f]([\mathbf{x}]) \cap \mathbb{Y} \neq \phi$ and $[f]([\mathbf{x}]) \not\subset \mathbb{Y}$ then, $[\mathbf{x}]$ is an undetermined solution.

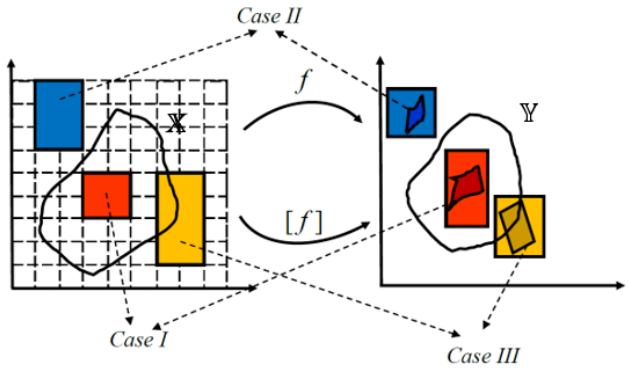

Fig. 2. Set inversion via interval analysis

The detailed illustration of set computation using SIVIA based on regular sub-pavings, bisections etc. may be found in [10]. The sub-pavings in SIVIA may be improved with the usage of contractors discussed in next section.

3.2 Contractor

Contractor: ([4, 10]) A contractor $\mathcal{C}$ associated with a set $\mathbb{X} \subset \mathbb{R}^n$ over domain $\mathbb{D}$ is an operator

$$\mathcal{C} : \mathbb{IR}^n \to \mathbb{IR}^n$$

satisfying the following properties:

- Contraction: $\mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}]$, $\forall [\mathbf{x}] \in \mathbb{IR}^n$,
- Completeness: $\mathcal{C}([\mathbf{x}]) \cap \mathbb{X} = [\mathbf{x}] \cap \mathbb{X}$, $\forall [\mathbf{x}] \in \mathbb{IR}^n$.

The pictorial representation of implementation of contractor over set $\mathbb{X} \subset \mathbb{R}^2$ is illustrated in Fig. 3

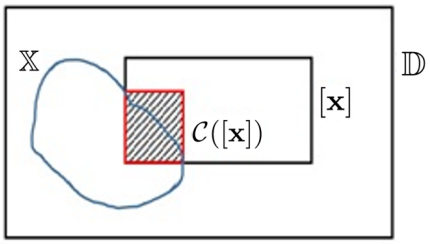

Fig. 3. Contraction of $[x]$

There exist various types of contractors viz. fixed-point, forward-backward, Newton, Gauss-Seidel contractors etc. Contractor based set inversion of leminscate curve $(x^2 + y^2)^2 + a^2(x^2 - y^2) = 0$ having width $a \in [2, 3]$ has been obtained based on the PyIbex library [6] and depicted in Fig. 4, where the initial search set is $[-4, 4] \times [-4, 4]$.

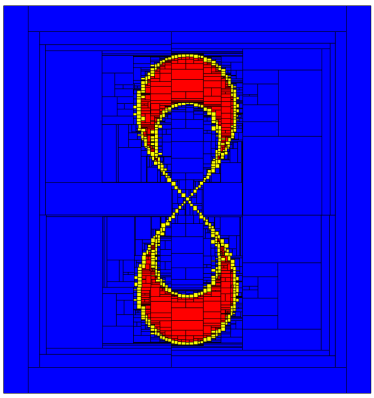

Fig. 4. SIVIA of leminscate curve with width $[2, 3]$

In order to perform SIVIA Monte-Carlo approach, we have used forward-backward and fixed-point contractors. Detailed implementation of forward-backward and fixed-point contractors have been incorporated in Appendix.

3.3 SIVIA Monte-Carlo

SIVIA Monte-Carlo is two form iterative methodology that includes implementation of SIVIA using contractor programming and the Monte-Carlo simulation till the exact enclosure is obtained satisfying (11). In this regard, the iterative procedure is incorporated in Algorithm 1 with respect to constraint $z = f(x_1, x_2, \ldots, x_n)$ such that each $x_i \in [x_i] \in \mathbb{IR}$ for $i = 1, 2, \ldots, n$. Here, the initial search set containing the exact enclosure is assumed as $[z_0]$.

Algorithm 1: Implementation of SIVIA Monte-Carlo approach

Input: $[x_i]$ for $i = 1, 2, \ldots, n$; Initial domain $[\mathbf{x}] = [x_1], [x_2], \ldots, [x_n] \in \mathbb{D}$;
Initial search set $[z_0]$

Step 1: Compute enclosure using Monte-Carlo
$\underline{z}^{\mathbf{MC}} = mcl([\mathbf{x}])$ and $\overline{z}^{\mathbf{MC}} = mcu([\mathbf{x}])$
Step 2: Compute enclosure using contractors
$\underline{z}^{\mathbf{IA}} = Ctcl([\mathbf{x}], [z_0])$ and $\overline{z}^{\mathbf{IA}} = Ctcu([\mathbf{x}], [z_0])$
Step 3: Improve lower and upper range of z
$\underline{z} \in \left[\underline{z}^{\mathbf{IA}}, \underline{z}^{\mathbf{MC}}\right]$ and $\overline{z} \in \left[\overline{z}^{\mathbf{MC}}, \overline{z}^{\mathbf{IA}}\right]$
Step 4: Compute improved lower $[\underline{\mathbf{x}}] = [f]^{-1}(\left[\underline{z}^{\mathbf{IA}}, \underline{z}^{\mathbf{MC}}\right])$ and
upper $[\overline{\mathbf{x}}] = [f]^{-1}(\left[\overline{z}^{\mathbf{MC}}, \overline{z}^{\mathbf{IA}}\right])$ domains using SIVIA
$[\underline{\mathbf{x}}], [\overline{\mathbf{x}}]$=SIVIA$([\mathbf{x}], [f], [z_0], \epsilon)$
Step 5: Repeat steps 1 to 3 for domains $[\underline{\mathbf{x}}]$ and $[\overline{\mathbf{x}}]$
Step 6: Repeat step 4 for different domains $[\underline{\mathbf{x}}]$ and $[\overline{\mathbf{x}}]$
Step 7: Iterate steps 4 and 5 till $\underline{z} = \underline{z}^{\mathbf{IA}} \sim \underline{z}^{\mathbf{MC}}$ and $\overline{z} = \overline{z}^{\mathbf{MC}} \sim \overline{z}^{\mathbf{IA}}$
Output: $[\underline{z}, \overline{z}]$

In Algorithm 1 , $mcl(\cdot), mcu(\cdot)$ are functions that compute the minimum and maximum function value with respect to domain $[\mathbf{x}] \in \mathbb{D}$. Then, $Ctcl(\cdot), Ctcu(\cdot)$ uses forward-backward contractor along with fixed-point contractor for computing interval enclosure based on classical IA. Further, SIVIA$(\cdot)$ computes the set inversion for domain $[\mathbf{x}] \in \mathbb{D}$ based on constraint function f with precision ϵ.

Let us again consider the Examples 1 and 2 in order to compute the exact enclosure using SIVIA Monte-Carlo in Example 3.

Example 3: Compute the interval bounds for the constraints

$$z_1 = x_1y_1 + x_1y_3 + x_3y_1 \text{ and } z_2 = z_1 + \frac{x_1^2y_2}{x_1 + y_2} + \frac{y_1^2x_2}{y_1 + x_2}$$

using SIVIA Monte-Carlo such that $x_1 + x_2 + x_3 = 1$ and $y_1 + y_2 + y_3 = 1$. Again, $x_1 \in [x_1] = [0.2, 0.3]$, $x_2 \in [x_2] = [0.1, 0.2]$, $y_1 \in [y_1] = [0.4, 0.6]$ and

$y_2 \in [y_2] = [0.2, 0.3]$. Using Algorithm 1 for SIVIA precision $\epsilon = 0.001$ and different sample values viz. $100000, 1000, 100, 10$, the tightest enclosures with respect to constraints $z_1 = x_1y_1 + x_1y_3 + x_3y_1$ and $z_2 = z_1 + \frac{x_1^2y_2}{x_1+y_2} + \frac{y_1^2x_2}{y_1+x_2}$ for different sample values are obtained and incorporated in Tables 2 and 3 respectively.

Table 2. Interval enclosure of z_1

Iterations	SIVIA (0.001 precision) and Monte Carlo samples			
	100000 samples		1000 samples	
	$\underline{z}_1 \in$	$\overline{z}_1 \in$	$\underline{z}_1 \in$	$\overline{z}_1 \in$
1	$[0.3796, 0.385]$	$[0.5935, 0.6007]$	$[0.3796, 0.3850]$	$[0.5935, 0.6007]$
2	$[0.3796, 0.3807]$	$[0.5993, 0.6007]$	$[0.3796, 0.3822]$	$[0.5971, 0.6007]$
3	$[0.3796, 0.3801]$	$[0.5999, 0.6006]$	$[0.3796, 0.3808]$	$[0.5987, 0.6008]$
4	—	—	$[0.3797, 0.3803]$	$[0.5995, 0.6007]$
$[z_1]$	$[0.38, 0.6]$		$[0.38, 0.6]$	
Time (s)	5.1388		5.5936	
Iterations	100 samples		10 samples	
	$\underline{z}_1 \in$	$\overline{z}_1 \in$	$\underline{z}_1 \in$	$\overline{z}_1 \in$
1	$[0.3796, 0.385]$	$[0.5935, 0.6007]$	$[0.3796, 0.385]$	$[0.5935, 0.6007]$
2	$[0.3796, 0.3833]$	$[0.5965, 0.6007]$	$[0.3796, 0.384]$	$[0.5945, 0.6007]$
3	$[0.3797, 0.3817]$	$[0.5982, 0.6007]$	$[0.3796, 0.3836]$	$[0.595, 0.6007]$
4	$[0.3797, 0.3814]$	$[0.5989, 0.6007]$	$[0.3797, 0.3829]$	$[0.5971, 0.6007]$
5	$[0.3797, 0.3808]$	$[0.5995, 0.6006]$	$[0.3797, 0.3811]$	$[0.5977, 0.6007]$
6	$[0.3797, 0.3805]$	$[0.5996, 0.6006]$	$[0.3797, 0.3808]$	$[0.5978, 0.6006]$
7	—	—	$[0.3797, 0.3805]$	$[0.5988, 0.6006]$
$[z_1]$	$[0.38, 0.6]$		$[0.38, 0.6]$	
Time (s)	6.0616		9.511	

It may be observed from Table 2 that the SIVIA Monte-Carlo method iteratively converge to exact enclosure $[0.38, 0.6]$ (upto two decimals) even for less sample values viz. 100 and 10 respectively. Also, it may be noted that the iterative enclosures converge to exact bound though the computational time increases from 5.1388 to 9.511 seconds for different samples ranging from 100000 to 10 respectively. From Table 2, the proposed method seems appealing as even for less sample values the convergent or exact solution bound is achieved. Many practical application problems do not yield sufficient data and sometimes availability of large data are cost effective, in such cases the proposed method may be used to obtain exact enclosure and the increase in computational time may be neglected.

Table 3. Interval enclosure of z_2

Iterations	SIVIA (0.001 precision) and Monte Carlo samples			
	100000 samples		1000 samples	
	$\underline{z}_2 \in$	$\overline{z}_2 \in$	$\underline{z}_2 \in$	$\overline{z}_2 \in$
1	$[0.4565, 0.4617]$	$[0.6825, 0.6889]$	$[0.4565, 0.4617]$	$[0.6825, 0.6889]$
2	$[0.4565, 0.4581]$	$[0.6869, 0.6887]$	$[0.4565, 0.4591]$	$[0.6859, 0.6887]$
3	$[0.4566, 0.4575]$	$[0.6872, 0.6886]$	$[0.4565, 0.4579]$	$[0.6864, 0.6887]$
4	—	—	$[0.4565, 0.4577]$	$[0.6867, 0.6887]$
5	—	—	$[0.4565, 0.4575]$	$[0.6869, 0.6887]$
6	—	—	$[0.4565, 0.4574]$	$[0.687, 0.6889]$
$[z_2]$	$[0.46, 0.69]$		$[0.46, 0.69]$	
Time (s)	6.7797		7.5464	
Iterations	100 samples		10 samples	
	$\underline{z}_2 \in$	$\overline{z}_2 \in$	$\underline{z}_2 \in$	$\overline{z}_2 \in$
1	$[0.4565, 0.4617]$	$[0.6825, 0.6889]$	$[0.3796, 0.385]$	$[0.5935, 0.6007]$
2	$[0.4565, 0.4609]$	$[0.6826, 0.6887]$	$[0.4565, 0.4609]$	$[0.6854, 0.6887]$
3	$[0.4565, 0.4599]$	$[0.6858, 0.6887]$	$[0.4565, 0.4601]$	$[0.6828, 0.6886]$
4	$[0.4566, 0.4578]$	$[0.6863, 0.6885]$	$[0.4565, 0.4589]$	$[0.6836, 0.6886]$
5	$[0.4566, 0.4577]$	$[0.6864, 0.6891]$	$[0.4565, 0.4584]$	$[0.6839, 0.6885]$
6	$[0.4566, 0.4576]$	$[0.6866, 0.689]$	$[0.4565, 0.4581]$	$[0.685, 0.6885]$
7	$[0.4566, 0.4574]$	$[0.6867, 0.689]$	$[0.4565, 0.4577]$	$[0.6856, 0.6885]$
8	—	—	$[0.4565, 0.4575]$	$[0.686, 0.6885]$
9	—	—	$[0.4566, 0.4574]$	$[0.6865, 0.6884]$
$[z_2]$	$[0.46, 0.69]$		$[0.46, 0.69]$	
Time (s)	9.2454		24.847	

Similar observations of exact enclosure convergence may be found in Table 3 with respect to different sample values. Moreover, due to complexity of the constraint (4), the required computational time 24.847 seconds for $[z_2]$ is comparatively higher than time 9.511 seconds required for $[z_1]$. Further, a nonlinear differential equations with respect to dynamic problems has been considered in next section for verification and effectiveness of SIVIA Monte-Carlo approach.

4 Nonlinear Oscillator

Sometimes, dynamic problems are governed by $m\ddot{x} + c\dot{x} + kx = f(t)$ having nonlinear stiffness $(k_1x + k_2x^2 + \ldots)$ which result to nonlinear differential equations (nonlinear oscillators). In case of uncertain nonlinear oscillators, the SIVIA Monte-Carlo method has been implemented using nonlinear equations obtained based on Runge-Kutta 4^{th} order [8, 5]. As such, the enclosure obtained in present section yield a validated enclosure rather than the tighest bound. There exists several validated interval methods and solvers in DynIbex [15] and CAPD [1] libraries for obtaining validated bounds.

Example 4: Consider Van der Pol equation (crisp or precise case given in Akbari et al. [2]),

$$\ddot{x}(t) + 0.15\left(1 - x^2\right)\dot{x} + 1.44x = 0 \tag{12}$$

subject to uncertain initial conditions $x(0) \in [0.1, 0.3]$ and $\dot{x}(0) = 0$.
The system of first-order differential equation corresponding to (12) is obtained as

$$\begin{aligned}\dot{u} =& v = f_u(t, u, v)\\ \dot{v} =& 0.15(u^2 - 1)v - 1.44u = f_v(t, u, v)\end{aligned}$$

subject to initial conditions $u(0) \in [0.1, 0.3]$ and $v(0) = 0$. Using Runge-Kutta fourth-order (RK4) method, the nonlinear constraints involved in computation of (12) are

$$u_{n+1} = u_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{13}$$

$$v_{n+1} = v_n + \frac{h}{6}(l_1 + 2l_2 + 2l_3 + l_4) \tag{14}$$

where,

$$k_1 = hf_u(t_n, u_n, v_n), l_1 = hf_v(t_n, u_n, v_n),$$

$$k_2 = hf_u\left(t_n + \frac{h}{2}, u_n + \frac{k_1}{2}, v_n + \frac{l_1}{2}\right), l_2 = hf_v\left(t_n + \frac{h}{2}, u_n + \frac{k_1}{2}, v_n + \frac{l_1}{2}\right),$$

$$k_3 = hf_u\left(t_n + \frac{h}{2}, u_n + \frac{k_2}{2}, v_n + \frac{l_2}{2}\right), l_3 = hf_v\left(t_n + \frac{h}{2}, u_n + \frac{k_2}{2}, v_n + \frac{l_2}{2}\right),$$

$$k_4 = hf_u\left(t_n + h, u_n + k_3, v_n + l_3\right) \text{ and } l_4 = hf_v\left(t_n + h, u_n + k_3, v_n + l_3\right).$$

Using Algorithm 1 with respect to constraints (13) and (14), the validated enclosure of $x(t)|_{t=T}$ is obtained and incorporated in Table 4 and Fig. 5.

Table 4. Instantaneous solution enclosure of $x(t)|_{t=T}$

T	Enclosures	
	$[x](T) = [u](T)$	$[v](T)$
0.1	[0.0993,0.2979]	[-0.0428,-0.0143]
0.2	[0.0972,0.2915]	[-0.0844,-0.0281]
0.3	[0.0937,0.281]	[-0.1242,-0.0413]

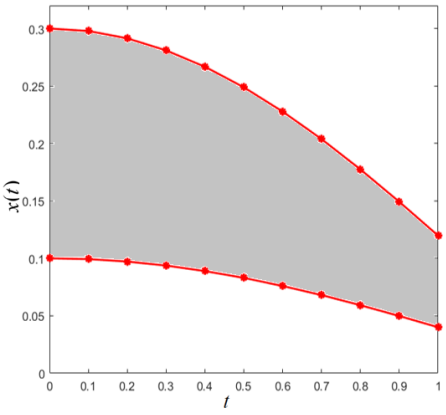

Fig. 5. Enclosure of $x(t)$ for $t \in [0, 1]$

5 Conclusion

Generally, dynamical systems occurring in various science and engineering problems are governed by nonlinear equations or nonlinear differential equations. An iterative procedure based on set inversion via interval analysis and Monte-Carlo method has been proposed for computation of exact enclosure of nonlinear equations having imprecise or uncertain variables. The effectiveness of SIVIA Monte-Carlo method has also been verified based on the considered test problems that yield exact enclosures even with respect to very less sample values. So, the method may be well implemented in computation of exact enclosures of various nonlinear equations irrespective of the dependency problem. Further, the method has also been implemented to compute validated enclosure in case of Van der Pol oscillator. Accordingly, the method may be applied to other practical nonlinear system of equations involving uncertain parameters.

Appendix

Forward-backward contractor: The forward-backward contractor is based on constraint $f(\mathbf{x}) = 0$ where $\mathbf{x} \in [\mathbf{x}]$ and $[\mathbf{x}] \in \mathbb{IR}^n$ which is illustrated using an example problem.

Example A1: Perform forward-backward contractor subject to constraint $w = 2u + v$ where, $[w] = [3, 20]$, $[u] = [-10, 5]$ and $[v] = [0, 4]$.

Here, the constraint $w = 2u + v$ may be expressed in terms of function f as $f(u, v, w) = w - 2u - v$. Further, the possible different forms of the constraint may be written are:

$$u = \frac{w - v}{2}$$

$$v = w - 2u$$

$$w = 2u + v$$

The forward-backward steps are then followed with respect to classical interval computations mentioned in Section 2 as:

$$[u] \cap \left(\frac{[w] - [v]}{2} \right) = [-10, 5] \cap \left(\frac{[3, 20] - [0, 4]}{2} \right) = [-0.5, 5]$$

$$[v] \cap ([w] - 2[u]) = [0, 4] \cap ([3, 20] - 2[-0.5, 5]) = [0, 4]$$

$$[w] \cap (2[u] + [v]) = [3, 20] \cap (2[-0.5, 5] + [0, 4]) = [3, 14]$$

As such, the new interval bounds are $[w] = [3, 14]$, $[u] = [-0.5, 5]$ and $[v] = [0, 4]$.

Fixed-point contractor: A fixed-point contraction associated with ψ is implemented with respect to the constraint $f(\mathbf{x}) = 0$ as $\mathbf{x} = \psi([\mathbf{x}])$, where $\mathbf{x} \in [\mathbf{x}] \in \mathbb{IR}^n$. The fixed-point contractor with respect to constraint $u^2 + 2u + 1 = 0$ is performed as

$$\begin{aligned} u \in [u] \text{ and } u = \psi(u) &\implies u \in [u] \text{ and } u \in \psi([u]) \\ &\implies u \in [u] \cap [\psi]([u]) \end{aligned}$$

In case of implementation of forward-backward contractor along with fixed point contractor helps in computation of forward-backward contractor until the fixed interval is reached.

References

1. Computer assisted proofs in dynamics group (capd), http://capd.ii.uj.edu.pl/
2. Akbari, M., Ganji, D., Majidian, A., Ahmadi, A.: Solving nonlinear differential equations of vanderpol, rayleigh and duffing by agm. Frontiers of Mechanical Engineering **9**(2), 177–190 (2014)
3. Alefeld, G., Herzberger, J.: Introduction to Interval Computation. Academic Press, London (2012)
4. Chabert, G., Jaulin, L.: Contractor programming. Artificial Intelligence **173**, 1079–1100 (2009)
5. Chakraverty, S., Mahato, N.R., Karunakar, P., Rao, T.D.: Advanced numerical and semi-analytical methods for differential equations, (in press)
6. Desrochers, B.: Pyibex, http://benensta.github.io/pyIbex/sphinx/index.html
7. Dezert, J., Han, D., Tacnet, J.M.: Multi-criteria decision-making with imprecise scores and bf-topsis. In: Information Fusion (Fusion), 2017 20th International Conference on. pp. 1–8. IEEE (2017)
8. Gerald, C.F.: Applied numerical analysis. Pearson Education India (2004)
9. Hansen, E.R.: A generalized interval arithmetic. In: Interval mathematics, pp. 7–18. Springer (1975)
10. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied interval analysis: with examples in parameter and state estimation, robust control and robotics, vol. 1. Springer-Verlag, London (2001)

11. Krämer, W.: Generalized intervals and the dependency problem. In: PAMM: Proceedings in Applied Mathematics and Mechanics. vol. 6, pp. 683–684. Wiley Online Library (2006)
12. Moore, R.E.: Interval arithmetic and automatic error analysis in digital computing. Ph. D. Dissertation, Department of Mathematics, Stanford University (1962)
13. Moore, R.E.: Methods and applications of interval analysis, vol. 2. Siam (1979)
14. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM Publications, Philadelphia, PA (2009)
15. Sandretto, J.A.d., Chapoutot, A., Mullier, O.: Dynibex, http://perso.ensta-paristech.fr/ chapoutot/dynibex/
16. Stolfi, J., De Figueiredo, L.: An introduction to affine arithmetic. Trends in Applied and Computational Mathematics **4**(3), 297–312 (2003)