Cairo University

SOLVING MULTIDIMENSIONAL NONLINEAR PERTURBED PROBLEMS USING INTERVAL ARITHMETIC

By

Islam Refaat Kamel Taha

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Engineering Mathematics

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
June - 2013

SOLVING MULTIDIMENSIONAL NONLINEAR PERTURBED PROBLEMS USING INTERVAL ARITHMETIC

By
Islam Refaat Kamel Taha

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Engineering Mathematics

Under the Supervision of

Assoc. Prof. Dr. Maha A. Hassanein

Associate Professor
Engineering Mathematics and Physics Department
Faculty of Engineering, Cairo University

Assoc. Prof. Dr. Hossam A. H. Fahmy

Associate Professor
Electronics and Communications Engineering Department
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
June - 2013

SOLVING MULTIDIMENSIONAL NONLINEAR PERTURBED PROBLEMS USING INTERVAL ARITHMETIC

By
Islam Refaat Kamel Taha

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Engineering Mathematics

Approved by the
Examining Committee

____
Prof. Dr. First S. Name, External Examiner

____
Prof. Dr. Second E. Name, Internal Examiner

____
Assoc. Prof. Dr. Maha A. Hassanein, Thesis Main Advisor

____
Assoc. Prof. Dr. Hossam A. H. Fahmy, Member

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
June – 2013

Engineer's Name:	Islam Refaat Kamel Taha
Date of Birth:	1/2/1988
Nationality:	Egyptian
E-mail:	i.r.kamel@ieee.org
Phone:	(+2)01004067053
Address:	El-Sheikh Zayed, Giza
Registration Date:	…./…./……..
Awarding Date:	…./…./……..
Degree:	Master of Science
Department:	Engineering Mathematics and Physics

Insert photo here

Supervisors:

Assoc. Prof. Maha A. Hassanein
Assoc. Prof. Hossam A. H. Fahmy

Examiners:

Prof. ………………… (External examiner)
Prof. ………………… (Internal examiner)
Assoc. Porf. Maha A. Hassanein (Thesis main advisor)
Assoc. Porf. Hossam A. H. Fahmy (Member)

Title of Thesis:

Solving multidimensional nonlinear perturbed problems using interval arithmetic

Key Words:
Interval Arithmetic; Perturbed Problems; Nonlinear Systems; Interval Newton Method

Summary:

…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………
…………………………………………………………………………………

Acknowledgments

I would like to express my deepest appreciation to all those who provided me the possibility to complete this thesis. A special gratitude I give to my supervisor Assoc. Prof. Dr. Maha Amin Hassanein, who showed a lot of interest in guiding my work. She gave me a lot of support and encouragement to complete this thesis.

Furthermore I would also like to acknowledge with much appreciation my supervisor Assoc. Prof. Dr. Hossam A. H. Fahmy for encouraging my interests in interval arithmetic and for his fruitful discussions and valuable suggestions.

Finally, my greatest debt is to my family who has always been supportive and encouraging.

Table of Contents

List of Tables

List of Figures

Nomenclature

a, b	Real constant symbols (with or without subscripts)
x, y, z	Real variable symbols (with or without subscripts)
$\mathbb{R}$	The set of real numbers
X, Y, P	Interval variable symbols (with or without subscripts)
$\mathbb{IR}$	The set of real interval numbers
$\underline{X}$	The infimum of an interval number X
$\overline{X}$	The supremum of an interval number X
$m, m(X), mid(X)$	The mid-point of an interval number X
$r, r(X), rad(X)$	The radius of an interval number X
$w(X)$	The width of an interval number X
$\cap$	The set intersection operator
$\cup$	The set union operator
$\underline{\cup}$	The interval hull operator
$\emptyset$	The empty set
$\lvert X \rvert$	Greatest absolute value of an interval number X
$<, >, \leq, \geq$	Ordering relations
$\subseteq$	The set inclusion relation
$\circ \in \{+, -, \times, \div\}$	A binary algebraic operator
$\lVert X \rVert$	Maximum norm for interval vectors and matrices
f, g	Real-valued function
$\bigcup_{x \in X} \{g(x)\}$	The finitary set union
$\mathbb{IR}^n$	The set of real interval vectors
$\mathbb{IR}^{n \times n}$	The set of real interval matrices
F, G	Interval-valued function (the interval extension of f)
$D_i F$	The interval extension of $\frac{\partial f}{\partial x_i}$
$F_{mv}(X)$	The mean value extension of f on X
$\{X_k\}$	Sequence of intervals
N	Natural number
$d(X, Y)$	The distance (*metric*) between two interval numbers
ε	A small real number
$\bigcap_{k=1}^{\infty} X_k$	The finitary set intersection
S	Set variable symbol
$\exists$	There exists symbol
$\rho(X)$	Spectral radius of matrix X
$\tilde{x}$	A real number in an interval X
$\check{A}$	Midpoint of interval vector or matrix A

$F'(X)$	The interval extension of $f'(x)$
x^*	Zero of a real-valued function
X^*	Zero set of an interval function
$N(X)$	Interval Newton operator
$int(X)$	The interior of an interval X
$K(X)$	Krawczyk operator
$\Gamma(A, B, X)$	Interval Gauss-Seidel operator in one-dimension
$\Gamma(A, B, X, Z)$	Multidimensional interval Gauss-Seidel operator
J	Jacobian or approximate Jacobian of F
D_f	Domain of f
$\mathrm{H}_{f,\mathrm{P}}(\mathrm{X}, \mathrm{Z})$	Parametric Hansen-Sengupta operator
$N(X, P)$	Parametric Interval Newton operator
$K(X, P)$	Parametric Krawczyk operator
$MN(X, Y, P)$	Parametric modified interval Newton operator (parametric two-stage interval Newton operator)

Abstract

In this thesis, we are interested in solving nonlinear systems of equations with inexact data, denoted by perturbed nonlinear systems, using interval arithmetic methods. Such perturbed problems appear in many engineering applications to study the sensitivity of design parameters to perturbations resulting either during manufacturing or during floating-point computations. The main scope of the dissertation is to present Newton-like interval methods to solve perturbed nonlinear systems by adapting existing methods for solving real-valued nonlinear systems to be used in solving perturbed problems. The existence and convergence of a solution to the perturbed nonlinear systems are derived and given in the dissertation.

In achieving this, we give a brief introduction to the interval arithmetic and the solution of interval linear systems and real-valued nonlinear problems using interval methods. Well-known methods in the literature for solving the nonlinear problems: interval Newton (INM), Hansen-Sengupta, and Krawczyk methods are presented. For each method a version for solving the perturbed problems is given. A discussion of the difficulties encountered and convergence of each method is given. Furthermore, a modified version of the INM, denoted by the two-stage INM, is derived. The two-stage method has the advantage of reducing the computational time required to find a solution. To illustrate the effectiveness of interval methods, we apply it to test problems from the literature by introducing perturbations into these problems. We observed that the width of the solution depends on the width of the perturbations. The two-stage INM is compared with the other interval methods under consideration. Regarding the time consumption, it gives an improvement over the other methods. We also compare the interval arithmetic solution with that of the Monte Carlo and conclude the superior performance of the interval algorithms over Monte Carlo methods with respect to the consumed time and the accuracy of the solution set obtained.

Chapter 1 : Introduction

1.1. Motivation

Many real problems are simulated and modeled using nonlinear systems of equations. For example, the sophisticated circuits which are modeled by circuit simulators as *nonlinear systems* with thousands and millions of parameters. Some of these parameters are vulnerable to temperature changes, manufacturing tolerances, and other effects. Those *perturbed parameters* should be taken into consideration while modeling and simulating the corresponding nonlinear systems. This simulation is commonly known as sensitivity analysis.

There are numerous methods and algorithms used in the sensitivity analysis which compute approximations to the solution in floating-point arithmetic [62], [33], and [47] like the exhaustive sampling and Monte Carlo methods [3]. However, usually it is not clear how good these approximations are, or if there exists a unique solution at all. In general, it is not possible to answer these questions with mathematical rigor if only floating-point approximations are used.

The use of self-verified methods can lead to more reliable results. Verified computing provides an interval result that surely contains the correct result. Like that the algorithm also proves the existence and uniqueness of the solution of the problem. The algorithm will, in general, succeed in finding an enclosure of the correct solution. If the solution is not found, the algorithm will let the user know. One possibility to implement verified computing is using interval arithmetic combined with suitable algorithms.

The use of verified computing makes it possible to find the correct result. However, finding the verified result often increases the execution time dramatically. But in the nonlinear systems with interval parameters, the use of verified computing is more suitable as its performance is better. This thesis shows that the execution time of some verified algorithms is much smaller than the execution time of algorithms that do not use this concept. Moreover, a modified interval Newton algorithm will be presented and shown to have better performance than other interval algorithms.

1.2. Outline of the Thesis

The thesis is structured, in six chapters, as follows:

This introductory chapter has formulated the motivation for this thesis. It has also provided an overview of the main problem which is under test in this thesis.

Chapter 2 provides a bit of perspective on the field of interval arithmetic. It also introduces some definitions, notations, and basic facts that will be used through this thesis. Section 2.8 gives a brief description of the commonly used interval computing software.

The first section of Chapter 3 treats the basics of interval-valued functions. Section 3.2 introduces theory and practice regarding the convergence of interval sequences. An overview of the interval linear systems is presented in section 3.3. Finally, an introductory to the next chapter is presented in section 3.4.

Chapter 4 starts with solving univariate nonlinear systems. Section 4.2 gives a survey on existing interval methods that are used in solving and bounding the solution of nonlinear systems. In section 4.4, we carefully use interval methods to solve perturbed nonlinear systems which are defined in section 4.3. We prove the convergence of the interval Newton method (INM) for perturbed problems in subsection 4.4.1. We introduce a modified version of INM for perturbed nonlinear systems in section 4.5. In the end of this chapter, numerical examples are solved to show that the verified algorithms, specially the new proposed algorithm, are faster than the traditional methods. A comparison between the different interval methods and the traditional methods is also discussed.

In Chapter 5, more practical examples are solved to illustrate the applicability of the algorithms discussed in Chapter 4.

Finally, the thesis closes with Chapter 6, which explicitly delineates the contributions of this research and outlines some directions for future research.

Chapter 2 : Introduction to Interval Arithmetic

Interval arithmetic is a method developed by mathematicians since the 1950s and 1960s to put bounds on rounding errors and measurement errors in mathematical computation. Numerical methods based on interval arithmetic are developed that yield reliable results in which each value is represented as a range of possibilities. For example, instead of estimating the height of someone using standard arithmetic as 2.0 meters, using interval arithmetic we might be certain that that person is somewhere between 1.97 and 2.03 meters. Archimedes was able to bracket π by taking a circle and considering inscribed and circumscribed polygons. Increasing the numbers of polygonal sides, he obtained both an increasing sequence of lower bounds and a decreasing sequence of upper bounds for this irrational number. That is exactly what is used in interval arithmetic to represent numbers. The motivation of this chapter is to give an insight of the basic definitions and properties of interval numbers which will be used throughout the thesis (For more details about the basics of the interval arithmetic, the reader may consult [44], [5], [35], [38], [39], and [58]). Moreover, a survey on the common and popular interval computation software is presented in section 2.8.

2.1. Basic Terms and Concepts

Using the ordered pair $[a, b]$ of computer numbers to represent an interval of real numbers $a \leq x \leq b$, an arithmetic for intervals and interval valued extensions of functions is defined and commonly used in computing. In this way, an interval $[a, b]$ has a dual nature. It is a new kind of number pair, and it represents a *set* $[a, b] = \{x \in \mathbb{R}: a \leq x \leq b\}$.

2.1.1. Inf-sup and Mid-rad Representations

Throughout this dissertation, we will denote intervals and their endpoints by capital letters. The left and right endpoints of an interval X will be denoted by $\underline{X}$ and $\overline{X}$, respectively. Thus,

$$X = [\underline{X}, \overline{X}], \tag{2.1}$$

where $\underline{X} = \min(X) \in \mathbb{R}$ and $\overline{X} = \max(X) \in \mathbb{R}$ are called the upper and lower bounds of X (also called supremum and infimum respectively). The notation used in (2.1) to represent the interval X is usually called inf-sup representation. There is another representation which is used to represent intervals: mid-rad representation. In the mid-rad representation, we can express the interval X as:

$$X =< m, r >, \tag{2.2}$$

where $m = \frac{max(X)+\min(X)}{2}$ which represents the mid-point of the interval X and $r = \max(X) - mid(X) = mid(X) - \min(X)$.

This mid-rad representation is useful when we employ an interval to describe a quantity in terms of its measured value m and a measurement uncertainty of no more than $\pm r$. However, the mid-rad representation fails to represent some intervals, specially the unbounded intervals (the intervals whose infimum equals $-\infty$ or supremum equals ∞).

There are special forms of intervals such as the degenerate ones which contain a single real number. An interval X is said to be degenerate if $\underline{X} = \overline{X}$ and is defined as $[x, x]$ with the real number x. For instance, we may write such equations as

$$0 = [0,0] \tag{2.3}$$

On the other hand the interval equality is defined as follows: we said that the two intervals X and Y are equal if they are defined by the same sets, i.e. their corresponding endpoints are equal:

$$X = Y \ iff \ \underline{X} = \underline{Y} \ and \ \overline{X} = \overline{Y} \tag{2.4}$$

2.1.2. Intersection, Union, and Interval Hull

The intersection of two intervals X and Y is defined as follows

$$\begin{aligned} X \cap Y &= \{z: z \in X \ and \ z \in Y\} \\ &= \left[\max\left(\underline{X}, \underline{Y}\right), \min\left(\overline{X}, \overline{Y}\right)\right] \end{aligned} \tag{2.5}$$

The intersection $X \cap Y$ is either an interval which is defined by the previous equation or empty. The latter case occurs if either $\overline{Y} < \underline{X}$ or $\overline{X} < \underline{Y}$. In this case we let $\emptyset$ denote the empty set and write

$$X \cap Y = \emptyset, \tag{2.6}$$

indicating that X and Y have no points in common. In case of the existence of the intersection between X and Y, the union of X and Y is also an interval:

$$\begin{aligned} X \cup Y &= \{z: x \in X \ or \ z \in Y\} \\ &= \left[\min\left(\underline{X}, \underline{Y}\right), \max\left(\overline{X}, \overline{Y}\right)\right] \end{aligned} \tag{2.7}$$

In general, it is not necessary that one is able to represent the union of two intervals as an interval. However, the interval hull of two intervals, defined by

$$X \underline{\cup} Y = \left[\min\left(\underline{X}, \underline{Y}\right), \max\left(\overline{X}, \overline{Y}\right)\right], \tag{2.8}$$

is always an interval and can be used in interval computations. We have

$$X \cup Y \subseteq X \underline{\cup} Y \tag{2.9}$$

for any two intervals X and Y.

Example 2.1 If $X = [-3,1]$ and $Y = [2,5]$, then $X \cap Y = \emptyset$ and $X \underline{\cup} Y = [-3,5]$. Although $X \cup Y$ is a disconnected set that cannot be expressed as an interval, relation (2.7) still holds. Information is lost when we replace $X \cup Y$ with $X \underline{\cup} Y$, but $X \underline{\cup} Y$ is easier to work with, and the lost information is sometimes not critical.

Intersection plays a key role in interval analysis. For instance, if we have two distinct intervals containing a result of interest, then we can obtain a narrower interval ,which also contains the result, from the intersection of the initial intervals**.**

2.1.3. Width, Absolute Value, and Midpoint of an Interval Number

The width of an interval X is defined and denoted by

$$w(X) = \overline{X} - \underline{X} \tag{2.10}$$

Thus the width of a degenerate interval number is zero, that is

$$w([x,x]) = x - x = 0$$

The absolute value of an interval X, denoted by |X|, is the maximum of the absolute values of its endpoints:

$$|X| = \max\{|\underline{X}|, |\overline{X}|\} \tag{2.11}$$

Note that $|x| \leq |X|$ for every $x \in X$.

The midpoint of an interval X is given by

$$m(X) = \frac{1}{2}(\underline{X} + \overline{X}) \tag{2.12}$$

Hence, the midpoint of a degenerate interval number $x = [x,x]$ is x. We observe that any interval X can be expressed in terms of its midpoint and width as follows:

$$X = m(X) + \left[-\frac{1}{2}w(X), \frac{1}{2}w(X)\right] = m(X) + \frac{1}{2}w(X)[-1,1] \tag{2.13}$$

As an illustration of the above definitions refer to Figure 2.1 and Example 2.2.

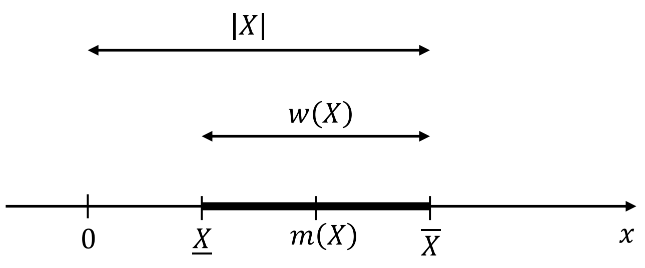

Figure 2.1: Width, absolute value, and midpoint of an interval

Example 2.2 Let $X = [0,3]$ and $Y = [-2,2]$. The intersection and union of X and Y are the intervals

$$X \cap Y = [max(0,-2)\,, min(3,2)] = [0,2]$$
$$X \cup Y = [min(0,-2)\,, max(3,2)] = [-2,3]$$

We have $w(X) = 3$, $w(Y) = 2$, and $|X| = max(0,3) = 3$. The midpoint of Y is $m(Y) = 0$. Using (2.13), we can write $Y = 1.5 + [-1.5,1.5]$.

After defining the width and midpoint of intervals, we can rewrite the mid-rad representation (2.2) of an interval X as follows:

$$X =< m, r >, \qquad where\; m = m(X)\; and\; r = \frac{w(X)}{2} \tag{2.14}$$

2.1.4. Order Relations on Intervals

Interval numbers are sets of real numbers. Thus ordering relations for interval numbers extend those of real numbers. For example the ordering relation $<$ can be extended and applied to intervals as follows

$$X < Y\; iff\; \overline{X} < \underline{Y}. \tag{2.15}$$

For instance, the interval $[1,3] < [4,6]$ is satisfied. This ordering relation for real numbers is known to be transitive, i.e. if $a < b$ and $b < c$, then $a < c$ for any $a, b\; and\; c \in \mathbb{R}$. The transitive property is still valid for intervals

$$X < Y\; and\; Y < Z \;\Rightarrow X < Z \tag{2.16}$$

Recalling the interval definition of the zero (2.3) and the ordering relation (2.15), an interval X is said to be positive if $\underline{X} > 0$ or negative if $\overline{X} < 0$. That is, we have $X > 0$ if $x > 0$ for all $x \in X$.

Another ordering is the set inclusion, $\subseteq$, for intervals which is defined by:

$$X \subseteq Y\; iff\; \underline{Y} \leq \underline{X}\; and\; \overline{X} \leq \overline{Y} \tag{2.17}$$

For instance, we have $[5,6] \subseteq [4,6]$. Both $\subseteq$ and $<$ are partial orderings on $\mathbb{IR}$, not every pair of intervals in $\mathbb{IR}$ are comparable. For example, if X and Y are overlapping intervals such as $X = [5,7]\; and\; Y = [4,6]$, then neither X is contained in Y nor Y is contained in X. However, $X \cap Y = [4,5]$, is contained in both X and Y.

2.2. Algebraic Operations for Interval Numbers

The basic algebraic operations for real numbers can be extended for interval numbers. In this subsection, we present some basic arithmetic operations for intervals by extending the arithmetic operations for real numbers. For instance, the result of adding two intervals is an interval containing the sums of all pairs of numbers, one from each of the two initial intervals. By definition then, the sum of two intervals X and Y is given by the set

$$X + Y = \{x + y : x \in X, y \in Y\}. \tag{2.18}$$

The difference of two intervals X and Y is given by the set

$$X - Y = \{x - y : x \in X, y \in Y\}. \tag{2.19}$$

The product of two intervals X and Y is given by

$$X.Y = \{x.y : x \in X, y \in Y\}. \tag{2.20}$$

Finally, the division X/Y is given by the set

$$\frac{X}{Y} = \left\{\frac{x}{y} : x \in X, y \in Y\right\} \tag{2.21}$$

provided that $0 \notin Y$. Since all algebraic operations of interval numbers have the same general form, they can be summarized by the following definition

$$X \circ Y = \{x \circ y : x \in X, y \in Y\}, \tag{2.22}$$

where $\circ$ stands for any of the four binary operations introduced above.

In the following subsections, we present endpoint formulas for the four binary operations introduced above.

2.2.1. Addition on $\mathbb{IR}$

Since $x \in X$ means that $\underline{X} \le x \le \overline{X}$ and $y \in Y$ means that $\underline{Y} \le y \le \overline{Y}$, we see by addition of those inequalities that the numerical sums $x + y \in X + Y$ must satisfy

$$\underline{X} + \underline{Y} \le x + y \le \overline{X} + \overline{Y}.$$

Hence, the formula

$$X + Y = \left[\underline{X} + \underline{Y}, \overline{X} + \overline{Y}\right] \tag{2.23}$$

can be used to implement (2.18).

Example 2.3 Let $X = [0,3]$ and $Y = [-2,2]$ as in Example 2.2. Then $X + Y = [0 + (-2), 3 + 2] = [-2,5]$. This is not the same as $X \cup Y = [-2,3]$.

2.2.2. Subtraction on $\mathbb{IR}$

Similar expressions to (2.23) can be derived for the remaining arithmetic operations. For subtraction we add the inequalities $\underline{X} \le x \le \overline{X}$ and $-\overline{Y} \le -y \le -\underline{Y}$ to get $\underline{X} - \overline{Y} \le x - y \le \overline{X} - \underline{Y}$. It follows that

$$X - Y = \left[\underline{X} - \overline{Y}, \overline{X} - \underline{Y}\right] \tag{2.24}$$

Note that $X - Y = X + (-Y)$, where $-Y = \left[-\overline{Y}, -\underline{Y}\right] = \{y : -y \in Y\}$.

Example 2.4 If $X = [-2,1]$ and $Y = [-1,3]$, then $-Y = [-3,1]$ and $X - Y = X + (-Y) = [-5,2]$.

2.2.3. Multiplication on $\mathbb{IR}$

The product of two intervals X and Y is given by

$$XY = [min\,S\,, max\,S], \;\; where\; S = \{\underline{XY}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \underline{XY}\}. \qquad (2.25)$$

Since the multiplication on $\mathbb{R}$ is continuous, it follows that the multiplication on $\mathbb{IR}$ is continuous and the product XY attains its maximum and minimum values.

Example 2.5 Let $X = [-2,1]$ and $Y = [-1,3]$. Then $S = \{-6, -1,2,3\}$ and $XY = [min\,S, max\,S] = [-6,3]$.
We also can evaluate the product of a scalar and an interval just by multiplying this scalar with the interval's infimum and supremum, for instance, $2\text{Y} = 2.[-1,3] = [-2,6]$.

The previous formula of interval multiplication requires calculating four real-number products; however, that is not necessary in all cases. Actually, by testing for the signs of the endpoints $\underline{X}$, $\overline{X}$, $\underline{Y}$, and $\overline{Y}$, the formula for the endpoints of the interval product can be broken into nine special cases. In eight of these cases, only two products need to be computed. Hence, this notion may be taken into consideration to improve the efficiency of any implementation of the interval product. Table 2.1 illustrates the different cases.

Table 2.1: Endpoint formulas for interval multiplication

Case	$\underline{\boldsymbol{X.Y}}$	$\overline{\boldsymbol{X.Y}}$
$0 \le \underline{X}$ and $0 \le \underline{Y}$	$\underline{X}.\underline{Y}$	$\overline{X}.\overline{Y}$
$\underline{X} < 0 < \overline{X}$ and $0 \le \underline{Y}$	$\underline{X}.\overline{Y}$	$\overline{X}.\overline{Y}$
$\overline{X} \le 0$ and $0 \le \underline{Y}$	$\underline{X}.\overline{Y}$	$\overline{X}.\underline{Y}$
$0 \le \underline{X}$ and $\underline{Y} < 0 < \overline{Y}$	$\overline{X}.\underline{Y}$	$\overline{X}.\overline{Y}$
$\overline{X} \le 0$ and $\underline{Y} < 0 < \overline{Y}$	$\underline{X}.\overline{Y}$	$\underline{X}.\underline{Y}$
$0 \le \underline{X}$ and $\overline{Y} \le 0$	$\overline{X}.\underline{Y}$	$\underline{X}.\overline{Y}$
$\underline{X} < 0 < \overline{X}$ and $\overline{Y} \le 0$	$\overline{X}.\underline{Y}$	$\underline{X}.\underline{Y}$
$\overline{X} \le 0$ and $\overline{Y} \le 0$	$\overline{X}.\overline{Y}$	$\underline{X}.\underline{Y}$
$\underline{X} < 0 < \overline{X}$ and $\underline{Y} < 0 < \overline{Y}$	$\min\{\underline{X}\overline{Y}, \overline{X}\underline{Y}\}$	$\max\{\underline{XY}, \overline{XY}\}$

2.2.4. Division on $\mathbb{IR}$

As with real numbers, division can be accomplished via multiplication by the reciprocal of the second operand. That is, we can implement equation (2.21) using

$$\frac{X}{Y} = X.\left(\frac{1}{Y}\right), \tag{2.26}$$

where

$$\frac{1}{Y} = \left\{\frac{1}{y} : y \in Y\right\} = \left[\frac{1}{\overline{Y}}, \frac{1}{\underline{Y}}\right], \tag{2.27}$$

assuming $0 \notin Y$.

Example 2.6 Let $X = [-1,2]$ and $Y = [5,7]$. Then $\frac{1}{Y} = \left[\frac{1}{7}, \frac{1}{5}\right]$ and $\frac{X}{Y} = X.\left(\frac{1}{Y}\right) = [-1,2].\left[\frac{1}{7}, \frac{1}{5}\right]$. Finally, $S = \left\{-\frac{1}{5}, -\frac{1}{7}, \frac{2}{7}, \frac{2}{5}\right\}$ and $\frac{X}{Y} = [min\, S, max\, S] = \left[-\frac{1}{5}, \frac{2}{5}\right]$.

2.2.4.1. Extended Interval Arithmetic

According to the *extended interval arithmetic* [26], the definition of interval division

$$[a,b]/[c,d] = [a,b](1/[c,d]),$$

where

$$1/[c,d] = \{1/y : y \in [c,d]\} \qquad (c < d \; any\; real\; numbers).$$

can be extended as follows to handle the case where $0 \in [c,d]$:

1. If $c = 0 < d$, then $1/[c,d] = [1/d, +\infty)$,
2. If $c < 0 < d$, then $1/[c,d] = (-\infty, 1/c] \cup [1/d, +\infty)$,
3. If $c < d = 0$, then $1/[c,d] = (-\infty, 1/c]$,

and if $0 \notin [c,d]$, the ordinary interval arithmetic can be used to implement the division.

Example 2.7 Let $X = [-10,5]$. Then $\frac{1}{X} = (-\infty, -1/10] \cup [1/5, +\infty)$.

2.3. Interval Vectors and Matrices

An n-dimensional interval vector, denoted by $(X_1, X_2, \dots, X_n)$ where $X_i \in \mathbb{IR}$, means an ordered n-tuple of intervals. Through the thesis, interval vectors will be denoted by capital letters such as $X \in \mathbb{IR}^n$. A two dimensional example is given for illustration.

Example 2.8 A two-dimensional interval vector

$$\mathrm{X} = (\mathrm{X}_1, \mathrm{X}_2) = \left(\left[\underline{\mathrm{X}_1}, \overline{\mathrm{X}_1}\right], \left[\underline{\mathrm{X}_2}, \overline{\mathrm{X}_2}\right]\right)$$

can be represented as a rectangle in the $\mathrm{x}_1\mathrm{x}_2$-plane, see Figure 2.2: it is the set of all points $(\mathrm{x}_1, \mathrm{x}_2)$ such that $\underline{\mathrm{X}_1} \le \mathrm{x}_1 \le \overline{\mathrm{X}_1}$, and $\underline{\mathrm{X}_2} \le \mathrm{x}_2 \le \overline{\mathrm{X}_2}$.

Many of the notions for point intervals mentioned in previous sections can be extended to interval vectors with suitable modifications. If $X = (X_1, X_2, \dots, X_n)$ and $Y = (Y_1, Y_2, \dots, Y_n)$ are n-dimensional interval vectors, then:

If $x = (x_1, x_2, \dots, x_n)$ is a real vector, then we write

$$x \in X \; if \; x_i \in X_i \; for \; i = 1,2, \dots, n. \tag{2.28}$$

The intersection of X and Y is empty if the intersection of any of their corresponding components is empty; i.e. $X \cap Y = \emptyset$, if $X_i \cap Y_i = \emptyset$ for some i. Otherwise, we have the following interval vector as a result of the intersection

$$X \cap Y = (X_1 \cap Y_1, X_2 \cap Y_2, \dots, X_n \cap Y_n). \tag{2.29}$$

We have the set inclusion for interval vectors which is given by:

$$X \subseteq Y \; if \; X_i \subseteq Y_i \; for \; i = 1,2, \dots, n. \tag{2.30}$$

We have many ordering relations in interval arithmetic. One of those relations is given by (For more details about the interval operators, see [6]):

$$X < Y \; if \; \overline{X}_i < \underline{Y_i} \; for \; i = 1,2, \dots, n. \tag{2.31}$$

The width of an interval vector X is the largest of the widths of any of its component intervals:

$$w(X) = \max\big(w(X_i)\big), \quad i = 1,2, \dots, n. \tag{2.32}$$

The midpoint of an interval vector X is

$$m(X) = \big(m(X_1), m(X_2), \dots, m(X_n)\big). \tag{2.33}$$

The norm of an interval vector X is

$$\|X\| = \max_i |X_i|. \tag{2.34}$$

where $|.|$ is given by (2.11). This serves as a generalization of absolute value.

Example 2.9 Consider the two-dimensional interval vector $X = ([1,3], [3,9])$. We have $w(X) = \max(3 - 1, 9 - 3) = 6$, $m(X) = \left(\frac{1+3}{2}, \frac{3+9}{2}\right) = (2,6)$ and $\|X\| = \max(\max(|1|, |3|), \max(|3|, |9|)) = \max(3,9) = 9$.

The width, norm, and midpoint of an interval vector in 2D are illustrated in Figure 2.2.

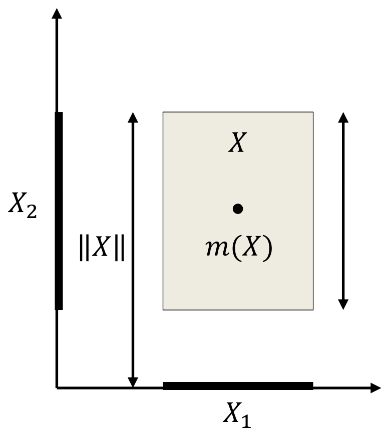

Figure 2.2: Width, norm, and midpoint of an interval vector X = (X_1,X_2)

2.4. Algebraic Properties of Interval Arithmetic

In a previous section, we introduced the definitions of the basic interval arithmetic operations. With proper understanding of the notation, the arithmetic operators summarized in (2.22). These definitions lead to a number of familiar looking algebraic properties which are presented in the following subsections.

2.4.1. Inclusion Monotonicity of Interval Arithmetic

Theorem 2.1 (Inclusion Monotonicity) Let X_1, X_2, Y_1 and Y_2 be interval numbers such that $X_1 \subseteq Y_1$ and $X_2 \subseteq Y_2$. Then for the interval operations $\circ \in \{+, \times\}$, we have

$$X_1 \circ X_2 \subseteq Y_1 \circ Y_2$$

Proof of Theorem 2.1 is presented in [6].

An important consequence is the following lemma.

Lemma 2.1 Let X and Y be interval numbers with $x \in X$ and $y \in Y$. Then for the interval operations $\circ \in \{+, \times\}$, we have

$$x \circ y \in X \circ Y$$

2.4.2. Commutativity and Associativity

Like the real-number operations, the interval addition and multiplication are commutative and associative. For any three intervals X, Y, and Z, we have:

$$\begin{array}{ll} X + Y = Y + X & (commutative\ addition) \\ X + (Y + Z) = (X + Y) + Z & (associative\ addition) \\ XY = YX & (commutative\ multiplication) \\ X(YZ) = (XY)Z & (associative\ multiplication) \end{array}$$

2.4.3. Additive and Multiplicative Identity Elements

Another common property between intervals and real numbers is the existence of additive and multiplicative identity elements. For any interval X, we have:

$$[0,0] + X = X + [0,0] = X \quad (Identity\ for\ addition)$$

$$[1,1].X = X.[1,1] = X \quad (Identity\ for\ multiplication)$$

The previous equations show that the degenerate intervals 0 and 1 are additive and multiplicative identity elements in the system of intervals, respectively. The degenerate interval 0 is also an absorbing element for the interval multiplication operation:

$$[0,0].X = X.[0,0] = [0,0]$$

2.4.4. Nonexistence of Inverse Elements

Unlike the ordinary arithmetic, additive and multiplicative inverses do not always exist for interval numbers. We must be caution that $-X$ is *not* an additive inverse for X in the system of intervals. That is

$$X + (-X) = [\underline{X}, \overline{X}] + [-\overline{X}, -\underline{X}] = [\underline{X} - \overline{X}, \overline{X} - \underline{X}] \neq [0,0]$$

But $X + (-X) = [0,0]\ iff\ \underline{X} = \overline{X}$, i.e. X is a degenrate interval, else

$$X - X = w(X)[-1,1] \tag{2.35}$$

Similarly, $\frac{1}{X}$ is not a multiplicative inverse for X. In general,

$$\begin{aligned} X/X \quad &= \begin{cases} [\underline{X}/\overline{X}, \overline{X}/\underline{X}] & if\ 0 < \underline{X} \\ [\overline{X}/\underline{X}, \underline{X}/\overline{X}] & if\ \overline{X} < 0 \end{cases} \\ &\neq [1,1] \end{aligned} \tag{2.36}$$

But $\frac{X}{X} = 1\ iff\ w(X) = 0$. To summarize, there is no additive or multiplicative inverses in interval arithmetic except for degenerate intervals. However, the inclusions $0 \in X - X$ and $1 \in X/X$ are always satisfied.

2.4.5. Subdistributivity

Another difference between the ordinary and interval arithmetic is that the distributive law does not always hold for intervals except in some cases provided below. The following counterexample proves the latter statement. If $X = [1, 2]$, $Y = [1, 3]$, and $Z = [-4, -1]$, then:

$$X(Y + Z) = [1, 2].([1, 3] + [-4, -1]) = [1, 2].[-3,2] = [-6,4]$$

whereas (2.35) gives

$$\begin{aligned} XY + XZ \quad &= [1,2].[1,3] + [1,2].[-4, -1] \\ &= [1,6] + [-8, -1] = [-7,5] \end{aligned}$$

That is in general $X(Y + Z) \neq XY + XZ$. However, interval arithmetic has the following *subdistributive law* [35]:

$$X(Y + Z) \subseteq XY + XZ \tag{2.37}$$

which can be seen in the previous example. Full distributivity holds in certain special cases. The first case occurs when X becomes a real number x, then we have

$$x(Y+Z) = xY + xZ \tag{2.38}$$

The second case occurs when the intervals Y and Z have the same sign, i.e. $YZ > 0$, then the interval multiplication can be distributed over the sum of those intervals:

$$X(Y+Z) = XY + XZ \tag{2.39}$$

We observe from the algebraic properties discussed previously two important properties peculiar to the classical theory of interval arithmetic:

Additive and multiplicative inverses do not always exist for interval numbers and there is no distributivity between addition and multiplication except for certain special cases. Thus, caution must be taken when using interval arithmetic to solve problems of uncertainty.

2.5. Outwardly Rounded Interval Arithmetic

In order to work effectively in a real-life implementation, intervals must be compatible with floating point computing. In practice, outward rounding is implemented at every interval operation rather than rounding to nearest which is commonly used in floating point computations. In *optimal outward rounding*, the outwardly rounded left endpoint is the closest machine number less than or equal to the exact left endpoint, and the outwardly rounded right endpoint is the closest machine number greater than or equal to the exact right endpoint. Those rounded endpoints are computed by rounding the exact endpoints down (towards negative infinity) and up (towards positive infinity), respectively.

Example 2.10 $f(x,y) = x + y$ for $x \in [0.1, 0.8]$ and $y \in [0.06, 0.08]$ are for example $[0.16, 0.88]$ where the same calculation is done with single digit precision, the result would normally be $[0.2, 0.9]$ but $[0.2, 0.9] \nsupseteq [0.16, 0.88]$ so this approach would contradict the basic principles of interval arithmetic, as a part of the domain of $f([0.1, 0.8], [0.06, 0.08])$ would be lost. Instead, it is the outward rounded solution $[0.1, 0.9]$ which is used. See Figure 2.3.

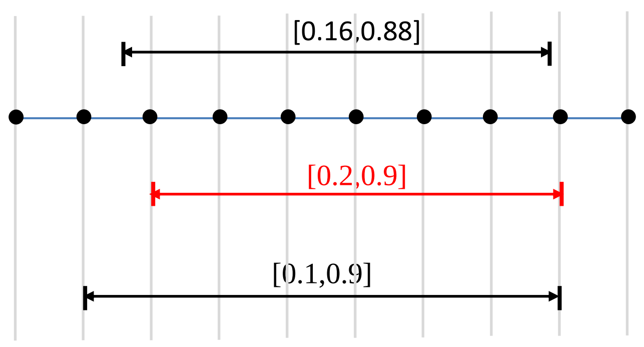

Figure 2.3: Outer bounds at different level of rounding

To summarize, by *outwardly rounded interval arithmetic*, we mean the rounding that must be used to be able to implement the interval arithmetic operations correctly. This rounding can be achieved by changing the rounding settings of the processor in the calculation of the upper limit (up) and lower limit (down). Alternatively, an appropriate small interval $[\varepsilon_1, \varepsilon_2]$ can be added, where ε_1 and ε_2 are small scalars.

2.6. The Interval Dependency Problem

The methods of classical numerical analysis cannot be transferred one-to-one into interval arithmetic, as dependencies between numerical values in the interval arithmetic are usually not taken into account.

The *dependency problem* is a major obstacle to the application of interval arithmetic. Although interval methods can determine the range of elementary arithmetic operations and functions very accurately, this is not always true with more complicated functions. If an interval number occurs several times in a calculation using parameters and each occurrence is taken independently then this can lead to an unwanted over-estimation of the resulting intervals. That is obvious in $X - X \neq [0,0]$ instead it results in the interval $\left[\underline{X} - \overline{X}, \overline{X} - \underline{X}\right]$.

Example 2.11 Let the function f is defined by

$$f(x) = x^2 + x. \tag{2.40}$$

The exact range of the function f over the interval $[-1,1]$ is $[-\frac{1}{4}, 2]$; however, using the natural interval extension produces a larger range as follows:

$$[-1,1]^2 + [-1,1] = [0,1] + [-1,1] = [-1,2]$$

This overestimation in range calculations is due to the multiple appearance of the variable x in the evaluated function f. The following function illustrates how interval arithmetic deals with multiple appearances of a variable in a given function:

$$h(x,y) = x^2 + y \tag{2.41}$$

over $x, y \in [-1,1]$. On the other hand, there is a better expression of f in which the variable x only appears once, namely by rewriting $f(x) = x^2 + x$ as addition and squaring in the quadratic $(x) = (x + \frac{1}{2})^2 - \frac{1}{4}$. So the suitable interval calculation is $([-1,1] + \frac{1}{2})^2 - \frac{1}{4} = \left[-\frac{1}{2}, \frac{3}{2}\right]^2 - \frac{1}{4} = [0, \frac{9}{4}] - \frac{1}{4} = [-\frac{1}{4}, 2]$ and gives the exact interval. Figure 2.4 illustrates the two expressions for the above example: the shaded area represents the overestimation due to the dependency problem, whereas the red curve represents the exact curve. Figure 2.5 gives a more illustrative insight of the dependency problem by plotting (2.41) rather than (2.40).

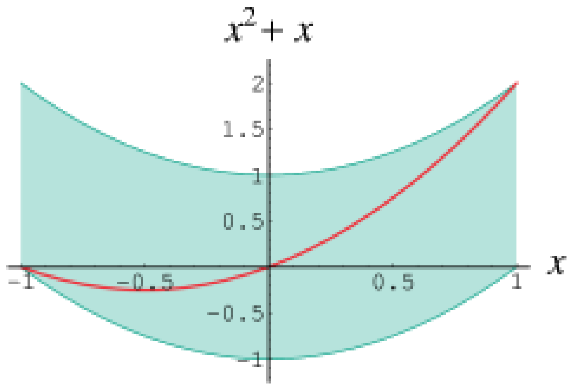

Figure 2.4: Approximate estimate of the value range

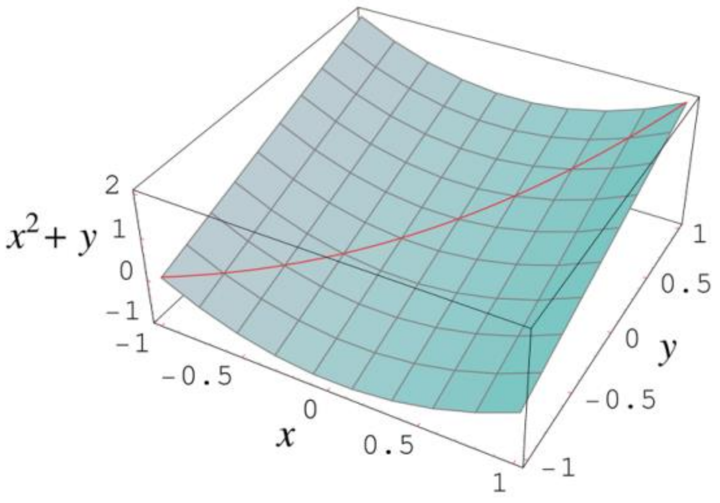

Figure 2.5: Treating each occurrence of a variable independently

In general, it can be shown that the exact range of values can be achieved, if each variable appears only once. However, not every function can be rewritten this way. The dependency and over-estimation problems can have worse effects rather than covering a large range such as preventing meaningful conclusions.

2.7. The Wrapping Effect

An additional increase in the range results from the solution of areas that do not take the form of an interval vector (a box). For example, the solution set of the linear system

$$\begin{aligned} x &= p \\ y &= p \end{aligned} \tag{2.42}$$

for $p \in [-1,1]$ is precisely the line between the points $(-1,1)$ and $(1,1)$. The best solution that the interval methods can deliver is the interval vector $([-1,1],[-1,1])$. Of course, the real solution is contained in this vector. This problem is known as *the wrapping effect*. See Figure 2.6, in which the shaded area represents the interval solution of the above linear system which includes the solid line that represents the exact solution.

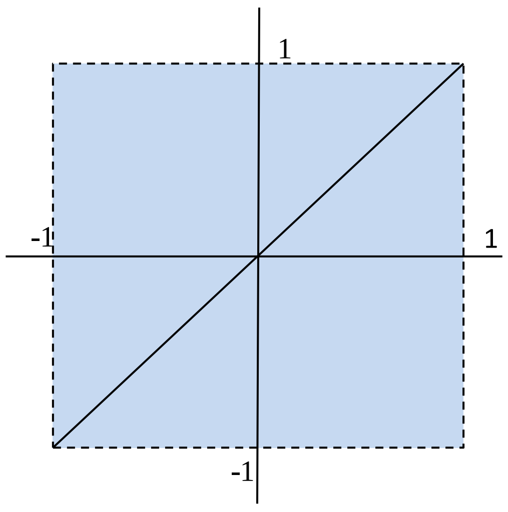

Figure 2.6: Solution set of (2.42) illustrating the wrapping effect

2.8. Interval computing software

In this section, we review a few interval computing software packages.

2.8.1. INTLIB

B. Kearfott and et al, published the interval library INTLIB [28] in 1994. First, INTLIB was written in Fortran 77 and portable to almost all commonly used computer platforms. Kearfott later converted it into Fortran 95.

Subroutines in INTLIB perform rigorous interval arithmetic with directed rounding. Subprograms in this library can be categorized into four groups according to their functionalities. They are interval arithmetic routines ($+, -, \times, \div, \ominus$); set operation routines ($\cap$;$\cup$); utility routines (direct rounding, and etc.) and routines that bound elementary mathematical functions (trigonometric, inverse trigonometric, logarithmic, exponential, hyperbolic, and etc.) with rigorous interval arithmetic.

2.8.2. Interval BLAS

Basic Linear Algebra Subprograms (BLAS) forms the fundamental tool in scientific computing. A group of international scientists from governmental agencies, computer industries, and universities formed a working committee to establish a new standard for BLAS technology from February 1996 to March, 1999.

This committee proposed the first interval BLAS standard. The functionalities included in the first release were interval vector operations, interval matrix-vector operations, interval matrix-matrix operations, set operations involving interval vectors and matrices, and utility functions involving interval vectors and matrices. Language binding and interface issues for Fortran 77, 95, and C are specified for about 200 functions and subroutines.

2.8.3. INTLAB

INTLAB is the Matlab toolbox for reliable computing and self-validating algorithms. It contains the following implementations:

- interval arithmetic for real and complex data including vectors and matrices,
- interval arithmetic for real and complex sparse matrices,
- automatic differentiation [49];
    - Gradients to solve systems of nonlinear equations,
    - Hessians for global optimization,
    - Taylor series for univariate functions,
- automatic slopes,
- rigorous real interval standard functions,
- rigorous complex interval standard functions,
- rigorous input/output,
- accurate summation, dot product and matrix-vector residuals,
- multiple precision interval arithmetic with error bounds,

and more.

INTLAB is used in many areas, from verification of chaos to population biology, from controller design to computer-assisted proofs, from PDEs to Petri Nets. For some selected references to INTLAB, see [50]. S. Rump [55] and Moore [35] include many examples based on INTLAB. Tiago Montanher wrote INTSOLVER [23], an interval based solver for Global Optimization based on INTLAB. A large collection of verification algorithms written in Matlab/INTLAB is Rohn's VERSOFT [61].

In INTLAB, everything is written in Matlab code to assure best portability. Rounding is already integral part of Matlab 5.3 (R11) and later versions under Windows. Preassumption to run INTLAB is IEEE 754 arithmetic and the possibility to permanently switch the rounding mode. This is true for a large number of PCs, workstations and main frames.

INTLAB extensively uses BLAS routines. This assures fast computing times, comparable to pure floating point arithmetic. Interval vector and matrix operations are very fast in INTLAB; however, nonlinear computations and loops may slow down the system significantly due to interpretation overhead and extensive use of the operator concept.

Consider, for example, the following code for timing of arithmetic operations (pure floating point, interval multiplication of two point matrices, point matrix times interval matrix and multiplication of two nondegenerate interval matrices):

```
n=200; A=2*rand(n)-1; intA=midrad(A,1e-12); k=100;
tic; for i=1:k, A*A;          end, toc/k
tic; for i=1:k, intval(A)*A;  end, toc/k
tic; for i=1:k, A*intA;       end, toc/k
tic; for i=1:k, intA*intA;    end, toc/k
```

The result in seconds on a 2.4 GHz Pentium IV is as follows in Table 2.2 (computing times for complex matrices are similar), where A is a point matrix and intA is the interval representation of A:

Table 2.2: Real matrix multiplication

Dimension	**Pure floating point**	**Verified** $\boldsymbol{A \times A}$	**Verified** $\boldsymbol{A \times int(A)}$	**Verified** $\boldsymbol{int(A) \times int(A)}$
100	0.0013	0.037	0.008	0.0114
200	0.012	0.025	0.051	0.074
500	0.18	0.36	0.66	0.92

In [54] and [55], information about background, implementation and timing of INTLAB is available.

2.8.4. A C++ Class Library for Extended Scientific Computing (C-XSC)

The original version of the C++ class library C–XSC (C for eXtended Scientific Computing) [30] is about twenty years old. But in the last decade the underlying programming language C++ has been developed significantly. Since November 1998 the C++ standard [24] is available and more and more compilers support (most of) the features of this standard. The new versions C–XSC [20] conform to the C++ standard.

The programming environment C–XSC is a powerful and easy to use programming tool, especially for scientific and engineering applications. C–XSC makes the computer more powerful arithmetically and significantly simplifies programming in the field of scientific computing (especially in the field of interval mathematics). C–XSC is implemented as a numerical class library in the programming language C++.

C–XSC consists of a run time system written in ANSI C and C++ including an optimal dot product and many predefined data types for elements of the most commonly used vector spaces such as real and complex numbers, vectors, and matrices. Operators for elements of these types are predefined and can be called by their usual operator symbols. Thus, arithmetic expressions and numerical algorithms are expressed in a notation that is very close to the usual mathematical notation. C–XSC allows writing verification algorithms in a way which is very near to pseudo-code used in scientific publications. All predefined numerical operators are of highest accuracy. That is, the computed result differs from the correct result by at most one rounding.

While the emphasis in computing is traditionally on speed, in C–XSC, the emphasis is more on *accuracy* and *reliability* of results. The total time for solving a problem is the sum of the programming effort, the processing time, and the time for the interpretation of results. We contend that C–XSC reduces this sum considerably.

C++ programmers should be able to use and write programs in C–XSC immediately. C–XSC simplifies programming by providing many predefined data types and arithmetic operators. Programs are much easier to read, to write, and to debug.

Chapter 3 : Interval Linear Algebraic Systems of Equations

Most of scientific computations begin with inexact initial data. Interval arithmetic is considered an efficient tool to measure uncertainties in any system by defining functions on intervals. The interval-valued functions can be very suitable tool to express many practical problems. For example, those interval functions can be used to represent the electrical circuits which contain many elements whose values cannot be exactly determined. The uncertainty in determining the electrical elements' values results from manufacturing problems. Using interval arithmetic, we can perfectly represent those elements' values and tolerances. The motivation of this chapter is to give an insight of the basic definitions and properties of interval-valued functions which will be used throughout the thesis (For more details about the interval-valued functions, the reader may consult [48] and [35]). In section 3.2, we introduce notions and definitions from literature for computing refinements of interval extensions and discuss some issues regarding the convergence for finite interval sequences. Finally, section 3.3 defines the problem of interval linear systems and introduces the interval Gauss-Seidel algorithm as a method to solve these linear systems.

3.1. Interval Functions Evaluation

3.1.1. Set Images and United Extension

In this subsection, we consider a real-valued function f of a single real variable x. Some definitions are introduced from [35] to define the precise range of values generated by $f(x)$ as x varies through a given interval X. We are interested in finding the image of the set X under the mapping f:

$$f(X) = \{f(x) : x \in X\}. \tag{3.1}$$

(3.2)In general, the concept of the image set for a function $f = f(x_1, \dots, x_n)$ is presented in [35] and is introduced here by (3.2):

$$f(X_1, \dots, X_n) = \{f(x_1, \dots, x_n) : x_1 \in X_1, \dots, x_n \in X_n\}, \tag{3.2}$$

where $X_1, \dots, X_n$ are specified intervals.

The next definition is presented in [35] to clarify definitions like the united extension and the set images.

Definition 3.1 Let $g: M_1 \to M_2$ be a mapping between sets M_1 and M_2, and denote by $S(M_1)$ and $S(M_2)$ the families of subsets of M_1 and M_2, respectively. The *united extension* of g is the set-valued mapping $\overline{g}: S(M_1) \to S(M_2)$ such that

$$\overline{g}(X) = \{g(x): x \in X, X \in S(M_1)\}, \tag{3.3}$$

3.1.2. Elementary Functions of Interval Arguments

The following example shows how it is easy for some functions to compute (3.1), consider

$$f(x) = (x-2)^2, \qquad x \in \mathbb{R}. \tag{3.4}$$

If $X = [\underline{X}, \overline{X}]$, it is obvious that the set

$$f(X) = \{(x-2)^2 : x \in X\}, \tag{3.5}$$

can be expressed as

$$f(X) = \begin{cases} \left[(\underline{X}-2)^2, (\overline{X}-2)^2\right], & 2 \le \underline{X} \le \overline{X}, \\ \left[(\overline{X}-2)^2, (\underline{X}-2)^2\right], & \underline{X} \le \overline{X} \le 2, \\ \left[0, \max\left\{(\underline{X}-2)^2, (\overline{X}-2)^2\right\}\right], & \underline{X} \le 2 \le \overline{X}, \end{cases} \tag{3.6}$$

The equation (3.5) will be used as the definition of $(X-2)^2$. But, note that this is not the same as $X^2 - 4X + 4$. For example, consider an interval $X = [1,3]$, the definition (3.6) will results in $([1,3]-2)^2 = [0,1]$, whereas $X^2 - 4X + 4 = [-1,1]^2 - 4[-1,1] + 4 = [0,9]$. However, $[0,9]$ contains $[0,1]$. The overestimation when we compute a bound on the range of $(X-2)^2$ as $X^2 - 4X + 4$ is due to the phenomenon of *interval dependency* discussed in section 2.6. Namely, for an unknown number x, where $x \in X$, if we use the expression $x^2 - 4x + 4$, the x in the second term is not only known to lie in X but also it must be the same as x in the first term, whereas, in the interval expression $X^2 - 4X + 4$, it is assumed that the values in the first and second terms vary independently.

Interval dependency is a major problem when using interval arithmetic. It is a main reason that it is not easy to replace floating point computations by intervals in an existing algorithm. This dependency may produce unsatisfying results.

3.1.3. Monotonic Interval Functions

In this subsection, we introduce some other familiar functions and apply it to interval arguments. The logic is straightforward with monotonic functions. Figure 3.1 illustrates an increasing function$f(X)$ which maps an interval $X = [\underline{X}, \overline{X}]$ into the interval

$$f(X) = [f(\underline{X}), f(\overline{X})].$$

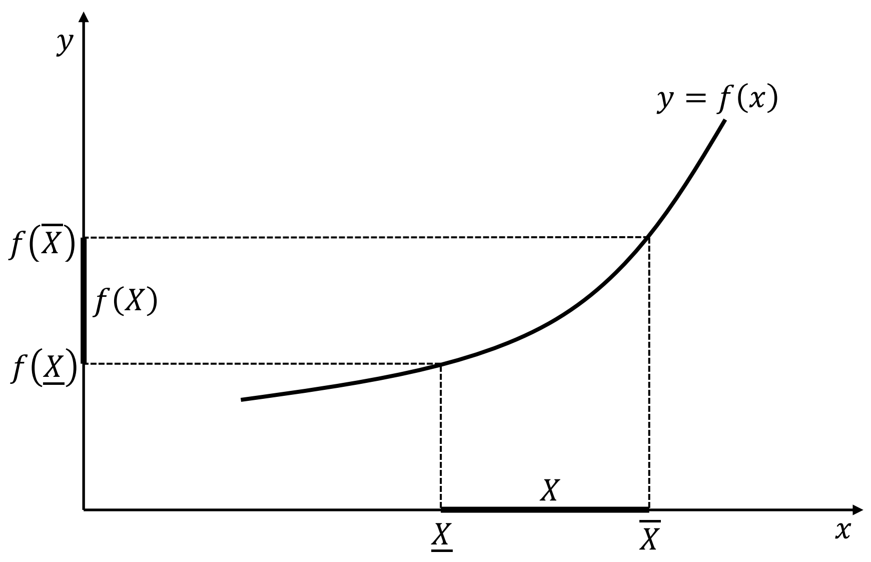

Figure 3.1: A monotonic (increasing) interval function

As an example of the monotonic functions, we may consider the exponential function

$$f(x) = \exp(x) = e^x, \qquad x \in \mathbb{R}$$

As x varies through an interval $X = [\underline{X}, \overline{X}]$, $f(x)$ takes values from $\exp(\underline{X})$ to $\exp(\overline{X})$. That is, we can define

$$\exp(X) = [\exp(\underline{X}), \exp(\overline{X})]. \tag{3.7}$$

Similarly, for the natural logarithm function

$$f(x) = \ln x, \qquad x > 0,$$

we have

$$\ln X = [\ln \underline{X}, \ln \overline{X}]. \tag{3.8}$$

The more general exponential function $f(x) = x^y$ with $x > 0$ and $y > 0$ leads us to write

$$X^y = [\underline{X^y}, \overline{X^y}] \;\; for \;\; \underline{X} > 0 \;\; and \;\; y > 0. \tag{3.9}$$

All these functions are increasing ones. With decreasing functions, the endpoints should be ordered correctly. For example, as x increases from $\underline{X}$ to $\overline{X}$, the values of $1/x$ with $x > 0$ decrease from $1/\underline{X}$ to $1/\overline{X}$. Therefore,

$$1/X = [1/\overline{X}, 1/\underline{X}]. \tag{3.10}$$

Regarding the non-monotonic functions, with some restrictions they could be monotonic. The function f given by

$$f(x) = \cos x, \qquad x \in \mathbb{R}$$

is not monotonic, but its restriction f_A to the set $A = [0, \pi]$ is decreasing. Hence,

$$\cos X = [\cos \overline{X}, \cos \underline{X}] \;\; \text{for} \;\; X \subseteq [0, \pi]. \tag{3.11}$$

3.1.4. Interval-Valued Extensions of Real Functions

In the previous sections, we discussed definitions of some interval-valued functions. Those definitions are obtained by computing the range of a real-valued function $f(x)$ as x varied through an interval X. This result was equal to the *set image* $f(X)$.

A different process is discussed in this subsection. We introduce some concepts from [35] to help us in defining more functions by *extending* a given real-valued function f by applying its *formula* directly to interval arguments.

First, we use the following definition from [35] to illustrate the meaning of the *interval extension* of a real-valued function.

Definition 3.2 We say that F is an *interval extension* of the real-valued function f, if for degenerate interval arguments, F agrees with f:

$$F([x,x]) = f(x). \tag{3.12}$$

In what follows, an example is given to clarify the previous definition. Consider the real-valued function f given by

$$f(x) = x^3, \qquad x \in \mathbb{R}. \tag{3.13}$$

The equation (3.13) represents a function which differs from the following equation

$$f(x) = x^3 \tag{3.14}$$

which is a *formula*—not a function. Any function is defined by two things: (1) the domain which it acts over, and (2) the *mapping rule* that specifies how elements of that domain are mapped. Whereas, these two things are specified in (3.13): the set of real numbers $\mathbb{R}$ represents the domain of f, and the mapping rule is $x \mapsto x^3$, the equation (3.14) does not contain a domain so it is just a formula.

Now, to develop *an extension* of the function (3.13), we apply the formula (3.14), that describes this function, to interval arguments. The resulting interval-valued function

$$F(X) = X^3, \qquad X = [\underline{X}, \overline{X}] \tag{3.15}$$

is an extension of the function (3.13).

After defining the interval extension of the real-valued function (3.13). We would like to compare $F(X)$ in (3.15) with the set image $f(X)$. We have

$$F(X) = \left[\underline{X}^3, \overline{X}^3\right]$$

On the other hand, as x increases through the interval $[\underline{X}, \overline{X}]$, the values $f(x)$ given by (3.13) clearly increase from $\underline{X}$ to $\overline{X}$; by definition then,

$$f(X) = \left[\underline{X}^3, \overline{X}^3\right]$$

As it is a simple example, we find that $F(X) = f(X)$: this interval extension of f yields the desired set image (3.1). Unfortunately, the situation is not so simple in general.

3.1.5. The Fundamental Theorem

We introduce in this subsection the fundamental theorem of interval arithmetic. To do that, we first introduce some preliminary definitions.

3.1.5.1. Subset Property of United Extension

According to [35], the following subset property is applied to the united extensions defined in (3.3):

$$X, Y \in S(M_1)\ with\ X \subseteq Y \Rightarrow \overline{g}(X) \subseteq \overline{g}(Y). \tag{3.16}$$

3.1.5.2. Interval Extensions of Multivariable Functions

In this subsection, we introduce generalized versions of the previous concepts which are suitable to multivariate functions,

$$f = f(X_1, \dots, X_n),$$

where $X_1, \dots, X_n$ are interval variables.

Definition 3.3 By an interval extension of f, we mean an interval-valued function F of n interval variables $X_1, \dots, X_n$ such that for real arguments $x_1, \dots, x_n$ we have

$$F(x_1, \dots, x_n) = f(x_1, \dots, x_n). \tag{3.17}$$

The latter definition is presented in [35]. As an example of these interval extensions, we may consider any of the interval arithmetic operations; for instance, the interval addition. This binary operation could be written as a function of two variables:

$$f(X_1, X_2) = X_1 + X_2 = \{x_1 + x_2 : x_1 \in X_1, x_2 \in X_2\}.$$

It is obvious from the above equation that interval addition is the united extension of the function

$$f(x_1, x_2) = x_1 + x_2$$

which describes ordinary numerical addition. Other interval arithmetic operations are defined in a similar manner—recall (2.22). An obvious conclusion is that the interval arithmetic functions are *united extensions* of the corresponding real arithmetic functions.

3.1.5.3. Inclusion Isotonicity

Definition 3.4 We say that $F(X_1, \dots, X_n)$ is inclusion isotonic if

$$Y_i \subseteq X_i\ for\ i = 1, \dots, n \Rightarrow F(Y_1, \dots, Y_n) \subseteq F(X_1, \dots, X_n).$$

The previous statement from [35] defines the inclusion isotonic functions. Special cases of those functions are the united extensions which all have the subset property. Then, the operations of interval arithmetic must satisfy

$$Y_1 \subseteq X_1, Y_2 \subseteq X_2 \Rightarrow Y_1 \odot Y_2 \subseteq X_1 \odot X_2, \tag{3.18}$$

because they are united extensions as mentioned before.

3.1.5.4. The Fundamental Theorem

The fundamental theorem of interval arithmetic is stated in this subsection.

Theorem 3.1 If F is an inclusion isotonic interval extension of f, then

$$f(x_1, \dots, x_n) \subseteq F(X_1, \dots, X_n)$$

Proof [38] By definition of an interval extension, $f(x_1, \dots, x_n) = F(X_1, \dots, X_n)$. If F is inclusion isotonic, then the value of f is contained in the interval $F(X_1, \dots, X_n)$ for every $(x_1, \dots, x_n)$ in $(X_1, \dots, X_n)$.■

3.2. Sequences of Intervals

This section deals with sequences of intervals and presents definitions and theories needed as preparation for the interval algorithms to be presented in Section 3.3 and Chapter 4.

3.2.1. Convergence in Interval Arithmetic

First, we introduce a definition from [35] to illustrate the notion of convergence of interval sequences in this subsection. After that some theories are presented to give a detailed overview on the convergence of interval sequences.

Definition 3.5 Let $\{X_k\}$ be a sequence of intervals. We say that $\{X_k\}$ is convergent if there exists an interval X^* such that for every $\varepsilon > 0$, there is a natural number $N = N(\varepsilon)$ such that $d(X_k, X^*) < \varepsilon$ whenever $k > N$, where d is called a metric on $\mathbb{IR}$ and given by

$$d(X, Y) = \max\{|\underline{X} - \underline{Y}|, |\overline{X} - \overline{Y}|\}.$$

As an analogue to real sequences, we can write

$$X^* = \lim_{k \to \infty} X_k \qquad or \qquad X_k \to X^*$$

and refer to X^* as the limit of $\{X_k\}$.

3.2.2. Lipschitz Interval Extensions

Lipschitz condition and functions are defined in many references such as [42]. Here, we use the definition presented in [35]. We begin with the definition of Lipschitz functions which is closely related to continuity.

Definition 3.6 An interval extension F is said to be *Lipschitz in* X_0 if there is a constant L such that

$$w\big(F(X)\big) \leq LW(X) \qquad (3.19)$$

for every $X \subseteq X_0$.

The latter definition is not applicable only to the univariate functions but also for multivariate ones; i.e. X may be an interval or an interval vector $X = (X_1, \dots, X_n)$.

Using this definition, it is easy to deduce that the width of $F(X)$ approaches zero at least linearly with the width of the interval X.

Lemma 3.1 If F is a natural interval extension of a real rational function and $F(X)$ is defined for $X \subseteq X_0$, where X and X_0 are intervals or n-dimensional interval vectors, then F is Lipschitz in X_0.

Proof [35] For any real numbers a and b and any intervals X_i and Y_i, we have the following relations:

$$\begin{aligned} &w(aX_i + bY_i) = |a|w(X_i) + |b|w(Y_i), \\ &w(X_iY_i) = |X_i|w(Y_i) + |Y_i|w(X_i), \\ &w(1/Y_i) \le \left|\frac{1}{Y_i}\right|^2 w(Y_i) \; if \; 0 \notin Y_i. \end{aligned} \tag{3.20}$$

Since the natural interval extension has interval values obtained by a fixed finite sequence of interval arithmetic operations on real constants and the components of X (if X is an interval vector) and since $X \subseteq X_0$ implies that $|X_i| \le \|X_0\|$ for every component of X, it follows that a finite number of applications of (3.19) will produce a constant L such that $w\big(F(X)\big) \le Lw(X)$ for all $X \subseteq X_0$.■

Not only the interval extensions of rational functions which are Lipschitz, but also there are certain interval extensions of irrational functions which are Lipschitz. That is stated and proved in the following lemma.

Lemma 3.2 If a real-valued function $F(x)$ satisfies an ordinary Lipschitz condition in X_0,

$$|f(x) - f(y)| \le L|x - y| \quad for \; x, y \in X_0 \tag{3.21}$$

then the united extension of f is a Lipschitz interval extension in X_0.

Proof [35] The function f is necessarily continuous. The interval (or interval vector) X_0 is compact. Thus, $w(f(X)) = |f(x_1) - f(x_2)|$ for some $x_1, x_2 \in X \subseteq X_0$. But $|x_1 - x_2| \le w(X)$; therefore, $w(f(X)) \le Lw(X)$ for $X \subseteq X_0$. ■

In what follows some examples of united extensions which are also Lipschitz in X_0:

1) $e^X = \left[e^{\underline{X}}, e^{\overline{X}}\right]$;
2) $\sqrt{X} = \left[\sqrt{\underline{X}}, \sqrt{\overline{X}}\right] \; for \; 0 < \underline{X}_0$;
3) $\ln X = \left[\ln \underline{X}, \ln \overline{X}\right] for \; 0 < \underline{X}_0$;
4) $\cos X = \left[\cos \overline{X}, \cos \underline{X}\right] for \; X_0 \subseteq [0, \pi]$.

Lemma 3.3 Let F and G be inclusion isotonic interval extensions with F Lipschitz in Y_0, G Lipschitz in X_0, and $G(X_0) \subseteq Y_0$. Then the composition $H(X) = F(G(X))$ is Lipschitz in X_0 and is inclusion isotonic.

Proof [35] The inequality

$$w(H(X)) = w(F(G(X))) \le L_2 w(G(X)) \le L_2 L_1 w(X)$$

shows that H is Lipschitz in X_0. ■

3.2.3. Convergence of Interval Sequences

Now, we proceed to define the basic concepts of convergence for a finite interval sequence.

Definition 3.7 [35] An interval sequence $\{X_k\}$ is *nested* if $X_{k+1} \subseteq X_k$ for all k.

The following lemmas show that nested sequences always converge.

Lemma 3.4 Every nested sequence $\{X_k\}$ converges and has the limit$\cap_{k=1}^{\infty} X_k$.

Proof [35] $\{\underline{X}_k\}$ is a non-decreasing sequence of real numbers, bounded above by $\overline{X}_1$, and so has a limit $\underline{X}$. Similarly, $\{\overline{X}_k\}$ is non-increasing and bounded below by $\underline{X}_1$ and so has a limit $\overline{X}$. Furthermore, since $\underline{X}_k \le \overline{X}_k$ for all k, we have $\underline{X} \le \overline{X}$. Thus $\{X_k\}$ converges to $X = [\underline{X}, \overline{X}] = \cap_{k=1}^{\infty} X_k$. ■

Lemma 3.5 Suppose $\{X_k\}$ is such that there is a real number $x \in X_k$ for all k. Define $\{Y_k\}$ by $Y_1 = X_1$ and $Y_{k+1} = X_{k+1} \cap Y_k$ for $= 1,2, \ldots$. Then Y_k is nested with limit Y, and

$$x \in Y \subseteq Y_k \ \ for\ all\ k \tag{3.22}$$

Proof [35] By induction, the intersection defining Y_{k+1} is nonempty so $\{Y_k\}$ is well defined. It is nested by construction. Relation (3.22) follows from Lemma 3.4. ■

Definition 3.8 [35] By the *finite convergence* of a sequence $\{X_k\}$, we mean there is a positive integer K such that $X_k = X_K$ for $k \le K$. Such a sequence *converges in K steps.*

The following example illustrates the above definitions and lemmas.

Example 3.1 Let $X_0 = [2,3]$, $X_{k+1} = 2 + \frac{X_k}{4}$ $(k = 0,1,2, \ldots)$, generates a nested sequence $\{X_k\}$. The rational interval function $F(X) = 2 + \frac{X}{4}$ is inclusion isotonic. Therefore, $X_1 = F(X_0) = 2 + \frac{[2,3]}{4} = [2.5,2.75] \subseteq X_0 = [2,3]$. It follows that $X_{k+1} = F(X_k) \subseteq X_k$ for all k by induction. By Lemma 3.4, the sequence has a limit X. If we compute $\{X_k\}$ using interval arithmetic, we will obtain a sequence $\{X_k^*\}$ with $X_k \subseteq X_k^*$ for all k. More precisely, let X_k^* be defined by $X_0^* = X_0 = [2,3]$, and

$$X_{k+1}^* = \left\{2 + \frac{X_k^*}{4}:\ \text{computed in IA}\right\} \cap X_k^*$$

For $k = 0,1,2$ .... It follows from Lemma 3.5 that X_k^* is nested and that the limit of $\{X_k\}$ is contained in the limit $X_k \subseteq X_k^*$. The sequence $\{X_k^*\}$ will converge in a finite number of steps. For example, with three-digit IA we find

$$\begin{aligned} X_0^* &= [2,3], \\ X_1^* &= [2.5,2.75], \\ X_2^* &= [2.62,2.69], \\ X_3^* &= [2.65,2.68], \\ X_4^* &= [2.66,2.67], \\ X_5^* &= [2.66,2.67], \end{aligned}$$

and $X_k^* = X^*$ for all $k \geq 4$. Whereas, we have finite convergence in four steps using IA, the real sequence $x_{k+1} = 2 + x_k/4$ converges to $\frac{8}{3}$ (after infinite number of steps) from any x_0.

3.2.4. Stopping Criterion

In this section, we are interested in studying the stopping criteria for nested sequences which we will use later in this thesis. In next chapter, we present numerical interval algorithms for solving systems of nonlinear equations. So, we have to define the criterion that we will use to check whether the algorithm reaches a suitable solution for the problem or not. The stopping criterion mainly depends on the accuracy required by the problem and the precision representation of machine numbers.

For any fixed precision representation of machine numbers using $(s+1)$ bits $b_0 b_1 \dots b_s$ with s fixed, there is a finite set of representable numbers. Hence, there is a finite set of intervals with machine number endpoints.

The following equation represents a stopping criterion for iterative interval algorithms that produce nested sequences of intervals with machine number endpoints. Since the sequence $\{X_k\}$ is a nested sequence, it converges in a finite number of steps according to Lemma 3.4 and Lemma 3.5. Hence, we can compute the X_k until

$$X_{k+1} = X_k. \tag{3.23}$$

If the intervals X_k are generated by a procedure of the form

$$X_{k+1} = F(X_k) \tag{3.24}$$

such that each X_{k+1} depends only on the previous X_k, then it can be shown that (3.23) is sufficient to guarantee convergence (For more details, the reader should consult section 6.3 in [35]).

In particular, if $F(X)$ is a rational expression in X and if X_0 is an interval such that $F(X_0) \subseteq X_0$, it follows that $\{X_k\}$ defined by

$$X_{k+1} = F(X_k), k = 0,1,2, \dots \tag{3.25}$$

is nested with

$$X_0 \supseteq X_1 \supseteq X_2 \supseteq \cdots$$

and converges to some X^* with $X^* = F(X^*)$ and $X^* \subseteq X_k$ for all $k = 0,1,2,\ldots$.

With interval arithmetic, it may happen that $X_1 = F(X_0) \subseteq X_0$ but $X_{k+1} \nsubseteq X_k$ for some k. Then we should compute

$$X_{k+1} = F(X_k) \cap X_k \tag{3.26}$$

instead of (3.25) and stop when (3.23) is satisfied.

The last possibility is that the interval $F(X_0) \cap X_0$ is empty, then there is no $X \in X_0$ such that $F(X) = X$. This follows from the inclusion isotonicity of F in [35], if $F(X) = X$ and $X \subseteq X_0$, then $X = F(X) \subseteq F(X_0)$ and so $X \subseteq F(X_0) \cap X_0$; therefore, if $F(X_0) \cap X_0$ is empty, there is no such X.

Whereas the following list summarizes the different possibilities of a sequence's convergence, the next two examples from [35] illustrate the above notions:

- If $F(X_0) \cap X_0 = \emptyset$, that sequence does not converge,
- Else if $F(X_0) \cap X_0 \neq \emptyset$ and $F(X_0) \subseteq X_0$, then this sequence converges to an interval X^*,
- Else if $F(X_0) \cap X_0 \neq \emptyset$ but $F(X_0) \nsubseteq X_0$, then we cannot conclude anything.

Example 3.2 Let

$$F(X) = \frac{1}{2}X + 2.$$

If we take $X_0 = [1,2]$, then

$$F(X_0) = \frac{1}{2}[1,2] + 2 = \left[\frac{5}{2}, 3\right].$$

Since $F(X_0) \cap X_0 = \emptyset$, there is no fixed point of F in[1,2]. If we take $X_0 = \left[2, \frac{7}{2}\right]$ instead, then

$$F(X_0) = \frac{1}{2}\left[2, \frac{7}{2}\right] + 2 = \left[3, \frac{15}{4}\right].$$

Here $F(X_0) \cap X_0 = \left[3, \frac{7}{2}\right]$; we cannot conclude anything since $F(X_0) \nsubseteq X_0$. Finally, with $X_0 = [2,5]$, we have $F(X_0) = F([2,5]) = \left[3, \frac{9}{2}\right]$. This time, $F(X_0) \subseteq X_0$, so F has a fixed point X in X_0. The iterations

$$X_{k+1} = F(X_k) \cap X_k, \qquad k = 0,1,2,\ldots \tag{3.27}$$

produce, in three-digit IA,

$$X_1 = [3,4.5] \cap [2,5] = [3,4.5],$$
$$X_2 = [3.5,4.25] \cap [3,4.5] = [3.5,4.25],$$
$$X_3 = [3.75,4.13] \cap [3.5,4.25] = [3.75,4.13],$$
$$X_4 = [3.87,4.07],$$
$$X_5 = [3.93,4.04],$$
$$X_6 = [3.96,4.02],$$
$$X_7 = [3.98,4.01],$$
$$X_8 = [3.99,4.01],$$
$$X_9 = [3.99,4.01],$$
$$X_{k+1} = X_k, \qquad k = 8,9,10,\dots.$$

F has a fixed point in $[3.99,4.01]$.

If the process generating the sequence depends explicitly on k as well as on X_k, say,

$$X_{k+1} = F(k, X_k),$$

then we might have $X_{k+1} = X_k$ for some k and yet $X_{k+2} \neq X_k$ even though $\{X_k\}$ is nested. The following example illustrates this point.

Example 3.3 Let

$$X_{k+1} = ([0,2]/k) \cap X_k, \qquad X_1 = [0,1]. \tag{3.28}$$

Here

$$X_1 = [0,1],$$
$$X_2 = [0,1],$$
$$X_3 = [0,1],$$
$$X_4 = \left[0, \frac{2}{3}\right],$$
$$X_5 = \left[0, \frac{1}{2}\right],$$
$$\vdots$$
$$X_{k+1} = [0, 2/k], \qquad k > 2.$$

Hence, (3.23) is a valid stopping criterion if and only if $\{X_k\}$ is nested and generated by (3.27) with $F(X_k)$ depending only on X_k.

3.3. Interval Linear Algebraic System of Equations (ILASE)

In this section, we are interested in finding solution of interval linear systems of equations. The material presented here helps us in the next section where we study solving nonlinear systems using interval arithmetic algorithms.

An ILASE is defined as follows

$$AX = B \tag{3.29}$$

where $A \in \mathbb{IR}^{n,n}$ is a regular matrix and $B \in \mathbb{IR}^{n}$ is an interval vector. The vector of unknowns is denoted by X. The corresponding solution set, known as the united solution set, S is defined in [19] by

$$S = \{x \in \mathbb{R}^n : (\exists a \in A), (\exists b \in B), (ax = b)\}. \tag{3.30}$$

This is the most popular form of the solution of (3.29). Other forms for the solution set of (3.29) are given by Shary [57].

The solution set S is generally not an interval vector and is non-convex. Thus, it is common to seek the interval vector X which is the minimum possible interval vector containing S. It is called the optimal interval solution vector, and it is written in the form

$$\begin{aligned} & X = [\underline{X}, \overline{X}] \\ , where \quad & \underline{X} = \min_i (x_i : x_i \in S) \\ & \overline{X} = \max_i (x_i : x_i \in S) \\ & for\ i \in \{1,2,\dots,n\} \end{aligned} \tag{3.31}$$

Definition 3.9 A is regular if and only if the corresponding solution set S is bounded, provided that A contains at least one regular matrix, see Rohn[52]. The regularity of an interval matrix is insured if

$$\rho(A_c^{-1}\delta A) < 1 \tag{3.32}$$

where $\rho(.)$ is the spectral radius. The more restricted condition for regularity is

$$\||A_c^{-1}|\Delta A\| < 1 \tag{3.33}$$

with $|.|$ denoting the corresponding matrix with the absolute values of its entries taken component-wise.

Initially, methods based on the interval version of the Gaussian elimination techniques, i.e. using interval arithmetic, were denoted by IGA and were extensively studied by many authors. Those methods were found to yield excellent results for narrow interval systems only under some preconditioning of the interval matrix. Preconditioning was introduced by Hansen[18] to retard the growth of interval widths. The most commonly used method for preconditioning is to multiply by an approximate inverse of the central matrix A_c. However, when the intervals are wide those methods were found to yield loose and misleading bounds (because of the accumulation of errors and the *dependency* problem which is the main drawback in interval arithmetic) and the solution obtained is quite far from being sharp, see Hansen[17] and Ning and Kearfott[45] for an illustration.

Several other methods were developed since the pioneering work of Oettli and Prager to obtain as sharp bounds as possible for the solution set. Oettli-Prager Theorem [46] states that:

Theorem 3.2 (Oettli-Prager Inequality) Let $A \in \mathbb{IR}^{m\times n}, b \in \mathbb{IR}^{m}$. Then $\tilde{x} \in \Sigma(A,b) \Leftrightarrow \left|\check{A}\tilde{x} - \check{b}\right| \le rad(A)|\tilde{x}| + rad(b)$.

Proof[46] Put $a := A\tilde{x}$. Then $A\tilde{x} \cap b \neq \emptyset \Leftrightarrow a \cap b \neq \emptyset \Leftrightarrow \left|\check{a} - \check{b}\right| \le rad(a) + rad(b)$. Since $\check{a} = \check{A}\tilde{x}$ and $rad(a) = rad(A)|\tilde{x}|$, $\left|\check{A}\tilde{x} - \check{b}\right| \le rad(A)|\tilde{x}| + rad(b)$. ■

Many methods have been introduced in the literature for the solution of an ILASE, refer to the text books Deif [7], Hansen [17] and Neumaier [42], and the references therein. Other solution methods can be also found in papers of Gay [9], Hansen [16], Jansson [25], Rohn [52] and Shary [57].

3.3.1. Interval Gauss–Seidel

Here, we introduce one of the most commonly used methods in solving interval linear systems; namely, Interval Gauss-Seidel. It is the interval version of the classical Gauss-Seidel method. It was originally introduced by Hansen et al. in the Hansen–Sengupta method [13]. In the interval version of GS, Kearfott (see [29, Chapter 3]) uses a preconditioner matrix Y which is typically, but not necessarily, chosen to be $\left(m(A)\right)^{-1}$, obtaining

$$GX = C, where\ G = YA\ and\ C = YB \tag{3.34}$$

Like the classical GS, the ith row of the preconditioned system (3.34) is solved to obtain a new range for X_i by substituting the ranges for $X_j, j \neq i$, using the most recent value of X_j wherever possible. The following equations represent the iteration equations for the interval Gauss–Seidel method:

$$\begin{aligned} &X_i^{(0)}\ is\ given, && 1 \le i \le n, \\ &\tilde{X}_i^{k+1} \leftarrow \frac{1}{G_{i,i}}\left\{C_i - \sum_{j=1}^{i-1} G_{i,j}X_j^{k+1} - \sum_{j=i+1}^{n} G_{i,j}X_j^{k}\right\}, && i = 1,2,\dots,n, \\ &X_i^{k+1} \leftarrow \tilde{X}_i^{k+1} \cap X_i^{k}, && \end{aligned} \tag{3.35}$$

where $G = \{G_{i,j}\}$. A big advantage of the interval version of GS is that it can find more than one solution in one initial box, if exist. That may happen if a step of (3.35) results in two boxes rather than one box. In this case, we have more than one solution in the initial box and we should consider each box separately. The C++ and Matlab codes of the interval GS is introduced in Appendix A.1. For more development, including use of preconditioners other than the inverse midpoint matrix, see [29].

In the next chapter, we use the interval Gauss-Seidel to solve perturbed linear systems as a step in our approach to bound solutions of the perturbed nonlinear systems.

Further discussion of interval methods for solving perturbed linear systems may be found in [1], [14], [35], and in various other references. The following example uses the interval Gauss-Seidel method to find the solution set of a perturbed linear system.

Example 3.4 Let

$$\begin{pmatrix} [0.9,1.1] & [-1.1,-0.9] \\ [0.8,1.2] & [-2.3,-1.7] \end{pmatrix} x = \begin{pmatrix} [0.8,1.2] \\ [-1.1,-0.9] \end{pmatrix} \tag{3.36}$$

If we take $X_0 = ([-10,10],[-10,10])^T$, then we can use the interval Gauss-Seidel to bound the solution of this system. A step of GS is done below:

$$Y = (m(A))^{-1} = \begin{pmatrix} 1 & -1 \\ 1 & -2 \end{pmatrix}^{-1} = \begin{pmatrix} 2 & -1 \\ 1 & -1 \end{pmatrix}$$

$Gx = C, where\ G = YA\ and\ C = YB$

$$G = YA = \begin{pmatrix} [0.6,1.4] & [-0.5,0.5] \\ [-0.3,0.3] & [0.6,1.4] \end{pmatrix}$$

$$C = YB = \begin{pmatrix} [2.5,3.5] \\ [1.7,2.3] \end{pmatrix}$$

$$X_i^{(k+1)} = \frac{1}{G_{i,i}} \left\{ C_i - \sum_{j=1}^{i-1} G_{i,j} X_j^{(k+1)} - \sum_{j=i+1}^{n} G_{i,j} X_j^{(k)} \right\}, i = 1,2,\ldots,n$$

$$X^{(0)} = \begin{pmatrix} [-10,10] \\ [-10,10] \end{pmatrix}$$

$$X^{(1)} = \begin{pmatrix} [-4.667,14.167] \\ [-4.252,10.919] \end{pmatrix} \cap \begin{pmatrix} [-10,10] \\ [-10,10] \end{pmatrix} = \begin{pmatrix} [-4.667,10] \\ [-4.252,10] \end{pmatrix}$$

We may use Oettli-Prager inequality to find a narrower solution set to (3.36). Figure 3.2 shows the solution set obtained from the Oettli-Prager inequality.

$$\check{A} = \begin{pmatrix} 1 & -1 \\ 1 & -2 \end{pmatrix}, \qquad \check{b} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\mathrm{rad(A)} = \begin{pmatrix} 0.1 & 0.1 \\ 0.2 & 0.3 \end{pmatrix}, \qquad \mathrm{rad(b)} = \begin{pmatrix} 0.2 \\ 0.1 \end{pmatrix}$$

$$|\check{A}\tilde{x} - \check{b}| \le \mathrm{rad(A)}|\tilde{x}| + \mathrm{rad(b)}$$

$$|x_1 - x_2 - 1| \le 0.1|x_1| + 0.1|x_2| + 0.2$$

$$|x_1 - 2x_2 + 1| \le 0.2|x_1| + 0.3|x_2| + 0.1$$

<u>2nd quad</u>	<u>1st quad</u>
$1.1x_1 - 1.1x_2 \le 1.2$	$0.9x_1 - 1.1x_2 \le 1.2$
$0.9x_1 - 0.9x_2 \ge 0.8$	$1.1x_1 - 0.9x_2 \ge 0.8$
$1.2x_1 - 2.3x_2 \le -0.9$	$0.8x_1 - 2.3x_2 \le -0.9$
$0.8x_1 - 1.7x_2 \ge -1.1$	$1.2x_1 - 1.7x_2 \ge -1.1$
<u>3rd quad</u>	<u>4th quad</u>
$1.1x_1 - 0.9x_2 \le 1.2$	$0.9x_1 - 0.9x_2 \le 1.2$
$0.9x_1 - 1.1x_2 \ge 0.8$	$1.1x_1 - 1.1x_2 \ge 0.8$
$1.2x_1 - 1.7x_2 \le -0.9$	$0.8x_1 - 1.7x_2 \le -0.9$
$0.8x_1 - 2.3x_2 \ge -1.1$	$1.2x_1 - 2.3x_2 \ge -1.1$

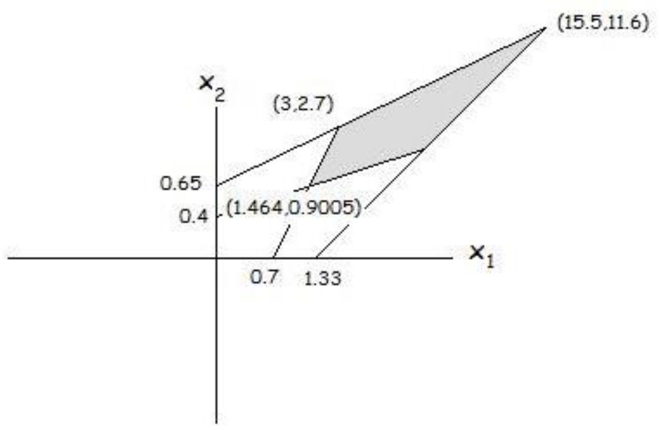

Figure 3.2: Solution set to (3.36)

3.4. Interval nonlinear systems of equations

In the next chapter, we consider finite systems of nonlinear equations

$$\begin{gathered} f_1(x_1, \dots, x_n) = 0, \\ \vdots \\ f_n(x_1, \dots, x_n) = 0 \end{gathered} \tag{3.37}$$

which we may write in vector notation as

$$f(x) = 0 \tag{3.38}$$

We can consider two cases: (1) the functions f_i are exactly representable real-valued functions, or (2) the functions f_i have coefficients known only to lie in certain intervals. We will discuss case (1) first and extend the results to case (2).

In the next chapter, we will consider three methods that can be used to solve interval nonlinear systems of equations. Each method has its own advantages that will be covered.

Chapter 4 : Perturbed Nonlinear Systems of Equations

Interval methods that bound the solutions of real-valued nonlinear systems have been studied in the literature for decades. Our focus here is on solving nonlinear perturbed systems. We present some interval methods such as Interval Newton, Krawczyk, and Hansen-Sengupta to solve those perturbed problems. In section 4.5, we introduce a modified version of the Interval Newton Method to bound the solutions of nonlinear systems with inexact data.

Interval Newton methods of various types have been published for finding and bounding solutions of systems of real-valued nonlinear equations. Hansen and Greenberg [12] combine various features of these methods into a single algorithm of greater efficiency.

In his 2006 paper, Hansen [15] considers various issues in multidimensional interval Newton methods and gives an algorithm based on consideration of them. These issues include choice of points of expansion, computing and reusing Jacobians, and choice of the preconditioner.

There are other interval algorithms which were introduced to solve nonlinear systems such as Krawczyk method. Krawczyk introduced a modification on the interval Newton method which avoided solving interval linear systems using Gauss-Seidel by not attempting to obtain a sharp solution of the system (see [32]). Another algorithm was introduced by Hansen and Sengupta in 1981 (see [13]) and a parametric form of this algorithm to solve the perturbed problems was introduced later in 2008 (see [11]).

There are other papers which consider the area of optimization with perturbed data. For example, Dinkel, Tretter, and Wong [8] show how interval analysis methods can be used to study the optimal solution of perturbed constrained optimization problems.

In this chapter, we also consider some problems where there might be uncertainty about the values of its parameters. For example, those parameters might be measured quantities of uncertain accuracy. The function f, which we seek to find its zeros, might involve numbers that cannot be exactly expressed in the computer's number system. For example, the function might be expressed in terms of transcendental numbers such as π.

Any such parameters or numbers can be expressed as intervals that contain their true values and whose endpoints are machine-representable numbers. The value of a function $f(x)$ involving such intervals is itself an interval for any x. In this research, we discuss a slight modification made on interval Newton method to solve those perturbed nonlinear systems.

In this thesis, we study the use of interval Newton operator for finding and bounding the zeros of functions whose coefficients are not exactly known. This may occur in applications such as the sensitivity analysis of most of the engineering problems such as the sensitivity analysis that is performed on electrical circuits.

The chapter is organized as follows: In section 4.1, the univariate interval Newton method is introduced. Interval methods for solving multivariate nonlinear systems of equations are given in section 4.2. The perturbed problem formulation and the derivation of the existence and convergence theories are given in section 4.3. In section 4.4, we consider the interval algorithms for bounding the solutions of perturbed nonlinear systems of equations. In section 4.5, the modified interval (two-stage) Newton method is introduced. Numerical examples that illustrate the method are given in section 4.6.

4.1. Univariate Interval Newton's Method

In this subsection, the interval Newton's method for solving a nonlinear equation is presented. Our approach is similar to what has been introduced in presenting interval Newton method in [35].

Let f be a real-valued function of a real variable x, and suppose that f is *continuously differentiable*. We search for a solution of the equation

$$f\ (x)\ =\ 0 \tag{4.1}$$

in a given interval $[a, b]$. Using the mean value theorem, we have

$$f(x) = f(y) + f'(s)(x - y) \tag{4.2}$$

for some s between x and y. If there exists a solution x to (4.1) in the interval $[a, b]$, then it would satisfy

$$f(y) + f'(s)(x - y) = 0. \tag{4.3}$$

for any real number $y\ \in\ [a, b]$. The previous equation becomes

$$x = m([a, b]) - \frac{f(m([a, b]))}{f'(s)}, \tag{4.4}$$

if we take $y\ =\ m([a, b])$. As $s \in [a, b]$, we may write the interval form of (4.4) as follows

$$N(X) =\ m(X) - \frac{f\big(m(X)\big)}{F'(X)}, \tag{4.5}$$

where $X = [a, b]$, $F'(X)$ is an *inclusion monotonic interval extension* of $f'(x)$, and $N(X)$ is the interval Newton operator which is an interval that contains the solution x if $y\ =\ m(X)$. Finally, the algorithm uses the following equation

$$X^{(K+1)} =\ X^{(K)} \cap\ N\big(X^{(K)}\big) \quad (k\ =\ 0, 1, 2, \dots) \tag{4.6}$$

to update the interval X at each step. The following theorem discusses the existence and convergence of the algorithm.

Theorem 4.1 If an interval $X^{(0)}$ contains a zero x of $f(x)$, then so does $X^{(K)}$ for all $k\ =\ 0, 1, 2, \dots$, defined by (4.6). Furthermore, the intervals $X^{(K)}$ form a nested sequence converging to x if $0 \notin F'\big(X^{(0)}\big)$.

Proof [35] If $0 \notin F'(X^{(0)})$, then $0 \notin F'(X^{(K)})$ for all k and $m(X^{(K)})$ is not contained in $N(X^{(K)})$, unless $f\left(m(X^{(K)})\right) = 0$. Therefore $w(X^{(K+1)}) < \frac{1}{2}w(X^{(K)})$. Convergence of the sequence follows. ■

Example 4.1 Suppose we wish to solve (4.1) with

$$f(x) = \cos x. \tag{4.7}$$

$F'(X) = -\sin X$ is an interval extension of $f'(x) = -\sin x$. Hence,

$$N(X) = m(X) + \frac{\cos m(X)}{\sin X},$$

and (4.6) looks like

$$X^{(K+1)} = X^{(K)} \cap \left\{ m(X^{(K)}) + \frac{\cos m(X^{(K)})}{\sin X^{(K)}} \right\}.$$

Taking $X^{(0)} = [1/2, 2]$, we obtain

$$X^{(1)} = [1.56532236239526, 1.90770873056387],$$
$$X^{(2)} = [1.56172717707345, 1.57155380653374],$$
$$X^{(3)} = [1.57079631483236, 1.57079648574536].$$
$$X^{(4)} = [1.57079632679489, 1.57079632679490],$$
$$X^{(5)} = [1.57079632679489, 1.57079632679490],$$

Of course, (4.7) has solution $x = \pi/2$. Solving it using IA, we see that $\pi/2$ lies in the interval $[1.57079632679489, 1.57079632679490]$.

The previous example illustrates the fast convergence of the interval Newton method which is *asymptotically error squaring*. The following lemma from [35] states the rate of convergence of the method.

Lemma 4.1 Given a real rational function f of a single real variable x with rational extensions F, F' of f, f', respectively, such that f has a simple zero y in an interval $[x_1, x_2]$ for which $F([x_1, x_2])$ is defined and $F'([x_1, x_2])$ is defined and does not contain zero, there is an interval $x_0 \subseteq [x_1, x_2]$ containing y and a positive real number C such that

$$w(X^{(K+1)}) \leq C\left(w(X^{(K)})\right)^2. \tag{4.8}$$

For a proof, see [38] and for an illustrating example that compares between the convergence of the interval and traditional Newton methods, see [27].

Like the traditional Newton method, the univariate interval Newton method has a geometric interpretation. Whereas the traditional method is interpreted as an intersection of a single tangent line with the *x*-axis, the interval method defines the new interval $X^{(K+1)}$ by the intersection of two tangent lines, with slopes corresponding to the lower and upper bounds of $F'(X^{(K)})$.

This is illustrated with $F'(X^{(K)})$ equal to the range of f' over $X^{(K)}$ in Figure 4.1 (In this figure, the dashed slope represents the lower bound on $F'(X^{(K)})$, and the dotted slope represents the upper bound on $F'(X^{(K)})$).

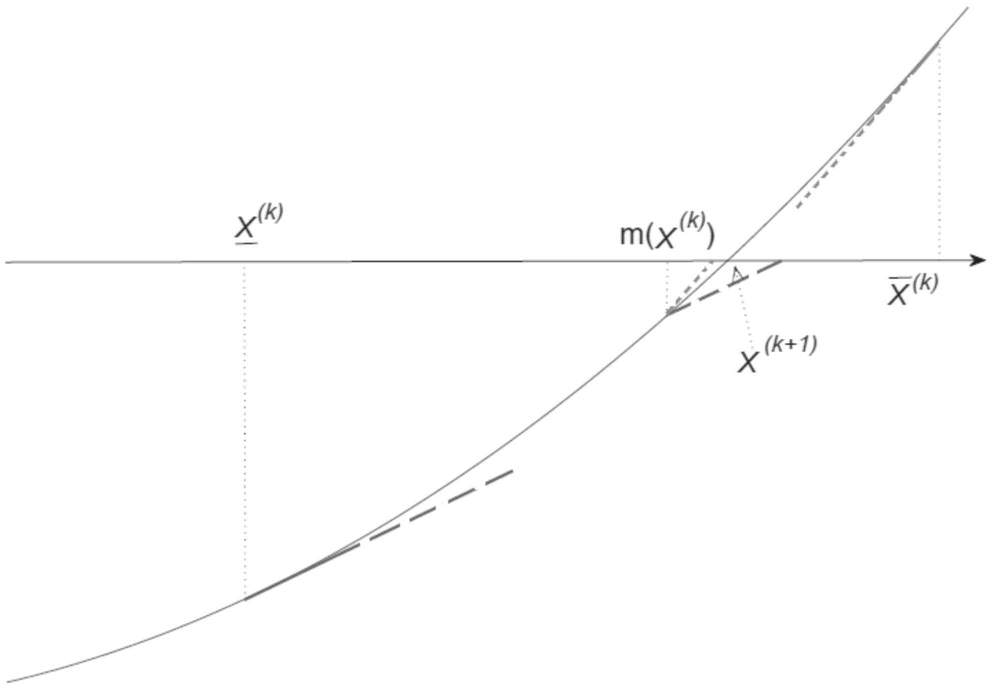

Figure 4.1: Geometrical interpretation of the univariate interval Newton method

The software implementation of the interval Newton method are available in Appendix A.2. There is no need to explicitly program the derivative, since the INTLAB and C-XSC provide *automatic differentiation* capabilities which may be used directly.

4.1.1. Extended Interval Newton's Method

Back to the function (4.7), it has infinite number of solutions which are defined as $x = n\pi$, where n is an integer number. Here, we try to widen the initial interval to include more zeros and see if we can find them all. We may start with a starting interval $[-5,5]$ but that would violate the condition $0 \notin F'(X^{(0)})$ in the proof of Theorem 4.1. This situation can be handled using *extended interval arithmetic*.

Using the extended interval arithmetic discussed in subsection 2.2.4.1, we can allow the range of $F'(X)$ to contain zero. Hence, the quotient $f\,(x)/F'(X)$ existing in the computation of

$$N(X) = m(X) - f\,(x)/F'(X)$$

will split into two unbounded intervals. Then, upon intersecting $N(X)$ with the finite interval X in the iteration formula

$$X^{(K+1)} = N(X^{(K)}) \cap X^{(K)},$$

we obtain two disjoint finite intervals. For the function (4.7), by taking $X^{(0)} = [-5,5]$, we obtain

$$m\left(X^{(0)}\right) = 0, \qquad f\left(m\left(X^{(0)}\right)\right) = 1,$$

and

$$F'\left(X^{(0)}\right) = [-1,1].$$

Hence,

$$\begin{aligned} N(X^{(0)}) &= m(X^{(0)}) - f(m(X^{(0)}))/F'(X^{(0)}) \\ &= 0 - \{1/[-1,1]\} \\ &= (-\infty,-1] \cup [1,\infty). \end{aligned}$$

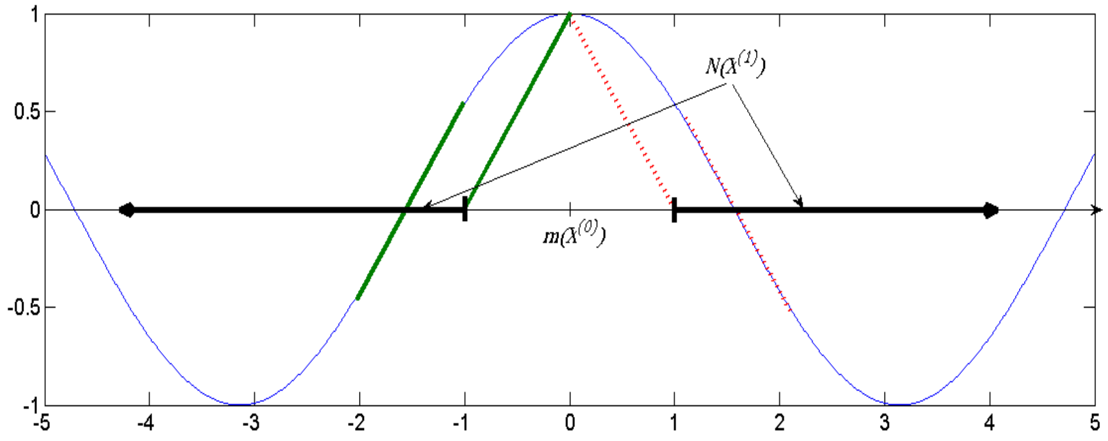

Figure 4.2: Extended interval Newton step over X$^{(0)}$=[-5,5], function (4.7)

Intersecting these two infinite intervals with $X^{(0)} = [-5,5]$, we get the union of two disjoint finite intervals,

$$X^{(1)} = [-5,-1] \cup [1,5].$$

This is illustrated in Figure 4.2.

Now, we can solve the function (4.7) for each interval individually. Considering the interval $[1,5]$ as a new initial box, we obtain the Newton operator as two disjoint infinite intervals $(-\infty, 2.704187084467255] \cup [3.642092615934331, \infty)$. Intersecting those two intervals with the interval $[1,5]$ results in two finite intervals $[1,2.704187084467255]$ and $[3.642092615934331,5]$. Again, we should solve (4.7) for each interval individually. After convergence for these two intervals, we can go back and set $X^{(0)} = [-5,-1]$ to find the other roots in $[-5,5]$. In this way, the interval Newton method can find all zeros of a function in a given starting interval. In this example, the interval Newton method finds the four zeros of (4.7) which lie in the interval $[-5,5]$.

The operation of interval division by any interval that contains zero results in the interval $[-\infty,\infty]$. Hence, we use a custom implementation of the interval division to find solutions for the previous example. The custom function that performs the division by an interval containing zero is developed in [35] using concepts of extended interval arithmetic and its implementation can be found in Appendix A.6.

The main difference between the interval Newton method and the ordinary Newton method is that the interval version deals with sets instead of points which are used in the ordinary version. Usage of sets rather than points enables the interval Newton method to find all roots of a function in a given starting interval. Whereas the ordinary Newton method does not always converge, the convergence of the interval method is always guaranteed (for more details the reader may consult [40]).

4.2. Multivariate Nonlinear System of Equations

In this section, the most commonly used interval algorithms that find and bound the solution of nonlinear systems are considered. Each approach has its own advantages, and all of them are used in practice.

4.2.1. Multivariate Interval Newton Method

We start with the interval Newton method for solving nonlinear systems. The interval Newton method for solving nonlinear systems can be developed more closely analogously to the univariate case discussed in section 4.1. The following equations

$$f\,(y) - f\,(x) = A(y - x) \tag{4.9}$$

$$A(y - x) = f\,(y), \tag{4.10}$$

$$x = y - A^{-1} f(y), \tag{4.11}$$

are analogues to (4.2), (4.3), and (4.4), respectively, where x and y lie in the domain of f and A is a *nonsingular* matrix whose ith row is defined as

$$A_{i,:} = \nabla^T f_i(c_i) = \left(\frac{\partial f_i}{\partial x_1}(c_i), \dots, \frac{\partial f_i}{\partial x_n}(c_i)\right),$$

with $c_i \in \mathbb{R}^n$ is some point between x and y. The point matrix A is analogue to $f'(s)$. Although it is not necessary for A to be the Jacobian matrix $F'(c)$, it must satisfy the equation $A \in F'(X)$, where $F'(X)$ is an element-wise interval extension of the Jacobian matrix over some box X that contains both x and y. To obtain the multivariate interval Newton operator, we may use the following equation

$$N(X) = m(X) - \left(F'(X)\right)^{-1} f\big(m(X)\big), \tag{4.12}$$

which is analogue to (4.5). But in this case, we need to find inverse of an interval matrix which is not an easy operation. Instead of doing such complex operation, the multivariate Newton operator is redefined as

$$N(X) = y + V, \tag{4.13}$$

where V bounds the solution set to

$$F'(X)v = -f(y). \tag{4.14}$$

The following theorem is a special case of Theorem 5.1.7 found in [42].

Theorem 4.2 Suppose f has continuous partial derivatives over X and $F'(X)$ is an element-wise interval extension to the Jacobian matrix of F over X. Suppose y is any point in X and suppose $N(X)$ is defined as in (4.13), where V is computed by any method for enclosing the solution set to the linear system $F'(X)v = -f(y)$. Then $N(X) \subset \operatorname{int}(X)$ implies that the function f has a unique solution in X that is also in $N(X)$.

The linear system (4.14) can be solved using any interval method that bounds the solutions of linear systems, for example, the interval Gauss–Seidel method, mentioned in Section 3.3.1, or the Krawczyk method for solving linear systems (See [35] for more details about Krawczyk method). Finally, we stated the different possibilities which may be encountered during computations:

1) $N(X^{(0)}) \subset int(X^{(0)}) \Longrightarrow$ convergence to the unique solution in $X^{(0)}$, where $int(X^{(0)})$ denotes the topological interior of the box $X^{(0)}$.
2) $N(X^{(0)}) \cap X^{(0)} = \emptyset \Longrightarrow$ no solution in $X^{(0)}$.
3) $N(X^{(0)}) \cap X^{(0)} \neq \emptyset \Longrightarrow$ no conclusion, but we can restart with $X_{NEW} = N(X^{(0)}) \cap X^{(0)}$.
4) $N(X^{(0)})$ is not defined. In this case, we can bisect $X^{(0)}$ and process each half separately.

In addition, if we use the interval Gauss–Seidel method or a similar approach, then a denominator $G_{i,i}$ could contain zero, and we have a fifth possibility:

5) $N(X^{(0)}) \cap X^{(0)}$ is the union of two boxes; we can process each of these two boxes separately.

The following example from [35] illustrates the multivariate interval Newton method.

Example 4.2 Consider the system of equations

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 + x_2^2 - 1 = 0, \\ f_2(x_1, x_2) &= x_1 - x_2^2 = 0. \end{aligned} \tag{4.15}$$

For f' we have the Jacobian matrix

$$f'(x) = \begin{pmatrix} 2x_1 & 2x_2 \\ 1 & -2x_2 \end{pmatrix} \tag{4.16}$$

For obtaining the interval extensions F and F' of f and f' respectively, we take the natural interval extensions of the corresponding real functions, simply evaluating (4.15) and (4.16) in interval arithmetic:

$$\begin{aligned} F_1(X) &= X_1^2 + X_2^2 - 1, \\ F_2(X) &= X_1 - X_2^2, \end{aligned} \quad \text{and} \quad F'(X) = \begin{pmatrix} 2X_1 & 2X_2 \\ 1 & -2X_2 \end{pmatrix}.$$

Solving (4.15) using $X = ([0.5, 0.8], [0.6, 0.9])^T$ as an initial box. Then, we compute $N(X)$ and the new box X by solving (4.13). Subsequently, we have

$$N(X) = \begin{pmatrix} [0.5687, 0.6588] \\ [0.7702, 0.8130] \end{pmatrix}$$

This method produces a nested sequence of interval vectors containing the solution, and using IA converges in a finite number of steps to an interval vector containing a solution of (4.15) and is given by.

$$X = \begin{pmatrix} [0.6175, 0.6176] \\ [0.7850, 0.7851] \end{pmatrix}$$

The solution obtained is the smallest box that can be found bounding the solution of (4.15).

4.2.2. The Krawczyk Method

In this subsection, we consider another interval method for solving nonlinear systems known as: the Krawczyk method [32].

Suppose that f in (3.37) is *continuously differentiable* in an open domain D. Suppose that we can compute inclusion isotonic interval extensions F and F' for f and f' respectively, defined on interval vectors $X \subseteq D$. We have the following computational test for the existence of a solution [32, 37].

Theorem 4.3 Let Y be a nonsingular real matrix approximating the inverse of the real Jacobian matrix $f'(m(X))$ with elements $f'(m(X))_{ij} = \partial f_i(x)/\partial x_i$ at $x = m(X)$. Let y be a real vector contained in the interval vector $X \subseteq D$. $K(X)$ is defined as follows

$$K(X) = y - Yf(y) + \{I - YF'(X)\}(X - y). \qquad (4.17)$$

If the Krawczyk operator denoted by K(X) ⊆ X, then (3.37) has a solution in X, it is also in K(X).

An early work containing the proof of this theorem is given in [37]. The proof is based on a generalization of the Brouwer fixed-point theorem, or a specific instance of the Schauder fixed-point theorem [56]. Here, we present the proof found in [35].

Proof[35] Define $g(y) = y - Yf(y)$. Then, since Y is a nonsingular matrix, $g(y) = y$ if and only if $f(y) = 0$. Thus, if there exists an $x \in X$ such that $g(x) = x$, that is, if there is a fixed point of g in x, then there is a solution to $f(x) = 0$ in x. However, if the Jacobian matrix of g at y is denoted by $g'(y)$, then we have

$$g'(y) = I - Yf'(y).$$

Therefore, the mean value extension of g over X about the point $y \in X$ is simply

$$y - Yf(y) + \{I - YF'(X)\}(X - y) = K(X).$$

Thus, $K(X)$ must contain the range of g over X, that is, $g(X) \subseteq K(X)$. Thus, if $K(X) \subseteq X$, then $g(X) \subseteq X$, the hypotheses of the Schauder fixed-point theorem hold, so g has a fixed point in X, so $f(x) = 0$ has a solution in X.■

For instance, if we choose $y = m(X)$, then $K(X)$ lies in the interior of X if

$$\|K(X) - m(X)\| < \frac{w(X)}{2}. \tag{4.18}$$

Thus, for an n-cube X, (4.18) is sufficient for the existence of a solution to (3.38) in X. The following theorem from [35] states that the same condition (4.18) is also sufficient to guarantee convergence of the interval Krawczyk method.

Theorem 4.4 Let X be an n-cube, $y = m(X)$, and Y a nonsingular real matrix. Suppose (4.18) is satisfied. Put $X^{(0)} = X$, $Y^{(0)} = Y$ and consider an arbitrary real vector $X^{(0)}$ in $Y^{(0)}$. Then the system (3.37) has a unique solution in X, and the following algorithm converges to the solution [36,37]:

$$X^{(k+1)} = X^{(k)} \cap K(X^{(k)}) \; (k = 1, 2, \ldots), \tag{4.19}$$

where

$$K(X^{(k)}) = y^{(k)} - Y^{(k)} f\,(y^{(k)}) + \{I - Y^{(k)} F'(X^{(k)})\} Z^{(k)}$$

and

$$y^{(k)} = m(X^{(k)}), Z^{(k)} = X^{(k)} - m(y^{(k)}),$$

and where $y^{(k)}$ is chosen as

$$Y^{(k)} = \begin{cases} Y, an\ approximation\ to\ [m(F'(X^{(K)}))]^{-1}, if \left\|I - YF'(X^{(K)})\right\| \le \left\|I - Y^{(K-1)} F'(X^{(K-1)})\right\| \\ Y^{(K-1)}\ otherwise. \end{cases}$$

The following example illustrates the Krawczyk method.

Example 4.3 Consider the system of equations solved in Example 4.2. Suppose we decide to try $X = ([0.5, 0.8], [0.6, 0.9])^T$. Then, we have

$$m\big(F'(X)\big) = \begin{pmatrix} m([1, 1.6]) & m(\,[1.2, 1.8]) \\ m([1, 1]) & m([-1.8, -1.2]) \end{pmatrix} = \begin{pmatrix} 1.3 & 1.5 \\ 1 & -1.5 \end{pmatrix}.$$

As an approximate inverse of this matrix, we will take

$$Y = \begin{pmatrix} 0.43 & 0.43 \\ 0.29 & -0.37 \end{pmatrix}. \tag{4.20}$$

Putting $\mathrm{y} = (0.65, 0.75)^{\mathrm{T}} = \mathrm{m(X)}$, we find from (4.1) for the 2-cube X,

$$\mathrm{K(X)} \subseteq ([0.559, 0.68], [0.74, 0.84]\,)^{\mathrm{T}}.$$

Since $\|\mathrm{K(X)} - \mathrm{m(X)}\| = 0.091 < \mathrm{w(X)}/2 = 0.15$, the hypotheses for Theorem 4.4 are satisfied. The iterative method (4.19) converges to a solution of (4.15) using Y given by (4.20) from $\mathrm{X}^{(0)} = ([0.5, 0.8], [0.6, 0.9])^{\mathrm{T}}$. It produces a nested sequence of interval vectors containing the solution, and using IA converges in a finite number of steps to an interval vector containing a solution of (4.15).

It is shown in [34] that the widths of the containing interval vectors converge quadratically to zero if $F'(X)$ is a Lipschitz extension of $f'(X)$. The Krawczyk method can be implemented in INTLAB and C-XSC, for more details and the codes see Appendix A.3.

4.2.3. The Modified Krawczyk Method

In this subsection, we consider another interval method for solving nonlinear systems known as: the Krawczyk method [32]. A simple modification of Krawczyk's algorithm for the solution of a system of nonlinear equations is presented in [63]. It is shown that under the hypotheses imposed in [34], [36], and [37], the modified algorithm converges more rapidly than Krawczyk algorithm and with greater computational efficiency.

The modified Krawczyk algorithm is as follows:

```
1) Compute $x^{(0)} = m(X^{(0)})$.
2) Compute $B^{(0)} = \{m(F'(X^{(0)}))\}^{-1}$.
3) Set $A^{(0)} = B^{(0)}$.
4) Compute $r^{(0)} = \|I - A^{(0)}F'(X^{(0)})\|$.
5) Set $k = 0$.
6) Set $i = 1, X_0^{(k)} = X^{(k)}, x_0^{(k)} = x^{(k)}$.
7) Compute $Y_i^{(k)} = x_{i-1}^{(k)} - A^{(k)}f\left(x_{i-1}^{(k)}\right) + \{I - A^{(k)}F'(X^{(k)})\}(X_{i-1}^{(k)} - x_{i-1}^{(k)})$.
8) Compute $Z_i^{(k)} = Y_i^{(k)} \cap X_{i-1}^{(k)}$.
9) If $Z_i^{(k)} = X_{i-1}^{(k)}$ and $i \neq 1$, then stop; else if $i = 1$, then enlarge $X^{(0)}$ and go
   to 1).
10) Set $X_i^{(k)} = Z_i^{(k)}$.
11) Compute $x_i^{(k)} = m(X_i^{(k)})$.
12) If $i = p$, then go to 14.
13) Set $i = i + 1$ and go to 7.
14) Set $X^{(k+1)} = X_p^{(k)}, x^{(k+1)} = x_p^{(k)}$.
15) Compute $B^{(k+1)} = \{m(F'(X^{(k+1)}))\}^{-1}$.
16) Compute $s^{(k+1)} = \|I - B^{(k+1)}F'(X^{(k+1)})\|$.
17) If $s^{(k+1)} \leq r^{(k)}$, then set $A^{(k+1)} = B^{(k+1)}, r^{(k+1)} = s^{(k+1)}$, and go to 20.
18) Set $A^{(k+1)} = A^{(k)}$.
19) Compute $r^{(k+1)} = \|I - A^{(k+1)}F'(X^{(k+1)})\|$.
20) Set $k = k + 1$ and go to 6.
```

4.2.4. Hansen-Sengupta Method

In this subsection, we introduce a more efficient method than Krawczyk: Hansen-Sengupta. Hansen-Sengupta method produces, in each iteration, a smaller box than what is produced by Krawczyk's iteration. Hence, unlike Krawczyk method, Hansen-Sengupta method needs fewer steps for convergence.

The presentation given here follows the one given by Neumaier in [42] and Goldsztejn in [10]. Recall the interval Gauss-Seidel method from Section 3.3.1, the interval GS operator is defined in this section as follows: First in dimension one,

$$\Gamma(A, B, X) := \{x \in X | \exists a \in A, \exists b \in B, AX = B\}. \tag{4.21}$$

And, we have the special case where $0 \notin A$, the previous expression is reduced to $\Gamma(A, B, X) := \left\{\frac{B}{A}\right\} \cap [x]$ (for the expression in the case $0 \in A$, see [42]). Second, the multidimensional Gauss-Seidel is defined as follows: $\Gamma(A, B, X, Z) := X^{(k+1)}$ where

$$X_i^{(k+1)} := \Gamma\left(A_{ii}, B_i - \sum_{j<i} A_{ij} X_j^{(k)} - \sum_{j>i} A_{ij} X_j^{(k)}, Z_i\right). \tag{4.22}$$

The interval vector Z in the latter equation does not appear in the definition of the interval Gauss-Seidel discussed in Section 3.3.1 because it is considered equal to the interval vector X. Then, the Hansen-Sengupta operator [13] can be defined as follows

$$\tilde{x} + \Gamma(J, -Y, X - \tilde{x}, Z - \tilde{x}), \tag{4.23}$$

where $J \in \mathbb{IR}^{n\times n}$ and $X, Y, Z \in \mathbb{IR}^n$. The following theorem proves the existence and convergence of the Hansen-Sengupta method (see [42] and [11]).

Theorem 4.5 Let $X, Y, Z \in \mathbb{IR}^n$, $\tilde{x} \in X$ and $J \in \mathbb{IR}^{n\times n}$ such that: $X \subseteq Z$, $f(\tilde{x}) \in Y$ and $J \supseteq \left\{\frac{df}{dx}(x) \in \mathbb{R}^{n\times n} | x \in X\right\}$. If $X^{(k+1)}$ denotes (4.23) then:

1) $x \in X$ and $f(x) = 0$ implies $x \in X^{(k+1)}$.
2) If $\emptyset \neq X^{(k+1)} \subseteq int(X)$ then f has a unique zero in $X^{(k+1)}$.

One can use *Lipschitz* interval matrices instead of the interval derivatives to release the differentiability hypothesis, and can use slope matrices to improve the enclosure; however, the uniqueness of solution is lost (the reader may consult [42] for details).

To improve the efficiency of the Hansen-Sengupta operator, a preconditioning matrix can be used to solve the nonlinear system. Hence, we will solve the preconditioned system $C.f(x) = 0$, where C is a nonsingular real matrix. The preconditioning matrix C is always chosen so that $C.f$ is close to the identity.

4.3. Perturbed Nonlinear Systems

Solving *parameter-dependent* systems of equations is an important part of scientific computation. Traditionally, this is done either by continuation methods which trace a particular solution curve, as in [2] and [51], or by linearizing the equations around a particular solution and to deduce from this linearization the effect on the solution of small changes in one or several parameters. The latter technique has become known under the name of *sensitivity analysis*. Because of the negligence of higher-order nonlinearities, traditional sensitivity analysis is valid only for sufficiently small changes. In this chapter, we use interval analysis to find and bound the solution of *perturbed nonlinear systems*.

While some interval methods that solve the real-valued nonlinear systems, like interval Newton, is known to be always convergent, the convergence in solving perturbed nonlinear systems is not guaranteed. In the spirit of earlier work by Neumaier [43], we show here by examples that finding and bounding solutions of perturbed problems depends on the parameters width. The numerical examples discussed here show that for narrow parameter intervals the verified algorithms give excellent result, and unreasonable overestimation need not be feared. But for wider ranges of parameters, interval algorithms produce enclosures which are much wider than the true range of the solution of the problem considered.

In this section, we state the definition of the perturbed nonlinear system for which we are interested in finding a solution. Let $\mathrm{F}: \mathbb{R}^{\mathrm{n}} \times \mathbb{IR}^{\mathrm{m}} \rightarrow \mathbb{IR}^{\mathrm{n}}$ be a *continuously differentiable* function of n variables and m parameters. Then we write the problem in the form of:

$$\begin{gathered} F_1(x_1, \dots, x_n, P_1, \dots, P_m) = 0 \\ \vdots \\ F_n(x_1, \dots, x_n, P_1, \dots, P_m) = 0 \end{gathered} \tag{4.24}$$

where $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ and $P = (P_1, P_2, \dots, P_n) \in \mathbb{IR}^m$. In vector notation, we define the solution to this problem to be the set $S = \{x: f(x, p) = 0\}$ for $p \in P$. For a given value of p, we expect the function f to have a set of discrete zeros. As p varies over P, a given zero, say x^* becomes an interval, say X^*. The solution is defined as a box and given by the algorithms discussed later. In vector notation, the system (4.24) is written as:

$$F(x, P) = 0, \tag{4.25}$$

4.3.1. An Illustrative Example

In this subsection, an illustrative example is introduced to help readers in understanding the main problem which is under test in this thesis.

Consider the system of equations $F: \mathbb{R}^2 \times \mathbb{IR}^2 \rightarrow \mathbb{IR}^2$

$$\begin{gathered} F_1(x, P) = x_2 - P_1 x_1^2, \\ F_2(x, P) = x_1 - P_2 x_2^2 \end{gathered} \tag{4.26}$$

where P_1 and P_2 are two parameters which are only known to lie in the interval $[0.9, 1.1]$. Figure 4.3 illustrates the solution of $F = 0$. Because of the interval nature of the functions, each of F_1 and F_2 is represented as a group of curves, not only one curve. The intersection between the two groups of functions represents the solution of the above nonlinear system. Using interval arithmetic, this solution is represented as a box (2-D interval vector). As shown in Figure 4.3, the over-estimation problem is present, when interval algorithms are used in calculating the solution.

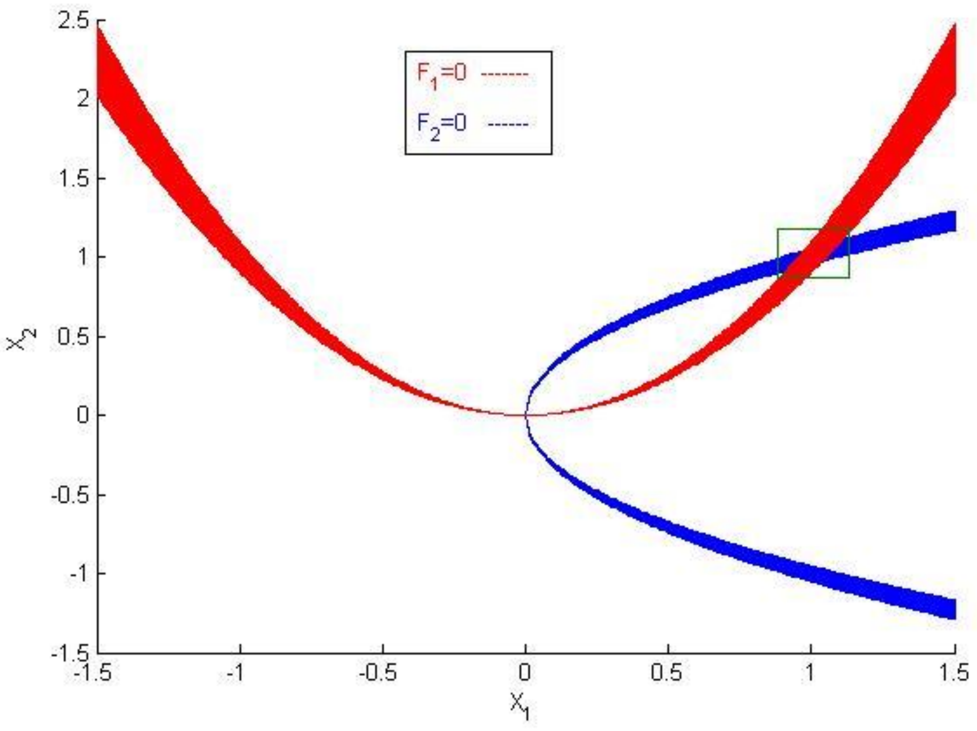

Figure 4.3: Solution of the illustrative example

4.4. Interval Methods for Solving Perturbed Nonlinear Systems of Equations

In this section, we consider the interval methods for solving the problem (4.24) defined in section 4.3; namely, perturbed nonlinear systems of equations.

Let $F: D_f \subseteq \mathbb{R}^n \times P \subseteq \mathbb{IR}^m \rightarrow \mathbb{IR}^n$ be a map that associates with each $x \in D_f \subseteq \mathbb{R}^n$ an interval; given by

$$F(x, P) = \left[\underline{f}(x, \zeta), \overline{f}(x, \xi)\right] \tag{4.27}$$

where $\zeta, \xi \in P \subseteq \mathbb{IR}^{\mathrm{m}}$. Such a map is called a function strip. A zero of $F(x, P)$ is the set of zeros of $f(x, p)$ as p varies over P, i.e.;

$$X^* = \left\{x \in D_f \middle| 0 \in F(x, P)\right\} \tag{4.28}$$

is satisfied, i.e.; $\underline{f}(x, \zeta) \leq 0 \leq \overline{f}(x, \xi)$. The zero set of F, from which an interval box containing this set (which can be empty), X^* is computed.

In the following subsections, we use the interval Newton method to solve the perturbed nonlinear systems and prove the convergence of the method. A parametric version of Hansen-Sengupta, proposed in [10], is introduced. Regarding the Krawczyk method, there is no modification in it. It is used as it is to solve perturbed problems.

4.4.1. Interval Newton Method (INM) for Perturbed Nonlinear Systems

In this subsection, we use the Interval Newton Method (INM) to solve perturbed nonlinear systems. We start with an initial box $X^0 \subseteq D_f \subseteq \mathbb{IR}^n$. Assume $f(x,p)$ is a *continuously differentiable* function of x for each $p \in P$. Recall the interval Newton operator $N(X)$ for real-valued nonlinear systems, see [35]; is given by

$$N(X^{(k)}) = m(X^{(k)}) - \left(F'(X^{(k)})\right)^{-1} f\left(m(X^{(k)})\right) \tag{4.29}$$

and

$$X^{(k+1)} = X^{(k)} \cap N(X^{(k)}) \tag{4.30}$$

for $X^{(k)} \subseteq X^0$. No change in the interval Newton algorithm is necessary. It is already designed to solve this problem. Rewriting the operator to consider the variations or perturbations of the coefficients of the function F; we have

$$N(X^{(k)},P) = m(X^{(k)}) - \left(F'(X^{(k)},P)\right)^{-1} F(m(X),P) \tag{4.31}$$

in terms of the parameters $P \subseteq \mathbb{IR}^m$. In this case, the best that the interval method can do is to compute the smallest box containing the solution set (4.28) or to cover the solution set with a number of small boxes.

To reduce the method's complexity, the interval linear system $F'(X^{(k)},P)Z = -F(m(X^{(k)}),P)$ is solved, instead of calculating the inverse of the interval matrix $F'(X^{(k)},P)$. Hence, (4.31) becomes

$$N(X^k,P) = m(X^k) + Z \tag{4.32}$$

In what follows, we derive the existence and convergence of INM for interval-valued functions (4.31).

Theorem 4.6 Let X be a finite interval $X \subseteq D_f \subseteq \mathbb{IR}^n$. If $N(X,P) \subseteq X$, then there exists a simple zero of $f(x,p)$ in X for each real $p \in P \subseteq \mathbb{IR}^m$.

Proof Assume p is a single parameter to simplify exposition. We show that $f(x,p)$ changes sign in X for each $p \in P$.

Let p be a point in P and let x, y points in X. From the mean value theorem (MVT) for each p

$$f(y,p) = f(x,p) + (y-x)f'(\zeta,p), p \in P$$

where $\zeta \in X$. Thus:

$$f(y,p) \in f(x,p) + (y-x)F'(X,p)$$

for each $p \in P$. If $0 \in F'(X,p)$; then $N(X,P)$ is not finite and $N(X,P) \not\subset X$. But if $0 \notin F'(X,p)$, this assumes that there is no more than one zero of f in X and if there is a zero, it is a simple one; denoted by x^*.

Thus there is at least one $p \in P$ such that $f(x^*,p) = 0$; and we conclude that

$$0 \in F(x^*,P). \blacksquare$$

This proves the existence. Now, we prove the convergence in case of univariate system.

Theorem 4.7 If an interval $X^{(0)}$ contains a zero X^* of $F(x,P)$, then so does $X^{(K)}$ for all $k = 0, 1, 2, \dots$, defined by (4.34). Furthermore, the intervals $X^{(K)}$ form a nested sequence converging to X^* if $0 \notin F'\left(X^{(0)}, P\right)$.

Proof If $0 \notin F'\left(X^{(0)}, P\right)$, then $0 \notin F'\left(X^{(K)}, P\right)$ for all k and $m(X^{(K)})$ is not contained in $N(X^{(K)}, P)$, unless $0 \in F\left(m\left(X^{(K)}\right), P\right)$. Therefore $w\left(X^{(K+1)}\right) < \frac{1}{2} w(X^{(K)})$. Convergence of the sequence follows. ■

Similarly, we can generalize the previous theorem to prove the convergence of the multidimensional systems. To summarize, the parametric Newton iteration is given by:

1) Choose a starting box X^0. Put $\mathrm{k} = 0$.
2) Compute $F'(X^k, P)$ and $F(m(X^k), P)$.
3) Solve the interval linear system $F'(X^k, P)Z = -F(m(X^k), P)$ using the interval Gauss-Seidel method.
4) Calculate the Newton operator
$$N\left(X^{(k)}, P\right) = m\left(X^{(k)}\right) + Z \tag{4.33}$$
5) Calculate the intersection between the previous box with the Newton operator and take a suitable decision considering the five possibilities that may occur and listed below.
$$X^{(k+1)} = X^{(k)} \cap N\left(X^{(k)}, P\right) \tag{4.34}$$
6) Stop when $X^{(k+1)} = X^{(k)}$ or after a certain number of iterations, else $k+1$ and go to 2.

Different possibilities of choosing the initial box, still hold:

1) $N\left(X^{(0)}, P\right) \subset int\left(X^{(0)}\right) \Longrightarrow$ convergence to the unique solution in $X^{(0)}$, where $int\left(X^{(0)}\right)$ represents the topological interior of the box $X^{(0)}$.
2) $N\left(X^{(0)}, P\right) \cap X^{(0)} = \emptyset \Longrightarrow$ no solution in $X^{(0)}$.
3) $N\left(X^{(0)}, P\right) \cap X^{(0)} \neq \emptyset \Longrightarrow$ no conclusion, but we can restart with $X_{NEW}^{(0)} = N\left(X^{(0)}, P\right) \cap X^{(0)}$.
4) $N\left(X^{(0)}, P\right)$ is not defined. In this case, we can bisect $X^{(0)}$ and process each half separately.
5) $N\left(X^{(0)}\right) \cap X^{(0)}$ is the union of two boxes; we can process each of these two boxes separately.

4.4.2. Hansen-Sengupta Method for Perturbed Nonlinear Systems

In this subsection, another method that finds the zero set of (4.27), namely the parametric Hansen-Sengupta, is presented. The parametric Hansen-Sengupta operator is expressed by applying its non-parametric version to different inputs. A more general parametric Hansen-Sengupta was proposed and used in [10].

Theorem 4.8 Let $X, Y, Z \in \mathbb{IR}^n$, $P \in \mathbb{IR}^m$, $\tilde{x} \in X$ and $J \in \mathbb{IR}^{n\times n}$ such that: $X \subseteq Z$, $f(P, \tilde{x}) \subseteq Y$ and $J \supseteq \left\{\frac{df}{dx}(p, x) \in \mathbb{R}^{n\times n} | p \in P, x \in X\right\}$. If X' denotes (4.22) then:
$p \in P$ and $x \in X$ and $f(x, p) = 0$ implies $x \in X'$.
If $\emptyset \neq X' \subseteq int(X)$ then for every $p \in P$, $f(., p)$ has a unique zero in X'.

Proof[11] Fix an arbitrary $\hat{p} \in$ P and define $g: \mathbb{R}^n \to \mathbb{R}^n$ by $g(x) = f(\hat{p}, x)$.

We are going to apply Theorem 4.5 to g. First, $g(\tilde{x}) = f(\hat{p}, x) \in Y$. Second, as $\frac{dg}{dx}(x) = \frac{df}{dx}(\hat{p}, x)$,

$$\left\{\frac{dg}{dx}(x) \in \mathbb{R}^{n\times n} | x \in \mathrm{X}\right\} = \left\{\frac{df}{dx}(\hat{p}, x) \in \mathbb{R}^{n\times n} | x \in \mathrm{X}\right\} \subseteq \mathrm{X}. \tag{4.35}$$

Therefore, Theorem 4.5 can be applied to g and the domain X, and shows that if X' denotes (4.22) then

1) $g(x) = 0$ (that is $f(\hat{p}, x) = 0$) implies $x \in X'$.
2) $\emptyset \neq X' \subseteq int(X)$ implies the existence of a unique zero of g (that is of $f(\hat{p}, .)$) in X'.

This holds for every $\hat{p} \in$ P and hence concludes the proof.■

To compute the interval vector Y, we may use an interval extension of the real-valued function f satisfying $f(\mathrm{P}, \tilde{x}) \subseteq \mathrm{Y}$. Goldsztejn, in his paper [11], uses the mean-value extension to compute Y and the inverse midpoint preconditioning to define the parametric Hansen-Sengupta operator, denoted by $\mathrm{H}_{f,\mathrm{P}}(\mathrm{X}, \mathrm{Z})$:

$$\tilde{x} + \Gamma(C.\mathrm{Y}, -C.\mathrm{f}(\tilde{p}, \tilde{x}) - (C.\mathrm{B}).(\mathrm{P} - \tilde{p}), \mathrm{X} - \tilde{x}), \tag{4.36}$$

with $\mathrm{Y} = \frac{dF}{dx}(\mathrm{P}, \mathrm{X}), \mathrm{B} = \frac{dF}{dp}(\mathrm{P}, \tilde{x}), C = \left(m(Y)\right)^{-1}, \tilde{x} = m(X)$ and $\tilde{p} = m(P)$.

4.5. Two-stage INM for Perturbed Nonlinear Systems

In this subsection, we develop a simple modification to the interval Newton method, denoted by MN, for finding a box that contains the zero set defined by (4.28).

One of the most useful properties of the two-stage interval Newton operator MN is that we are provided with a means of detecting when a region does not contain a root of F. As this is a common situation, it is important that we can quickly discard any set that does not contain any roots. Another important contribution from the properties of MN is the simple verifiable condition that guarantees the existence of a unique root within an interval. The following theorem addresses this.

Theorem 4.9 Suppose f is a *continuous differentiable* function on an initial interval X^0, and $0 \notin F'(X^{(k)})$ for $k = 0,1,2,\dots$. Let MN be defined by (4.37), then if $x^* \in X^{(0)}$ and $MN\left(X^{(k)}, Y^{(k)}, P\right) \subseteq X^{(k)}$, $X^{(k)}$ contains exactly one zero of f. Also

$$x^* \in X^* = \lim_{k\to\infty} X^{(k)}$$

If $X^{(k)} \cap MN\left(X^{(k)}, Y^{(k)}, P\right) = \emptyset$, then $X^{(k)}$does not contain any zero of f.

Proof Part (1) Since $F'\left(X^{(k)}, P\right)$ is non-singular, then $0 \neq f'(x,p)$ for all $x \in X^{(k)}, p \in P$ and therefore f is monotonic on $X^{(k)}$ for every $p \in P$. In other words, it has at most one zero in X for every $p \in P$. Hence, it is sufficient to find a zero $X^* \subseteq X^{(k)}$ that includes all zeros corresponding to all values of p. Since $MN(X^{(k)}, Y^{(k)}, P) \subseteq X^{(k)}$, using Lemma 3.5, so f has exactly one root in $X^{(k)}$ and $X^* = lim_{k\to\infty} X^{(k)}$.

Part (2) Now, suppose X^* is a zero of f and $X^* \subseteq X^0$, then previous part results $X^* \subseteq MN(X^k, Y^k, P)$. Consequently $X^* \subseteq X^k \cap MN(X^k, Y^k, P)$ which is a contradiction. So the proof is completed.■

The convergence of (4.37) can be proved similarly like the proof of Theorem 4.7, if the assumptions of the Theorem 4.9 are hold.

Using interval analysis tools as well as interval Newton method, we consider the following iteration:

1) Choose a starting box X^0. Put $k = 0$.
2) Compute $F'\left(X^{(k)}, P\right)$ and $F\left(m\left(X^{(k)}\right), P\right)$.
3) Solve the interval linear system $F'\left(X^{(k)}, P\right)Z = -F\left(m\left(X^{(k)}\right), P\right)$ using the interval Gauss-Seidel method.
4) Calculate the Newton operator $N\left(X^{(k)}, P\right) = m\left(X^{(k)}\right) + Z$.
5) Calculate a new box Y^k by intersecting the previous box with the Newton operator $Y^{(k)} = X^{(k)} \cap N\left(X^{(k)}, P\right)$.
6) Compute $F\left(m\left(Y^{(k)}\right), P\right)$.
7) Solve the interval linear system $F'\left(X^{(k)}, P\right)Z = -F\left(m\left(Y^{(k)}\right), P\right)$ using the interval Gauss-Seidel method.
8) Calculate the modified Newton operator
$$MN\left(X^{(k)}, Y^{(k)}, P\right) = m\left(Y^{(k)}\right) + Z. \tag{4.37}$$
9) Calculate the new box $X^{(k+1)}$ by intersecting the previous box with the Newton operator $X^{(k+1)} = Y^{(k)} \cap MN\left(X^{(k)}, Y^{(k)}, P\right)$.
10) Stop when $X^{(k+1)} = X^{(k)}$ or after a certain number of iterations, else $k + 1$ and go to 2.

4.6. Numerical Examples

To illustrate the effectiveness of the interval methods in solving perturbed nonlinear systems, we present in this section four test problems. Those test problems are solved using interval Newton, modified Krawczyk, Hansen-Sengupta, and the two-stage interval newton algorithm. The software implementations of those algorithms are presented in Appendix A. Furthermore, the results of the interval methods are compared with the results of traditional methods, which are used to solve the perturbed problems such as the Monte-Carlo methods, to illustrate the superior performance of the interval-based methods over the statistical methods.

Numerical experiments were performed on a computer with an Intel Core2 Duo T7300 with 2.0GHz. The machine ran under control of a Vista Windows operating system. The algorithms and problems were implemented using MATLABR2010a and INTLAB [22].

4.6.1. Univariate Problems

Consider the univariate equation $F: \mathbb{R} \times \mathbb{IR} \rightarrow \mathbb{IR}$

$$F(x,P) = x^3 + Px - 106, \qquad \text{where } P = [1,2].$$

Starting with an initial box $\mathrm{X}^0 = [4,5]$, the original interval Newton and the two-stage interval Newton both find the tightest box in X^0 that contains the solution of this perturbed equation X^* defined by (4.28). The smallest box that is obtained by the two methods is [4.59097696746431,4.66280139735915].

4.6.2. Multivariate Problems

In this subsection, we consider both large and small-sized problems appear in literature as real-valued nonlinear systems. Some modifications are done to those problems to become perturbed problems and be suitable as test problems for the algorithms discussed in sections 4.4 and 4.5.

4.6.2.1. Rosenbrock

Consider the system of equations $\mathrm{F}: \mathbb{R}^{\mathrm{n}} \times \mathbb{IR}^{\mathrm{m}} \rightarrow \mathbb{IR}^{\mathrm{n}}$

$$\begin{gathered} F_i(x,A) = 1 - P_i x_i, \qquad i = odd \\ F_i(x,P) = 10(P_i x_i - x_{i-1}^2), \qquad i = even \\ P_i = \begin{cases} [0.9,1.1] & , i = 1, \dots, 5 \\ 1 & , i = 6, \dots, m \end{cases} \end{gathered} \tag{4.38}$$

This problem is solved as a real-valued system in [53] and is solved here as a perturbed problem after adding some parameters to it. Table 4.1 shows the consumed time in solving (4.38) for different number of unknowns/equations using Krawczyk, interval Newton, Hansen-Sengupta, two-stage interval Newton, and the Monte-Carlo methods, where n represents the number of unknowns/equations.

Table 4.1: Results of Rosenbrock

Method	**Time (Sec)**			
	n=10	**n=50**	**n=100**	**n=300**

Modified Krawczyk	3.874190433651686	11.430137954703696	20.871010874134605	59.283599877169245
Newton	0.234718408849069	1.499954020572510	2.831574072857687	8.587294816804825
Hansen-Sengupta	0.247091801211230	0.966054456063373	1.748448604852800	5.187728255015042
Two-Stage Newton	0.261349939569895	1.722768273538183	3.379549500137939	10.192865224445292
Monte Carlo (Random sampling)	1115.6	2676.0	4734.2	23497.7

4.6.2.2. Broyden

Consider the system of equations $F: \mathbb{R}^n \times \mathbb{IR}^m \to \mathbb{IR}^n$

$$\begin{gathered} F_1(x,P) = -(3 + P_1x_1)x_1 + 2x_2 + 1, \\ F_i(x,P) = x_{i-1} - (3 + P_ix_i)x_i + 2x_{i+1} + 1, \qquad i = 2,3,\dots,n-1 \\ F_n(x,P) = x_{n-1} - (3 + P_nx_n)x_n + 1, \\ P_i = \begin{cases} [0.9,1.1] & ,i = 1,\dots,5 \\ 1 & ,i = 6,\dots,m \end{cases} \end{gathered} \tag{4.39}$$

This problem is solved as a real-valued system in [4] and is solved here as a perturbed problem after adding some parameters to it. Table 4.2 shows the consumed time in solving (4.39) for different number of unknowns/equations using Krawczyk, interval Newton, Hansen-Sengupta, two-stage interval Newton, and the Monte-Carlo methods, where n represents the number of unknowns/equations.

Table 4.2: Results of Broyden

Method	**Time (Sec)**			
	n=10	**n=50**	**n=100**	**n=300**
Modified Krawczyk	16.113261287949452	97.107854189382707	159.2457731769157	194.6128437783266
Newton	4.560893593752053	23.112668892289907	45.754363529774970	51.701028214947257
Hansen-Sengupta	5.045687950762602	65.6875989378752	114.8743304245872	595.5990343499167
Two-Stage Newton	4.327699074499810	24.844256184233000	45.826137416087548	51.031945849371397
Monte Carlo (Random sampling)	1859.2	8108.6	22341.3	185946.9

4.6.2.3. Interval Arithmetic Benchmark 1

We consider one of the benchmark problems proposed in the interval arithmetic community found in [21], [60], and [40]. The benchmark is modified by adding some interval parameters and becomes:

$$\begin{aligned}
&P_1x_1 - 0.25428722 - 0.18324757x_4x_3x_9 = 0,\\
&P_2x_2 - 0.37842197 - 0.16275449x_1x_{10}x_6 = 0,\\
&P_3x_3 - 0.27162577 - 0.16955071x_1x_2x_{10} = 0,\\
&P_4x_4 - 0.19807914 - 0.15585316x_7x_1x_6 = 0,\\
&P_5x_5 - 0.44166728 - 0.19950920x_7x_6x_3 = 0,\\
&P_6x_6 - 0.14654113 - 0.18922793x_8x_5x_{10} = 0,\\
&P_7x_7 - 0.42937161 - 0.21180486x_2x_5x_8 = 0,\\
&P_8x_8 - 0.07056438 - 0.17081208x_1x_7x_6 = 0,\\
&P_9x_9 - 0.34504906 - 0.19612740x_{10}x_6x_8 = 0,\\
&P_{10}x_{10} - 0.42651102 - 0.21466544x_4x_8x_1 = 0
\end{aligned} \tag{4.40}$$

where $P_i = \begin{cases}[0.95,1.05], & i = 1,2\\ 1, & i = 3,\dots,10\end{cases}$

The final results for (4.40), time consumption, and number of iterations are shown below in Table 4.3 given that the initial box for interval methods is

$$X^{(0)} = \begin{pmatrix}[0.2,0.3]\\ [0.35,0.45]\\ [0.25,0.3]\\ [0.17,0.22]\\ [0.4,0.45]\\ [0.1,0.15]\\ [0.4,0.45]\\ [0.05,0.1]\\ [0.3,0.35]\\ [0.4,0.45]\end{pmatrix},$$

and the initial point for the traditional algorithm is $x^{(0)} = (1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1)^T$.

Table 4.3: Results for Interval Arithmetic Benchmark 1

Method	Results	Time (Sec)	Number of iterations
Krawczyk	[0.24416289903528,0.27150387840325] [0.36074134097093,0.40145295339647] [0.27778693567346,0.27970310561291] [0.20051855613113,0.20081936860206] [0.44523347043033,0.44526944486158] [0.14917476207366,0.14919303329010] [0.43179927385240,0.43222015171078] [0.07323793338412,0.07356761809407] [0.34596284355806,0.34597075062816] [0.42725472344260,0.42739782515178]	7.424643076145153	100

Hansen-Sengupta	[0.24486774189222,0.27141678271063] [0.36099566945698,0.40130370443897] [0.27795228356957,0.27953774866458] [0.20053144814019,0.20080648031009] [0.44523987437411,0.44526297527635] [0.14917814274224,0.14918969719623] [0.43186375578489,0.43215564218231] [0.07325206277927,0.07355349277278] [0.34596480882303,0.34596884492807] [0.42728085835085,0.42737169363288]	0.875779673114879	8
Newton	[0.24486774189222,0.27141678271063] [0.36099566945698,0.40130370443897] [0.27795228356957,0.27953774866458] [0.20053144814019,0.20080648031009] [0.44523987437411,0.44526297527635] [0.14917814274224,0.14918969719623] [0.43186375578489,0.43215564218231] [0.07325206277927,0.07355349277278] [0.34596480882303,0.34596884492807] [0.42728085835085,0.42737169363288]	0.870657024845336	8
Two-stage Newton	[0.24486774189222,0.27141678276708] [0.36098694204337,0.40130370443960] [0.27795227955805,0.27953775287234] [0.20053144813923,0.20080648031109] [0.44523987432196,0.44526297533103] [0.14917814274190,0.14918969719660] [0.43186375572169,0.43215564224552] [0.07325206277821,0.07355349277387] [0.34596480882301,0.34596884492808] [0.42728085835066,0.42737169363308]	0.779798363789230	6
Monte Carlo (Random Sampling)	[0.24554526258497,0.27141651330731] [0.36282756226301,0.40130367953891] [0.27807982885538,0.27951827118235] [0.20054414090180,0.20080642540632] [0.44524105581311,0.44526271827972] [0.14917846220456,0.14918969204452] [0.43187829074617,0.43215534789383] [0.07326597380314,0.07355343259930] [0.34596478233034,0.34596884361277] [0.42728536799113,0.42737157562664]	144.5009847620298	10000

4.6.2.4. Interval Arithmetic Benchmark 2

This problem is described in [21] and is implemented here with slight modifications in some coefficients.

$0.5P_1x_1 + x_2 + 0.5x_3 - \frac{x_6}{x_7} = 0,$
$P_2x_3 + x_4 + 2x_5 - \frac{2}{x_7} = 0,$
$P_3x_1 + x_2 + x_5 - \frac{1}{x_7} = 0,$
$-28837P_4x_1 - 139009x_2 - 78213x_3 + 18927x_4 + 8427x_5 + \frac{13492}{x_7} - \frac{10690x_6}{x_7} = 0,$
$x_1 + P_5x_2 + x_3 + x_4 + x_5 - 1 = 0,$
$P_6k^2x_1x_4^3 - 1.7837 \times 10^5x_3x_5 = 0,$
$P_7x_1x_3 - 2.6058x_2x_4 = 0$

where $\begin{matrix} k = 20, \\ P_i = [0.99,1.01],\ i = 1,\dots,7 \end{matrix}$

The final results, time consumption, and number of iterations are shown below in Table 4.4 given that the initial box for interval methods is

$$X^{(0)} = \begin{pmatrix} [0.2,0.4] \\ [0.0,0.2] \\ [0.0,0.2] \\ [0.5,0.7] \\ [0.0,0.2] \\ [0.5,0.7] \\ [2.9,3.1] \end{pmatrix},$$

and the initial point for the traditional algorithm is $x^{(0)} = (0.3 \quad 0.3 \quad 0.3 \quad 0.3 \quad 0.3 \quad 0.3 \quad 0.3)^T$.

Table 4.4: Results for Interval Arithmetic Benchmark 2

Method	Results	Time (Sec)	Number of iterations
Krawczyk	[0.20748973714159,0.39942395366294] [0.00000000000000,0.05149833650428] [0.01588928107892,0.08351971318773] [0.50000000000000,0.69463030923352] [0.00000000000000,0.20000000000001] [0.50000000000000,0.70000000000000] [2.89999999999999,3.10000000000001]	5.8993543872196	100
Hansen-Sengupta	[0.32011969813163,0.32562198082145] [0.00880867411437,0.00963841296401] [0.04387277149462,0.04816141042664] [0.61590283640151,0.62044051374013] [0.00345963586837,0.00397406603726] [0.56715680879645,0.58627398307465] [2.96603521787489,2.98969168370741]	2.0949346215790	22
Newton	[0.32011969813163,0.32562198082145] [0.00880867411437,0.00963841296401] [0.04387277149462,0.04816141042664] [0.61590283640151,0.62044051374013] [0.00345963586837,0.00397406603726] [0.56715680879645,0.58627398307465]	1.7686881293572	21

	[2.96603521787488,2.98969168370741]		
Two-stage Newton	[0.32011969813163,0.32562198082145] [0.00880867411437,0.00963841296401] [0.04387277149462,0.04816141042664] [0.61590283640151,0.62044051374013] [0.00345963586837,0.00397406603726] [0.56715680879645,0.58627398307465] [2.96603521787489,2.98969168370741]	1.7594493154856	17
Monte Carlo (Random Sampling)	[0.32056783572975,0.32505436486501] [0.00890875411498,0.00953286539409] [0.04405737953297,0.04800753227533] [0.61620567847531,0.62021168954790] [0.00349906315363,0.00392798375378] [0.56800958737446,0.58477447845075] [2.96895847942973,2.98535848254923]	162.55809781690	10^7

We conclude from the previous examples that the interval algorithms used for solving perturbed nonlinear systems have *faster responses* than the Monte-Carlo methods. The difference in consumed time becomes more obvious when the number of interval parameters or the size of the problem itself increases. However, the interval algorithms are not always convergent and their convergence depends on the width of the interval parameters. In the previous examples, the convergence is lost when retesting the problems with wider interval parameters.

The results show that interval Newton and Hansen-Sengupta algorithms are nearly similar in performance and both of them are faster than the modified Krawczyk algorithm. Whilst, the new proposed algorithm, the two-stage interval Newton, shows the *best* performance from the execution time preview in most problems that are tested. The results generated from the two-stage interval Newton are *equal in width* with the results coming from the interval Newton.

Another important issue regarding applying the interval methods to find the solutions of the nonlinear systems is choosing the initial box. It can happen that little or no progress is made in reducing the size of the current box (in the first step or during a step of the method). In case of that happens in the first step, i.e. the initial box, we should consider the different possibilities mentioned in section 4.4.1. And for the little progress during the method, it is common practice to divide the box in half (say) and apply the algorithm to each sub-box separately.

Chapter 5 : Engineering Applications of Nonlinear Systems

To illustrate the applicability of the interval methods mentioned in Chapter 4, we shall consider more practical examples representing engineering problems in this chapter. The problems discussed here are related to many engineering fields such as electrical circuits, fluid mechanics, and dynamics of rigid body.

5.1. Zener Diodes

This problem models a circuit described in [31]. It contains two zener diodes as the active elements. The problem is implemented here with slight modifications in some coefficients (5.1) and results are listed in Table 5.1. While Figure 5.1 illustrates the large difference in time consumption between the interval methods and the traditional methods, Figure 5.2 gives a closer view on the time consumed by each interval method.

$$\begin{aligned}
&2P_1(e^{x_1}-1)-3.86548x_1+0.38126x_2+0.14836x_3+0.17986x_4=0,\\
&3P_2(e^{x_2}-1)-14.9484x_1-0.00764x_2+0.97901x_3-11.568x_4+9=0,\\
&2P_3(e^{x_3}-1)-13.3092x_1+4.99094x_2+4.25872x_3-9.76315x_4+8=0,\\
&5P_4(e^{x_4}-1)+8.91431x_1-3.34286x_2-4.17818x_3-2.61661x_4-5=0
\end{aligned} \tag{5.1}$$

where $\mathrm{P_i}=[0.99,1.01],\ \mathrm{i}=1,2,3,4$ and x_i denotes the diode voltage $v_i, i=1,2,3,4$.

Table 5.1: Results of Zener diodes problem

Method	Initial vector	Result	Time (Sec) /Number of iterations
Modified Krawczyk	[-0.300000000,-0.100000000] [-2.000000000,-1.199999999] [0.599999998,1.100000001] [0.800000000,1.000000000]	[-0.19785859125236,-0.14504558212951] [-1.86146862397589,-1.44807584435096] [0.73239721799372,0.99246500025139] [0.83214007241916,0.89546042209975]	7.3921638783700/ 100
Newton		[-0.18121217143835,-0.16162919190336] [-1.72753898836348,-1.58160264833527] [0.82090387675569,0.90354292117435] [0.85169368736668,0.87570332414066]	1.3452033009782/ 22
Hansen-Sengupta		[-0.18121217143835,-0.16162919190336] [-1.72753898836347,-1.58160264833528] [0.82090387675570,0.90354292117434] [0.85169368736668,0.87570332414066]	1.5275387654017/ 23
Two-Stage Newton		[-0.18121217143835,-0.16162919190336] [-1.72753898836347,-1.58160264833527] [0.82090387675570,0.90354292117434] [0.85169368736668,0.87570332414066]	1.0583995312253/ 15
Exhaustive Sampling	0.2 0.2 0.5 1	[-0.18037954837969,-0.16248175343666] [-1.72081196443950,-1.58797608044654] [0.82511328733773,0.89897357432029] [0.85283843762367,0.87472550116871]	123.42337308233/ 10^4
Monte Carlo (Random sampling)		[-0.17925876105317,-0.16269738302663] [-1.71725855591382,-1.59183120151122] [0.82694019923042,0.89734239059338] [0.85309409210239,0.87354559084360]	113.86178781102/ 10^4

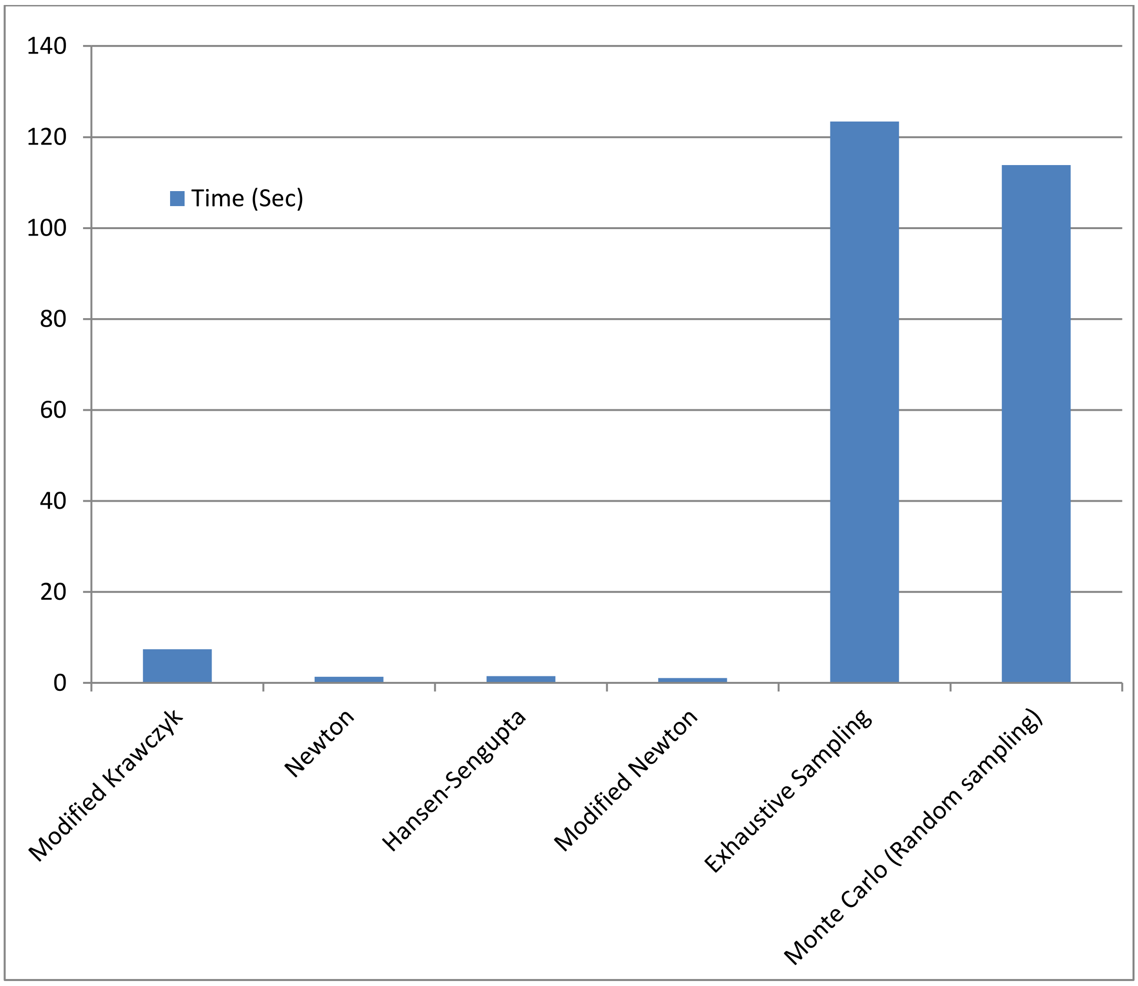

Figure 5.1: Consumed time by interval and traditional methods to solve (5.1)

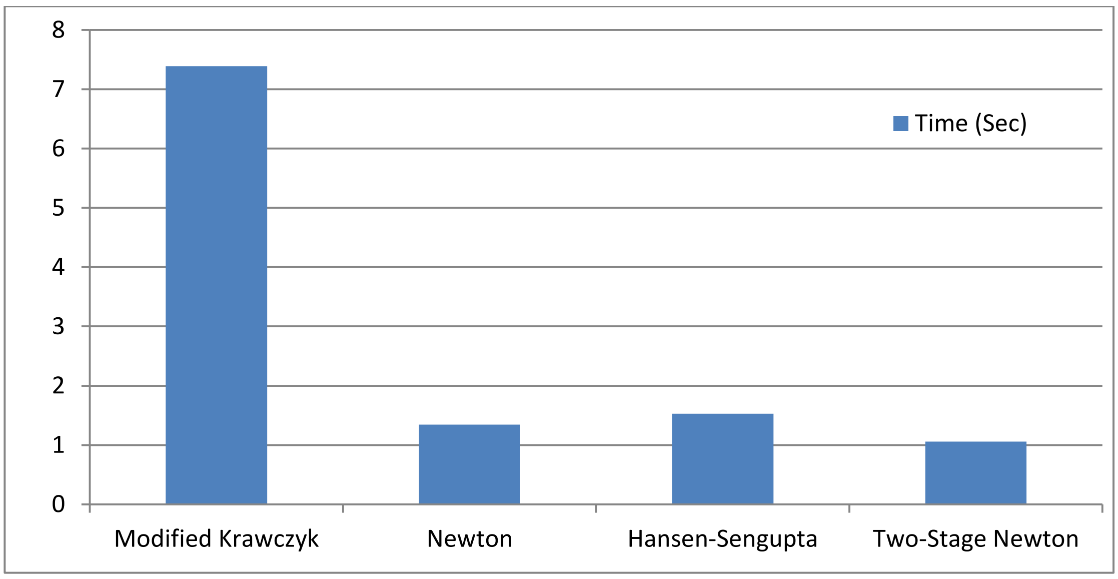

Figure 5.2: Consumed time by the four interval methods to solve (5.1)

5.2. Two Tunnel Diodes

This problem models a circuit described in [31] and shown in Figure 5.3. It contains two tunnel diodes, a linear resistor, and a voltage source connected in series. The problem is implemented here with slight modifications in some coefficients (5.2) and results are listed in Table 5.2. While Figure 5.4 illustrates the large difference in time consumption between the interval methods and the traditional methods, Figure 5.5 gives a closer view on the time consumed by each interval method.

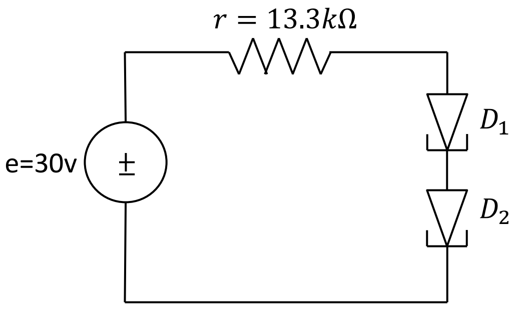

Figure 5.3: Nonlinear circuit given by (5.2)

$$
\begin{aligned}
&\mathrm{e} - \mathrm{P}_1\mathrm{r}(2.5\mathrm{x}_1^3 - 1.5\mathrm{x}_1^2 + 11.8\mathrm{x}_1) - \mathrm{x}_1 - \mathrm{x}_2 = 0, \\
&\mathrm{P}_2(2.5\mathrm{x}_1^3 - 1.5\mathrm{x}_1^2 + 11.8\mathrm{x}_1) - 0.43\mathrm{x}_2^3 + 2.69\mathrm{x}_2^2 - 4.5\mathrm{x}_2 = 0,
\end{aligned} \tag{5.2}
$$

where $P_i = [0.99,1.01],\ i = 1,2$
$r = 13.3k\Omega,$ and $e = 30V$

Table 5.2: Results of two tunnel diodes problem

Method	Initial vector	Result	Time (Sec) /Number of iterations
Modified Krawczyk	[1.600000000,1.700000000] [0.699999998,0.800000001]	[1.61341211121028,1.70000000000000] [0.69999999999998,0.80000000000001]	3.4030560448325/ 100
Newton		[1.65588719410402,1.67704108398090] [0.72287843197481,0.80000000000001]	0.2008001715302/ 5
Hansen-Sengupta		[1.65588719410403,1.67704108398089] [0.72287843197483,0.80000000000001]	0.2040561021024/ 5
Two-Stage Newton		[1.65528794069157,1.67828214063470] [0.72135752089704,0.80000000000001]	0.0904656876781/ 2
Exhaustive Sampling	0.5 0.5	[0.22485309291253,0.23066576172409] [0.83553527290137,0.94181003279859]	122.93846522393/ 10^4
Monte Carlo (Random sampling)		[0.22486037203781,0.23062166748250] [0.83562453751020,0.94064336366313]	122.23806936356/ 10^4

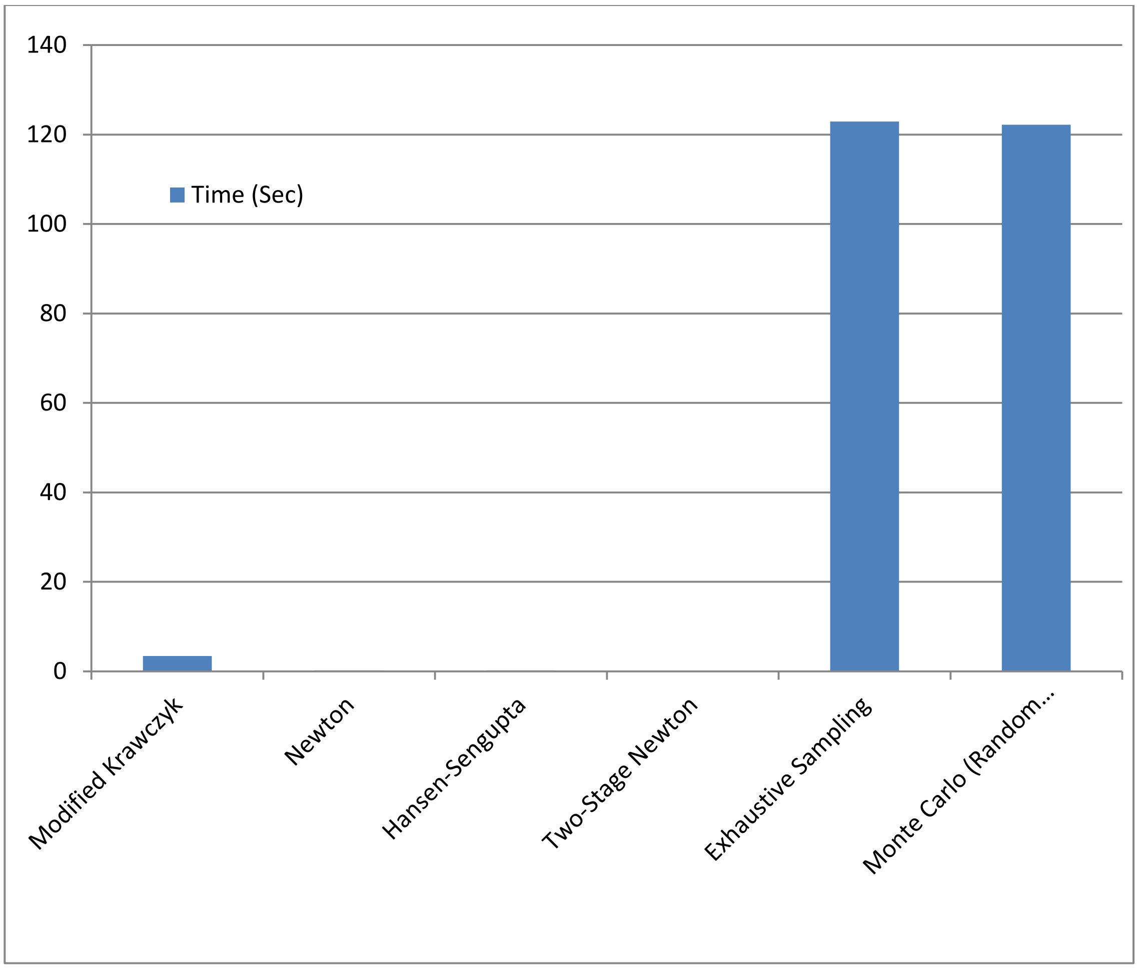

Figure 5.4: Consumed time by interval and traditional methods to solve (5.2)

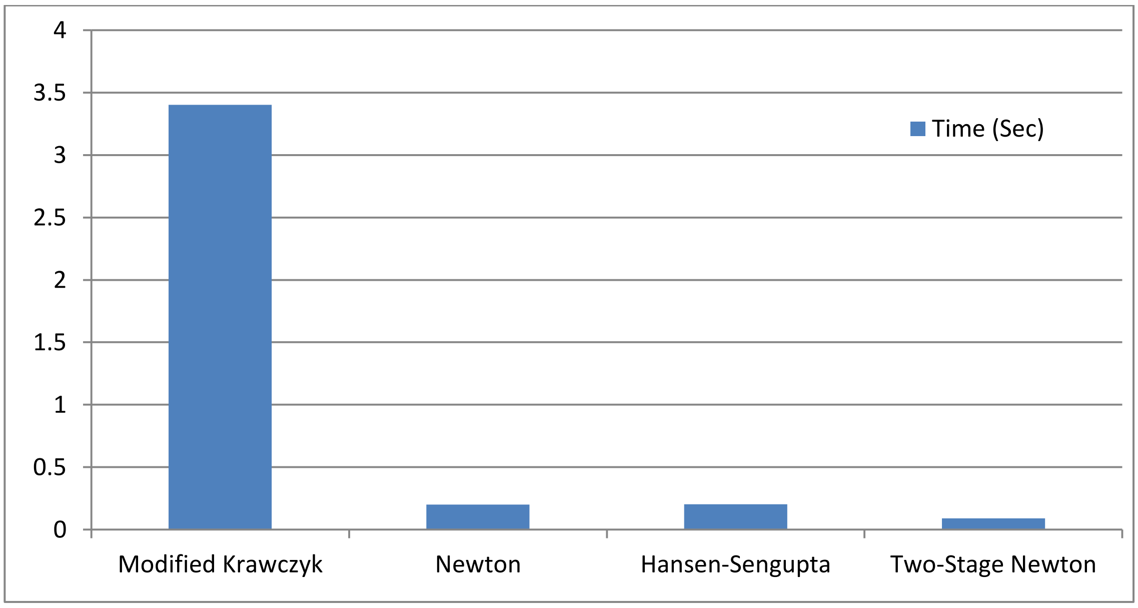

Figure 5.5: Consumed time by the four interval methods to solve (5.2)

5.3. Three Bar Mechanism

This problem models a mechanical system described in [59]. It consists of three bars. The problem is implemented here with slight modifications in some coefficients (5.3) and results are listed in Table 5.3. While Figure 5.6 illustrates the large difference in time consumption between the interval methods and the traditional methods, Figure 5.7 gives a closer view on the time consumed by each interval method.

$$\begin{aligned} &-\mathrm{A}_1\mathrm{L}_2 \sin \mathrm{x}_1 + \mathrm{L}_3 \sin \mathrm{x}_2 - \mathrm{L}_1 \sin \alpha = 0, \\ &\mathrm{L}_2 \cos \mathrm{x}_1 + \mathrm{A}_2\mathrm{L}_3 \cos \mathrm{x}_2 - \mathrm{L}_1 \cos \alpha - \mathrm{D} = 0, \end{aligned} \tag{5.3}$$

where

$$\begin{aligned} \mathrm{A}_\mathrm{i} = [0.99, 1.01],\ \mathrm{i} = 1,2 \\ \alpha = 1, \\ \mathrm{L}_1 = 6, \\ \mathrm{L}_2 = 3, \\ \mathrm{L}_3 = 7, \\ \mathrm{D} = 4 \end{aligned}$$

Table 5.3: Results of three bar mechanism problem

Method	Initial vector	Result	Time (Sec)
Modified Krawczyk	[0.000000000,0.500000000] [0.699999998,1.200000000]	[0.05121569038858,0.27622684283914] [0.86395803618009,0.96334550130593]	3.0659874369508/ 100
Newton		[0.15182972272616,0.17549676981885] [0.90541837620206,0.91975780675547]	0.5946115358237/ 14
Hansen-Sengupta		[0.15182972272617,0.17549676981885] [0.90541837620206,0.91975780675547]	0.9466079487756/ 14
Two-Stage Newton		[0.15182972272617,0.17549676981885] [0.90541837620206,0.91975780675547]	0.3935273198130/ 11
Exhaustive Sampling	0.5 0.5	[0.15219604746192,0.17496974590696] [0.90551426416666,0.91949001631011]	111.60235560665/ 10^4
Monte Carlo (Random sampling)		[0.15226451004175,0.17474357691244] [0.90553359627386,0.91947264958054]	108.36872263095/ 10^4

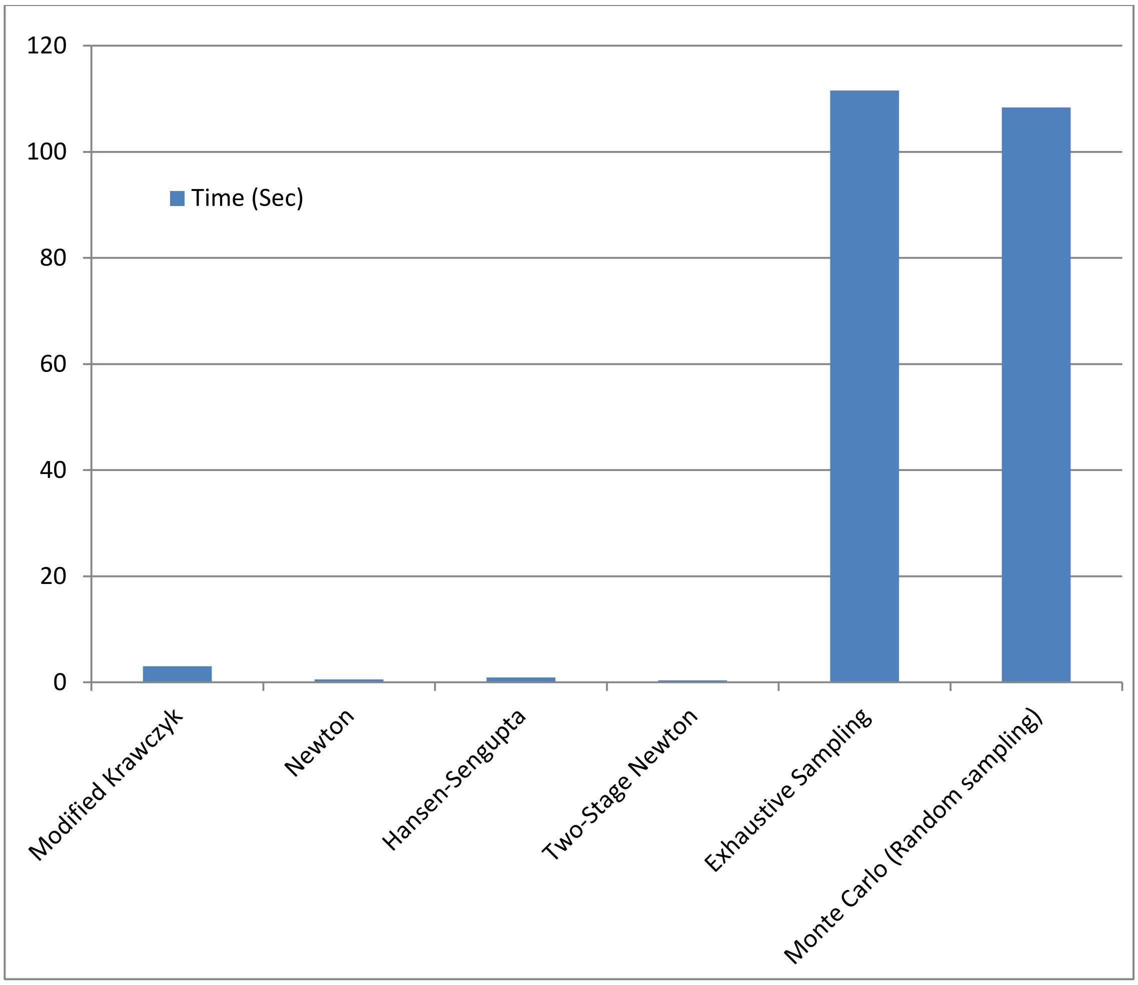

Figure 5.6: Consumed time by interval and traditional methods to solve (5.3)

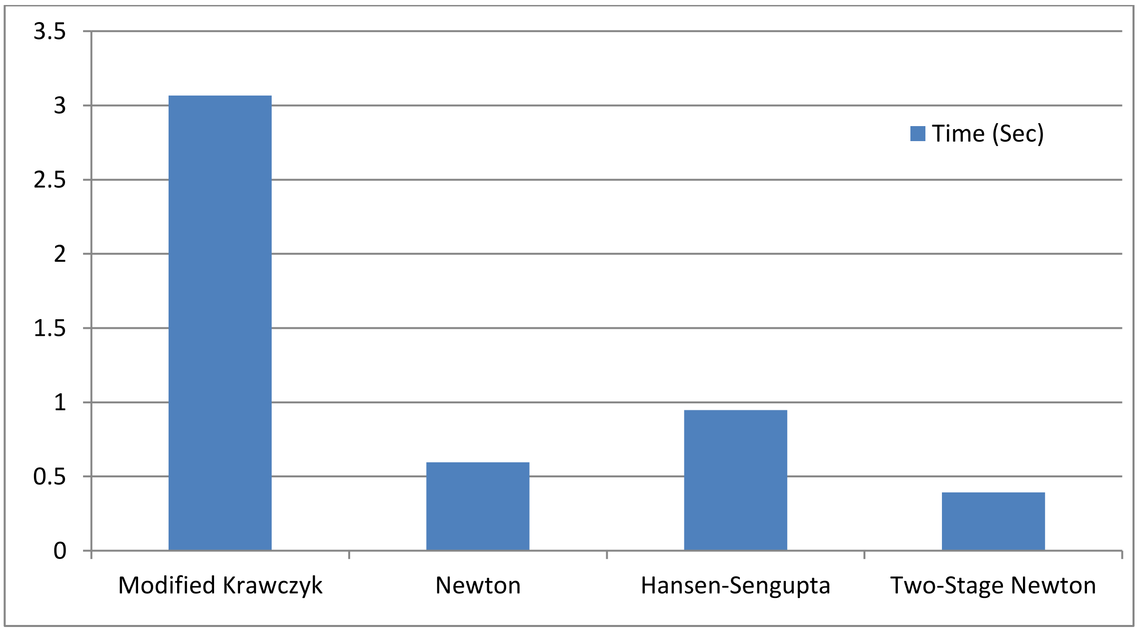

Figure 5.7: Consumed time by the four interval methods to solve (5.3)

5.4. Pipe Pump Problem

This problem models the flow through a horizontal pipe between two reservoirs (for more details see [59]). The problem is implemented here with slight modifications in some coefficients and results are listed in Table 5.4. While Figure 5.8 illustrates the large difference in time consumption between the interval methods and the traditional methods, Figure 5.9 gives a closer view on the time consumed by each interval method.

$$\begin{aligned} & x_2 - P_1 h - \frac{1.074 L x_1^2}{g D^5 \left(\ln \left(\frac{e}{3.75D} + 4.618 \left(\frac{D\upsilon}{x_1} \right)^{0.9} \right) \right)^2} = 0, \\ & P_2 x_2 - a - b x_1 - c x_1^2 = 0, \end{aligned} \tag{5.4}$$

$$\text{where } \begin{aligned} & \mathrm{h} = 20, \mathrm{L} = 150, \mathrm{g} = 9.806, \mathrm{e} = 0.00001, \mathrm{D} = 0.25, \\ & \upsilon = 10^{-6}, \mathrm{a} = 500, \mathrm{b} = 0, \mathrm{c} = -10, \\ & \mathrm{P_i} = [0.99, 1.01],\ i = 1,2 \end{aligned}$$

Table 5.4: Results of pipe pump problem

Method	Initial vector	Result	Time (Sec)
Modified Krawczyk	[1.6000000000,2.0000000000] [462.00000000,472.00000000]	[1.809539777879,1.839061797381] [462.007191354313,471.431126330832]	2.4039322036739/ 100
Newton		[1.814751391495,1.833856950443] [462.322316300895,471.115994188294]	0.2954694660914/ 10
Hansen-Sengupta		[1.814751391495,1.833856950443] [462.322316300895,471.115994188294]	0.3273747082381/ 9
Two-Stage Newton		[1.814751391495,1.833856950443] [462.322316300895,471.115994188294]	0.3076946930406/ 8
Exhaustive Sampling	0.5 0.5	[1.81495690432063,1.83377110753643] [462.407021312768,471.112145676090]	164.78153648019/ 10^4
Monte Carlo (Random sampling)		[1.81501691348726,1.83375935545884] [462.418521303061,471.110779097566]	162.55809781690/ 10^4

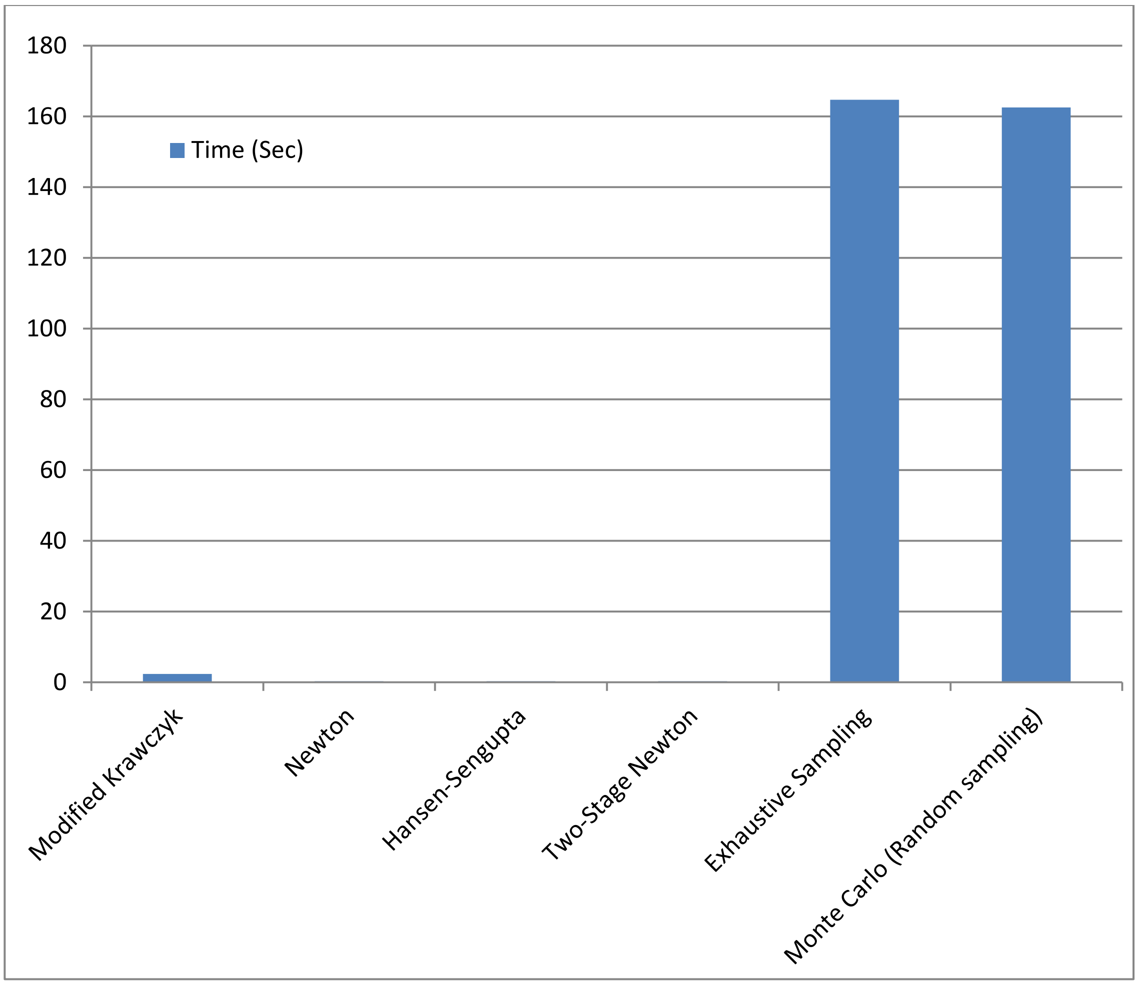

Figure 5.8: Consumed time by interval and traditional methods to solve (5.4)

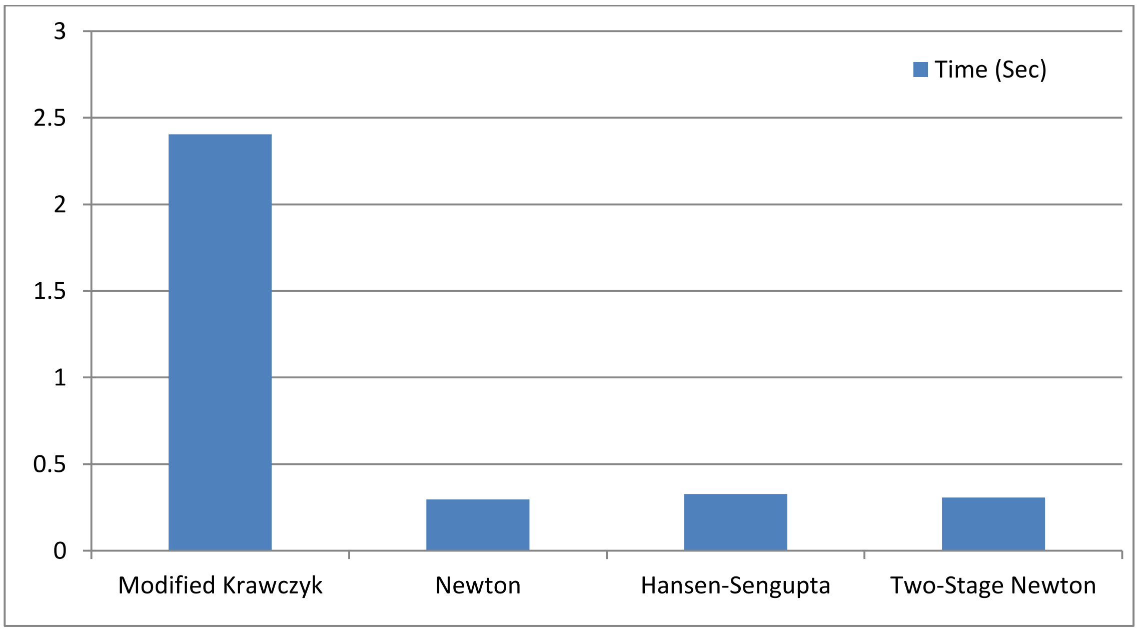

Figure 5.9: Consumed time by the four interval methods to solve (5.4)

5.5. Pipe Flow Problem

This problem models another pipe flow system (for more details see [59]). The problem is implemented here with slight modifications in some coefficients and results are listed in Table 5.5. While Figure 5.10 illustrates the large difference in time consumption between the interval methods and the traditional methods, Figure 5.11 gives a closer view on the time consumed by each interval method.

$$\begin{aligned} &P_1 x_1 - \frac{1.3254}{\left(\ln\left(\frac{e}{3.75x_2} + 4.618\left(\frac{x_2 \upsilon}{Q}\right)^{0.9}\right)\right)^2} = 0, \\ &P_2 h_f - \frac{8x_1 L Q^2}{\pi^2 g x_2^5} = 0, \end{aligned} \tag{5.5}$$

where $\mathrm{h}_f = 2, \mathrm{L} = 100, \mathrm{g} = 9.806, \mathrm{e} = 0.0001, \upsilon = 10^{-5}, \mathrm{Q} = 0.5,$
$P_i = [0.99,1.01],\ i = 1,2$

Table 5.5: Results of pipe flow problem

Method	Initial vector	Result	Time (Sec)
Modified Krawczyk	[0.0000000000,0.2000000001] [0.2999999998,0.6000000000]	[0.01537936747196,0.02081941065837] [0.29999999999998,0.60000000000000]	2.6868514713462/ 100
Newton		[0.01528432073079,0.02094913035837] [0.29999999999998,0.60000000000000]	0.1516917652942/ 4
Hansen-Sengupta		[0.01528432073079,0.02094913035837] [0.29999999999998,0.60000000000000]	0.1318404294401/ 4
Two-Stage Newton		[0.01541234911527,0.02079538396412] [0.29999999999998,0.60000000000000]	0.0741305808420/ 2
Exhaustive Sampling	0.5 0.5	[0.01791762262596,0.01829055887115] [0.44940042155211,0.45306449489084]	113.61246736666/ 10^4
Monte Carlo (Random sampling)		[0.01792168393879,0.01828832649226] [0.44942629983675,0.45305169867925]	121.77743672729/ 10^4

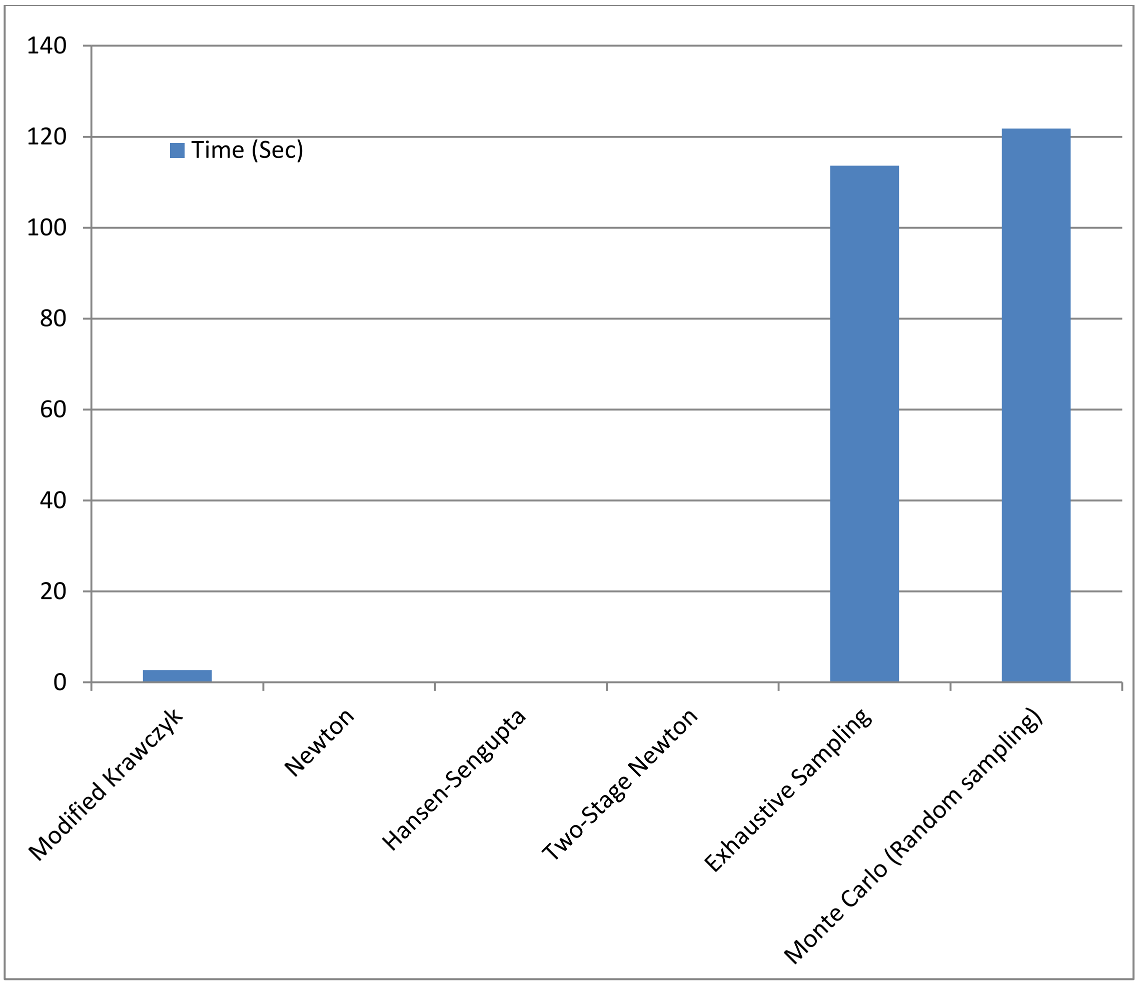

Figure 5.10: Consumed time by interval and traditional methods to solve (5.5)

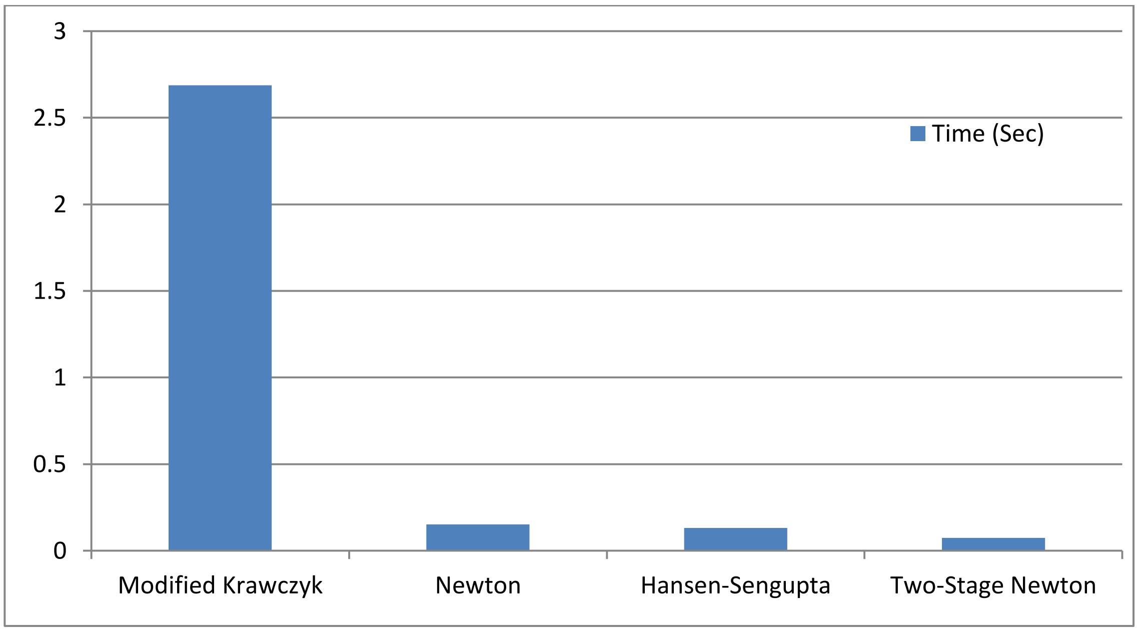

Figure 5.11: Consumed time by the four interval methods to solve (5.5)

5.6. Manning's Equation

Manning's equation is used to calculate the discharge on an open channel (for more details see [59]). The problem is implemented here with slight modifications in some coefficients and results are listed in Table 5.6. While Figure 5.12 illustrates the large difference in time consumption between the interval methods and the traditional methods, Figure 5.13 gives a closer view on the time consumed by each interval method.

$$
\begin{aligned}
&Q - \frac{C_u}{n} \frac{x_1^{\frac{5}{3}}}{x_3^{\frac{2}{3}}} \sqrt{A_1 S_0} = 0, \\
&x_1 - (A_2 b + A_3 z x_2) x_2 = 0, \\
&x_3 - A_2 b - 2x_2 \sqrt{1 + A_3 z^2} = 0,
\end{aligned}
\tag{5.6}
$$

where $\mathrm{b} = 1.5, \mathrm{z} = 0.5, \mathrm{n} = 0.023, S_0 = 0.00001, Q = 0.15, C_u = 1,$
$A_i = [0.99,1.01],\ i = 1,2,3$

Table 5.6: Results of the Manning's equation

Method	Initial vector	Result	Time (Sec)
Modified Krawczyk	[1.6000000000,1.8000000001] [0.8000000000,1.0000000000] [3.3999999999,3.6000000001]	[1.70712502508040,1.76986930627666] [0.87114558063228,0.91508047484763] [3.43087129905698,3.56324958711348]	2.5477020124066/ 100
Newton		[1.72190037835037,1.75511444766128] [0.87868357291775,0.90755287172710] [3.44776586642814,3.54637825060974]	0.8015268700352/ 20
Hansen-Sengupta		[1.73064581664707,1.74633746743833] [0.88293065740411,0.90329260696013] [3.48516613115853,3.50895051095151]	0.6229344092615/ 14
Two-Stage Newton		[1.72190037835037,1.75511444766128] [0.87868357291775,0.90755287172710] [3.44776586642814,3.54637825060974]	0.7380743921364/ 14
Exhaustive Sampling	0.2 0.2 0.5	[1.73110543210121,1.74597480821561] [0.88335854520642,0.90308985618793] [3.48595025732874,3.50826738786179]	12228.059960973/ 10^6
Monte Carlo (Random sampling)		[1.73125896506238,1.74588630494490] [0.88343726154489,0.90286648928931] [3.48614162061702,3.50813841220765]	11397.366936300/ 10^6

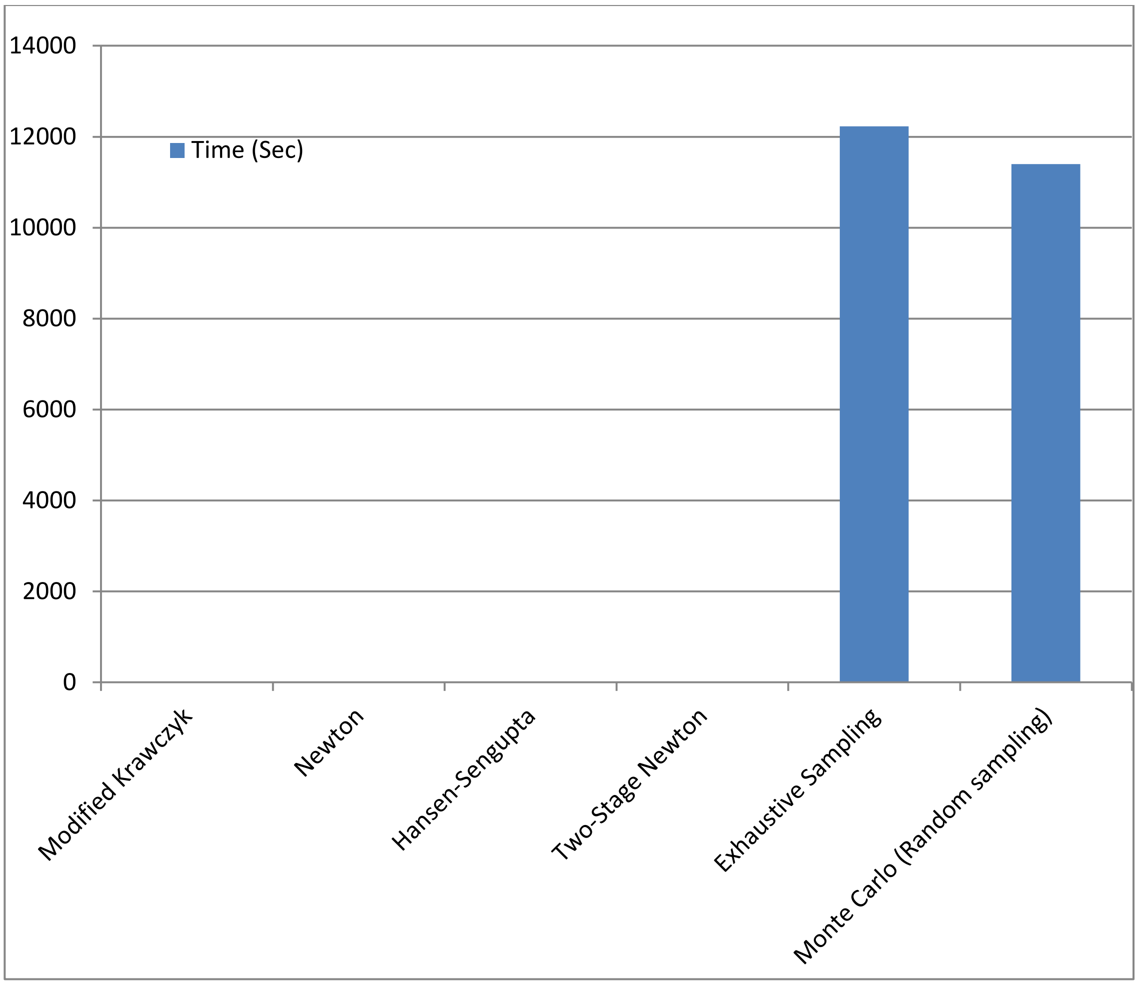

Figure 5.12: Consumed time by interval and traditional methods to solve (5.6)

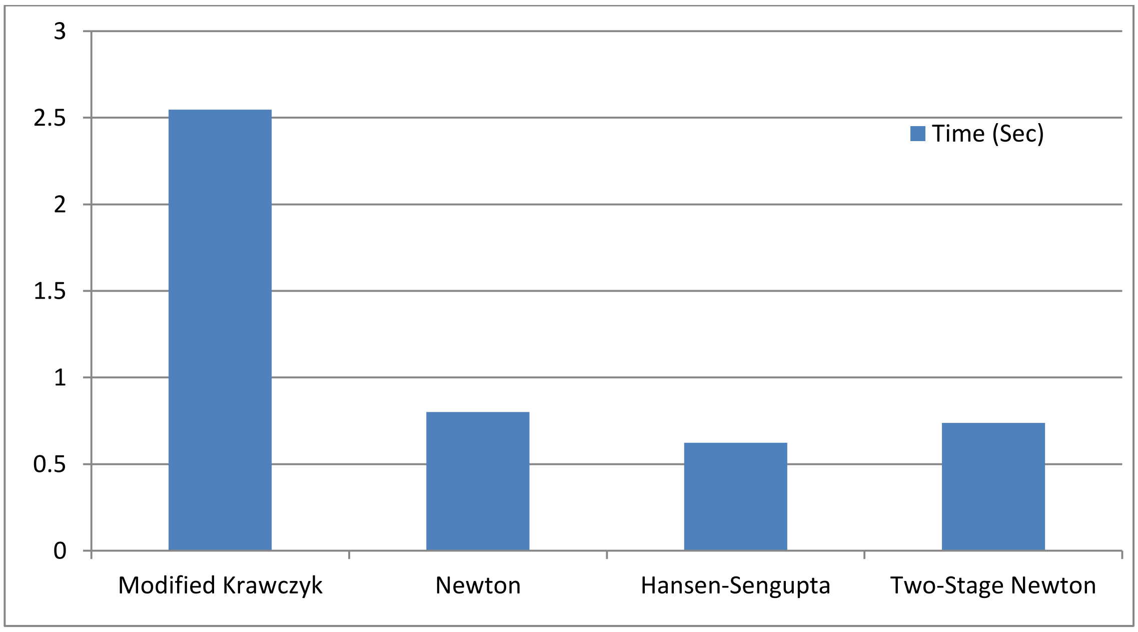

Figure 5.13: Consumed time by the four interval methods to solve (5.6)

Chapter 6 : Conclusion

This chapter offers a concluding view to the results obtained throughout this thesis. After that, section 6.3 provides a summary of contributions of this thesis and outlines several directions for future research.

6.1. A View to Solving Perturbed Problems Using Interval Arithmetic

Each algorithm has its advantages and disadvantages, some are more accurate and some are faster. A comparison among verified computing algorithms (interval Newton, Krawczyk, Hansen-Sengupta, and two-stage interval Newton) and ordinary techniques (exhaustive sampling and random sampling) was performed by running several examples.

Regarding the ordinary nonlinear systems, the verified algorithms are much slower than the algorithms that do not use this concept. However, the results of verified algorithms are more accurate.

But for the perturbed nonlinear systems, which are the main focus in this research, the verified algorithms, not only, *produce accurate results* but also have *faster responses* than other algorithms have. The difference in execution time becomes more obvious when the number of interval parameters increases. However, the interval algorithms are not always convergent and their convergence depends on the width of the interval parameters.

The results show that interval Newton and Hansen-Sengupta algorithms are nearly similar in performance and both of them are faster than the modified Krawczyk algorithm. Whilst, the new proposed algorithm-the two-stage interval Newton-shows the best performance from the execution time preview in most problems that are tested. The results generated from the two-stage interval Newton are equal in width with the results coming from the interval Newton.

The two-stage interval Newton method is proved to be the best choice for perturbed complex problems whose derivatives are dense matrices. On the other hand, interval Newton and Hansen-Sengupta are considered more suitable to the problems whose derivatives tend to be sparse.

6.2. A Closer View to the Interval Computing Software

We have implemented the interval Newton and Krawczyk algorithms using both INTLAB and C-XSC and reached to some concluding remarks. The execution time of verified algorithms varies with respect to the used tool. C-XSC and INTLAB present intervals as result. Both give enclosures of the exact result. INTLAB is based on BLAS, therefore it presents a good performance comparing with C-XSC. The performance presented by C-XSC is not so optimal because the algorithm uses special variables (data type dotprecision), which are simulated in software to achieve high accuracy. The results show that C-XSC has reliable results as INTLAB but slower response. The tests show that the method used in C-XSC is a good choice, but it should be optimized to gain performance.

6.3. Contributions and Future Work

This research makes the following contributions:

- We formally construct the two-stage interval Newton method for solving perturbed nonlinear systems and deduce its convergence analysis.
- Although the notion of solving perturbed nonlinear systems using interval arithmetic is already known, a few attempts are done in this field. Among the interval methods that are mentioned in this research, only the Hansen-Sengupta method was already known to have its parametric version. In this research, we use the interval Newton method to solve perturbed nonlinear systems and deduce the convergence analysis of the algorithm.
- Based on the idea that perturbed problems appear in many engineering applications, we provide a comparison between the interval methods and the methods which are currently used. We present many engineering examples that prove the efficiency of using the interval methods to perform the sensitivity analysis on perturbed problems.

The results obtained in this thesis give an insight into some further consequences and propose some directions for future research. We have already done some initial steps in some of those directions and we shall continue working on some of the following:

- Analog simulation focuses on the linear and non-linear behavior of a circuit over a continuous time or frequency interval. The circuit response is obtained by iteratively solving Kirchhoff's Laws for the circuit at time steps selected to ensure the solution has converged to a stable value and that numerical approximations of integrations are sufficiently accurate. One of the most time-consuming analyses is the sensitivity analysis in which the simulator calculates either the DC operating-point sensitivity or the AC small-signal sensitivity of an output variable with respect to all circuit variables, including model parameters. It calculates the difference in an output variable (either a node voltage or a branch current) by perturbing each parameter of each device independently. Since the method is a numerical approximation, the results are not verified and obtained after long time. The interval arithmetic if included with these simulators will solve the two problems: the unverified results and the time-consuming simulations.
- Currently all electronic simulators solve the nonlinear systems through a linearization process which again may result in unverified results. This problem may be handled by interval arithmetic as shown in this thesis. This

thesis show that solving perturbed nonlinear systems using interval arithmetic is not time-consuming and gives verified results.

References

1. Alefeld G., and Herzberger J., *Introduction to Interval Computations*, Academic Press, New York, 1983.
2. Allgower E. L., and Schmidt P. H., *An Algorithm for Piecewise-Linear Approximation of an Implicitly Defined Manifold*, SIAM J. Numer. Anal., 1985.
3. Brown G. W., *Monte Carlo Methods*, In E. F. Beckenbach, editor, Modern Mathematics for the Engineer. McGraw-Hill, New York, 1956.
4. Broyden C. G., *The convergence of a class of double-rank minimization algorithms*, Journal of the Institute of Mathematics and Its Applications 6: 76–90, 1970.
5. Corliss G. F., *Tutorial on Validated Scientific Computing Using Interval Analysis*, Second Scandinavian Workshop on Interval Methods and Their Applications, 2005.
6. Dawood H., *Theories of Interval Arithmetic: Mathematical Foundations and Applications*, LAP, 2011.
7. Deif A. S., *Sensitivity Analysis in Linear Systems*, Springer Verlag, Berlin, 1986.
8. Dinkel J., Tretter M., and Wong D., *Interval Newton Methods and Perturbed Problems*, Applied Mathematics and Computation, 28:211-222, 1988.
9. Gay D. M., *Solving Interval Linear Equations*, SIAM J. Numer. Anal., vol. 19, pp. 858-870, 1982.
10. Goldsztejn A., *A Branch and Prune Algorithm for the Approximation of Non-Linear AESolution Sets*, in Proc. of ACM SAC 2006, pp. 1650–1654.
11. Goldsztejn A., *Sensitivity Analysis Using a Fixed Point Interval Iteration*, hal-00339377, version 1, 17 Nov 2008.
12. Hansen E., and Greenberg R., *An Interval Newton Method*, Applied Mathematics and Computation, 12:89-98, 1983.
13. Hansen E., and Sengupta S., *Bounding Solutions of Systems of Equations Using Interval Analysis*, BIT 21, pp. 203-211, 1981.
14. Hansen E., and Walster G. W., *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 2003.
15. Hansen E., *A Multidimensional Interval Newton Method*, Reliable Computing, 12:253–272, 2006.
16. Hansen E., *Bounding the Solution of Interval Linear Equations*, SIAM J. Numer. Anal., vol. 29, pp. 1493-1503, 1992.
17. Hansen E., *Global Optimization Using Interval Analysis*, Marcel Decker, NY, 1992.
18. Hansen E., *On The Solution of Linear Algebraic Equations with Interval Coefficients*, Linear Algebra Appl., vol. 2, pp. 153-165, 1969.
19. Hassanein M. A., *Inverse Problems for Interval Matrices*,Ph.D. Thesis, Math. Physics Department., Fac. Eng., Cairo Univ., 1999.

20. Hofschuster W., Kramer W., Wedner S., and Wiethoff A., *C-XSC 2.0: A C++ Class Library for Extended Scientific Computing,* University of Wuppertal, 2001, pp. 1-24.

21. Hong H., and Stahl V., *Safe Starting Regions by Fixed Points and Tightening*, Computing, vol. 53, no. 3/4, pp. 323–335, Sep. 1994.

22. INTLAB. INTerval LABoratory, *http://www.ti3.tu-harburg.de/rump/intlab/.*

23. INTSOLVER,*http://www.mathworks.ch/matlabcentral/fileexchange/authors/38793.*

24. ISO/IEC 14882: *Standard for the C++ Programming Language,* 1998.

25. Jansson C., *Calculation of Exact Bounds for The Solution Set of Linear Interval Systems,* Linear Algebra Appl. , 251, pp. 321-340, 1997.

26. Kahan W. M., *A More Complete Interval Arithmetic: Lecture Notes for an Engineering Summer Course in Numerical Analysis at the University of Michigan,* Technical report, University of Michigan, 1968.

27. Kearfott R. B., and Walster G.W., *On Stopping Criteria in Verified Nonlinear Systems or Optimization Algorithms,* ACM Trans. Math. Software, 26(3):373–389, Sept. 2000.

28. Kearfott R. B., Dawande M., Du K., and Hu C., *Algorithm 787: INTLIB: a portable Fortran-77 interval standard function library,* ACM, Tran. On Math. Software, Vol. 20, No. 4, pp. 447-459, 1994.

29. Kearfott R. B., *Rigorous Global Search: Continuous Problems. Nonconvex Optimization and Its Applications 13,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

30. Klatte R., Kulisch U., Lawo C., Rauch M., and Wiethoff A., *C-XSC - A C++ Class Library for Scientific Computing,* Springer Verlag, Berlin, 1993.

31. Kolev L. V., *Interval Methods for Circuit Analysis,* World Scientific Publishing Co. Pte. Ltd., 1993.

32. Krawczyk R., *Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken,* Interner Bericht des Inst. für Informatik 68/6, Universität Karlsruhe, 1968. Computing, 4:187–201, 1969.

33. Miller I., and Freund J. E., *Probability and Statistics for Engineers,* Prentice-Hall, Englewood Cliffs, 1977.

34. Moore R. E., and Jones S. Z., *Safe Starting Regions for Iterative Methods,* SIAM J. Numer. Anal., 14, pp. 1051-1065, 1977.

35. Moore R. E., Kearfott R. B., and Cloud M. J., *Introduction to Interval Analysis,* SIAM, Philadelphia (2009).

36. Moore R. E., *A Computational Test for Convergence of Iterative Methods for Nonlinear Systems,* SIAM J. Numer. Anal., 15, pp. 1194-1196, 1978.

37. Moore R. E., *A Test for Existence of Solutions to Nonlinear Systems,* SIAM J. Numer. Anal., 14(4):611–615, 1977.

38. Moore R. E., *Interval Analysis,* Prentice-Hall, Englewood Cliffs, NJ, 1966.

39. Moore R. E., *Interval Arithmetic and Automatic Error Analysis in Digital Computing,* Ph.D. Thesis, Stanford University, 1962.

40. Moore R. E., *Methods and Applications of Interval Analysis,* SIAM, Stud. Appl. Math. 2, Philadelphia, 1979.

41. Moore R. E., *The Resolution of Close Minima,* Comput. Math. Appl., 25:57–58, 1993.

42. Neumaier A., *Interval Methods for Systems of Equations*, Encyclopedia Math Appl., Cambridge U.P., 1990.

43. Neumaier A., *Rigorous Sensitivity Analysis for Parameter-Dependent Systems of Equations*, Journal of Mathematical Analysis and Applications, 144, pp. 16–25, 1989.

44. Neumaier A., *Taylor Forms–Use and Limits*, http://www.mat.univie.ac.at/~neum , 2002.

45. Ning S., and Kearfott R. B., *A Comparison of Some Methods for Solving Linear Interval Equations,* SIAM J. Numer. Anal., vol. 34, pp. 1289-1305, 1997.

46. Oettli W., and Prager W., *Compatibility of Approximate Solution of Linear Equations with Given Error Bounds for Coefficients and Right Hand Side,* Numer. Math. , vol. 6, pp. 405-409, 1964.

47. Papoulis A., *Probability & Statistics,* Prentice-Hall, Englewood Cliffs, 1990.

48. Peter H., *a Lower bound for Range Enclosures in Interval Arithmetic*, Theoretical Computer Science, 279: 83-95, 1998.

49. Rall L. B., *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Computer Science 120. Springer-Verlag, Berlin, 1981.

50. References to INTLAB, *http://www.ti3.tu-harburg.de/rump/intlab/INTLABref.pdf.*

51. Rheinboldt W. C., *Numerical Analysis of Parametrized Nonlinear Equations*, Wiley, New York, 1986.

52. Rohn J., *Systems of Linear Interval Equations*, Linear Algebra Appl., vol. 126, pp. 39-78, 1989.

53. Rosenbrock H. H., *An automatic method for finding the greatest or least value of a function*, The Computer Journal 3: 175–184, 1960.

54. Rump S. M., *Fast and Parallel Interval Arithmetic*, BIT Numer. Math., 39(3), pp. 534–554, Sep. 1999.

55. Rump S. M., *Rigorous and Portable Standard Functions,* BIT Numer. Math., 41(3), pp. 540–562, 2001.

56. Schauder J., *Der Fixpunktsatz in Funktionalräumen*, Studia Math. 2, 171–180, 1930.

57. Shary S. P., *On Optimal Solution of Interval Linear Equations*, SIAM J. Numer. Anal., vol. 32, pp. 610-630, 1995.

58. Shayer S., *Interval Arithmetic with Some Applications for Digital Computers*, Technical Report Number LMSD5- 13-65-12, Lockheed General Research Program, 1965.

59. Urroz G. E., *Applications of Nonlinear Equations With SCILAB,* infoclearinghouse.com, 2001.

60. Van Hentenryck P., McAllester D., and Kapur D., *Solving Polynomial Systems Using a Branch and Prune Approach*, SIAM J. Numer. Anal., vol. 34, no. 2, pp. 797–827, Apr. 1997.

61. VERSOFT, *http://uivtx.cs.cas.cz/~rohn/matlab/*.

62. Wang Q., and Stengel R. F., *Probabilistic Control of Nonlinear Uncertain Systems*.

63. Wolfe M. A., *A modification of Krawczyk's algorithm*, SIAM J. Numer. Anal.,17(3) pp. 376-379, 1980.

Appendix A: Implementation of Interval Algorithms

A.1. Interval Gauss-Seidel

The interval Gauss-Seidel can be implemented easily with C-XSC in the following functions:

```
int Gauss_Seidel_image(ivector& X_kp1, bool& is_empty, bool&
error_occurred, imatrix& A, ivector& B, ivector& X_k) {

      int n = VecLen(X_k);
      int Error;
      rmatrix Y(n,n);
      MatInv(mid(A),Y,Error);
      if (Error)
            return -1;
      error_occurred = 0;
      is_empty = 0;
      int i=1;
      for (int index=1; index<=n; index++)
            X_kp1[index] = interval(0,0);
      interval new_x_i, second_new_x_i;
      bool there_are_2;
      interval num, denom;
      while((!error_occurred) & (!is_empty) & (i<= n)) {
            rvector Y_row(n);
            for (int index=1; index<=n; index++)
                  Y_row[index] = Y[i][index];
            gauss_seidel_step(new_x_i, second_new_x_i,
there_are_2, num, denom, i, Y_row, A, B, X_k);
            if(there_are_2)
                  error_occurred = 1;
            if (!error_occurred) {
                   is_empty = IsEmpty(new_x_i & X_k[i]);
                   if (!is_empty)
                        X_kp1[i] = (new_x_i & X_k[i]);
            }
            i=i+1;
      }
      return 0;
}
```

```
int gauss_seidel_step(interval& new_X_i, interval&
second_new_X_i, bool& there_are_2, interval& numerator,
interval& denominator, int i, rvector& Y_i, imatrix& A, ivector&
B, ivector& X) {
 //the result of applying a Gauss--Seidel step with variable i,
 //preconditioner matriX Y_i, and initial guess X.  The variable
 //there_are_2 is set to 1 if two semi-infinite intervals are
returned,
 //in which case second_new_X_i has the second interval;
there_are_2
 //is set to 0 and second_new_X_i is not set if there is only
 //one interval returned.

   int n = VecLen(X);
   real supnum;
   interval tmp1,tmp2;
   ivector G_i(n);    //1xn
   G_i = Y_i*A;
   interval C_i; //1x1
   C_i = interval(0,0);
   for (int index=1; index<=n; index++)
           C_i = C_i + Y_i[index]*B[index];
   numerator = C_i;
   new_X_i = X[i];
   second_new_X_i = X[i];
   if (n > 1) {
       if (i > 1)
              for (int index=1; index<i; index++)
               numerator = numerator - G_i[index]*X[index];
       if (i < n)
              for (int index=i+1; index<=n; index++)
               numerator = numerator - G_i[index]*X[index];
   }
   denominator = G_i[i];
   if (!((Inf(denominator)<=0) & (Sup(denominator)>=0))) {
       there_are_2 = 0;
       new_X_i = numerator / denominator;
   //elseif (~in(0,numerator))
   } else if (!((Inf(numerator)<=0) & (Sup(numerator)>=0))) {
       there_are_2 = 1;
        supnum = Sup(numerator);
       if(supnum < 0) {
           if (Sup(denominator)==0)
               tmp1 = interval(-Infinity,-Infinity);
           else
                   tmp1 = interval(supnum,supnum) /
interval(Sup(denominator),Sup(denominator));
           if (Inf(denominator) == 0)
               tmp2 = interval(Infinity,Infinity);
           else
               tmp2 = interval(supnum,supnum) /
interval(Inf(denominator),Inf(denominator));
           new_X_i = interval(-Infinity,Sup(tmp1));
           second_new_X_i = interval(Inf(tmp2),Infinity);
        } else {
           real infnum = Inf(numerator);
           if (Inf(denominator)==0)
               tmp1 = interval(-Infinity,-Infinity);
           else
               tmp1 = interval(infnum,infnum) /
interval(Inf(denominator),Inf(denominator));
           if (Sup(denominator) == 0)
               tmp2 = interval(Infinity,Infinity);
           else
               tmp2 = interval(infnum,infnum) /
interval(Sup(denominator),Sup(denominator));
           new_X_i = interval(-Infinity,Sup(tmp1));
           second_new_X_i = interval(Inf(tmp2),Infinity);
         }
   } else {
       there_are_2=0;
       new_X_i = interval(-Infinity,Infinity);
   }
   return 0;
}
```

Similarly, the interval Gauss-Seidel can be implemented easily with INTLAB in the following functions from [35]:

```
function [X_kp1,is_empty, error_occurred] =...
    Gauss_Seidel_image(A, B, X_k)
% X_kp1] = Gauss_Seidel_image(A,B,X_k) returns the image after a
% sweep of Gauss--Seidel iteration ( that is, (7.8) of the text)
% for the interval linear system A X = B, beginning with box
X_k,
% 1 <= i <= n.
% This is done using the inverse midpoint preconditioner.
% Upon return:
% if error_occurred = 1, then the computation could not proceed.
% (For example, the midpoint preconditioner may have been
% singular, or the denominator may have contained zero; the
% case of more than one box in the image is not handled
% with this routine.) Otherwise, error_occurred = 0.
% If error_occurred = 0 but is_empty = 1, this means that
% an intersection of a coordinate extent was empty.  In this
% case, there are no solutions to A X = B within X_k.
% If error_occurred = 0 and is_empty = 0, then the image under
% the Gauss--Seidel sweep is returned in X_kp1.

% Ralph Baker Kearfott, 2008/06/15 -- for the
% Moore / Kearfott / Cloud book.

n = length(B);
Y = inv(mid(A));
%Y=inv(mid(A)-rad(A)/2);
%Y=eye(size(A))

error_occurred = 0;
is_empty = 0;
i=1;
```

```
X_kp1 = midrad(zeros(n,1),0);
while(~error_occurred & ~is_empty & i<= n)
    [new_x_i, second_new_x_i, there_are_2, num, denom] ...
        = gauss_seidel_step(i, Y(i,:), A, B, X_k);
    if(there_are_2)
        error_occurred = 1;
    end
    if (~error_occurred)
%         is_empty = isempty_(intersect(new_x_i,X_k(i)));
%         if (~is_empty)
            X_kp1(i) = intersect(new_x_i, X_k(i));
%         end
    end
    i=i+1;
end



function [new_X_i, second_new_X_i, there_are_2,...
    numerator, denominator]...
    = gauss_seidel_step(i, Y_i, A, B, X)
% [new_X_i, second_new_X_i, there_are_2]...
%  = gauss_seidel_step(i, Y_i, A, B, X) returns
% the result of applying a Gauss--Seidel step with variable i,
% preconditioner matriX Y_i, and initial guess X.  The variable
% there_are_2 is set to 1 if two semi-infinite intervals are
returned,
% in which case second_new_X_i has the second interval;
there_are_2
% is set to 0 and second_new_X_i is not set if there is only
% one interval returned.

% Ralph Baker Kearfott, 2008/06/15 -- for the
% Moore / Kearfott / Cloud book.

   n = size(A,2);
   G_i = Y_i*A;
   C_i = Y_i*B;
   numerator = C_i;
   new_X_i = X(i);
   second_new_X_i = X(i);
   if (n > 1)
       if (i > 1)
           numerator = numerator - G_i(1:i-1)*X(1:i-1);
       end
       if (i < n)
          numerator = numerator - G_i(i+1:n)*X(i+1:n);
       end
   end
   denominator = G_i(i);
   numerator;
   denominator;
   if (~in(0,denominator))
       there_are_2 = 0;
       new_X_i = numerator / denominator;
   elseif (~in(0,numerator))
       there_are_2 = 1;
       supnum = sup(numerator);
       if(supnum < 0)
           if sup(denominator)==0
               tmp1 = infsup(-Inf,-Inf);
           else
               tmp1 = midrad(supnum,0) /
midrad(sup(denominator),0);
           end
           if inf(denominator) == 0
               tmp2 = infsup(Inf,Inf);
           else
               tmp2 = midrad(supnum,0) /
midrad(inf(denominator),0);
           end
           new_X_i = infsup(-Inf,sup(tmp1));
           second_new_X_i = infsup(inf(tmp2),inf);
       else
           infnum = inf(numerator);
           if inf(denominator)==0
               tmp1 = infsup(-Inf,-Inf)
           else
               tmp1 = midrad(infnum,0) /
midrad(inf(denominator),0);
           end
           if sup(denominator) == 0
               tmp2 = infsup(Inf,Inf)
           else
               tmp2 = midrad(infnum,0) /
midrad(sup(denominator),0);
           end
           new_X_i = infsup(-Inf,sup(tmp1));
           second_new_X_i = infsup(inf(tmp2),Inf);
       end
   else
       there_are_2=0;
       new_X_i = infsup(-Inf,Inf);
   end
end
```

A.2. Interval Newton Method for Perturbed Nonlinear Systems

The interval Newton method for perturbed nonlinear systems can be implemented easily with C-XSC in the following function:

```
int interval_newton_step(imatrix& G, ivector& v, ivector& i,
double& TimeElapsed, bool& is_empty, bool& Error) {
      //    G.v=i
      //    interval_newton_step does one step of interval Newton
algorithm
      //    Pass the initial vector v to the function and it will
return the final result also in v
      int size = VecLen(v);
      ivector v1(size);
      Error = false;

      clock_t launch = clock();
      Gauss_Seidel_image(v1, is_empty, Error, G, i, v);
      clock_t done = clock();

      TimeElapsed = ((double) (done - launch)) / CLOCKS_PER_SEC;

      v = v1;
      return 0;
}
```

Similarly, the interval Newton method for perturbed nonlinear systems can be implemented easily with INTLAB in the following function:

```
function [NX_intersect_X, TimeElapsed, NoIterations, Error] =
i_newton_mod(X,f,MaxIterations)
% Interval Newton using Gauss-Seidel
% roundoff error --
%format long;
NoIterations=0;
Error = false;
n = length(X);
NX_intersect_X = X;
tic
while(NoIterations<MaxIterations)
    NoIterations=NoIterations+1;
    y = mid(NX_intersect_X);
    iy = midrad(y,0);
    fy = feval(f,iy);

    % Now compute F'(X) and the preconditioning matrix Y --
    Xg = gradientinit(NX_intersect_X);
    FXg = feval(f,Xg);

    % Compute the initial V --
    V = NX_intersect_X-y;
    % Now, do the Gauss--Seidel sweep to find V --
    [new_V,is_empty,error_occurred] = Gauss_Seidel_image(FXg.dx,
-fy, V);

    NX = y+new_V;
    if(intersect(NX,NX_intersect_X)==NX_intersect_X)
        break;
    end
    NX_intersect_X = intersect(NX,NX_intersect_X);
%     is_empty = isempty_(NX_intersect_X);
%     if (is_empty)
%         break;
%     end
    if (any(isnan(NX_intersect_X)))
        Error = true;
        break;
    end;
end
TimeElapsed=toc;
```

A.3. Modified Krawczyk Method for Perturbed Nonlinear Systems

The Modified Krawczyk method for perturbed nonlinear systems can be implemented easily with C-XSC in the following function (this function also includes implementation of some test problems):

```
double ModKrawczyk(int fDim,int noIntervalVars,int p,int
problem,double delta){
      //nmax : max number of iterations
      const int  nmax = 100;
      //X  : solution vector at each iteration
      //Xg : version of X to be used in automatic
differentiation
      //fXg: f(Xg)
      //fx : f(Xg)=f(X)
      //Jfx: J(Xg)=J(X)
      //iy : mid(X)
      //fy : f(iy)
      //KX : Krawczyk operator at each iteration
      //Y  : inv(mid(Jfx))
      ivector fy(fDim), iy(fDim), fx(fDim), KX(fDim), X(fDim);
      imatrix Jfx(fDim,fDim);
      rmatrix Y(fDim,fDim);
      GTvector Xg(fDim), fXg(fDim);
      //rvector y;
      int n;
      int Error = 0;
      rvector eps(fDim);
      for (int i=1;i<=fDim;i++)
            eps[i] = 0.0001;
      cout << SetPrecision(23,15) << Scientific;
      cxsc::real temp1 = 5.0/fDim;
      cxsc::real temp2 = 5.0*fDim;
      //The exact solution of each problem (Hansen, Banana,
Mancino)
      if (problem==1)
      {
            //Result for this problem is not correct
            temp1 =
4.0*sqrt(temp1)/7.0*sinh(1.0/3.0*asinh(7.0*sqrt(temp2)/2.0));
      //Hansen
            //set the initial box
            for (int i=1; i<=fDim; i++) {
                  X[i] = interval(temp1*(1-
delta),temp1*(1+delta));
            }
      }
      else if (problem==2)
      {
            temp1=1;   //Banana
            //set the initial box
            for (int i=1; i<=fDim; i++) {
                X[i] = interval(temp1*(1-
delta),temp1*(1+delta));
            }
        }
        else if (problem==3)
        {
            //Mancino
            if (fDim!=4) return -1;
            //set the initial box
            X[1] = interval(1-delta,1+delta)*1.896515;
            X[2] = interval(1-delta,1+delta)*(-0.210191);
            X[3] = interval(1-delta,1+delta)*0.542070;
            X[4] = interval(1-delta,1+delta)*(-0.023893);
        }
        else
            return -1;
        cout << endl << X[1];
        cout << endl << X[2];
        cout << endl << X[3];
        //n : number of iterations
        n = 0;
        clock_t launch = clock();
        iy = mid(X);
        Xg = GradVar(X);
        //Substitute with Xg in the problem
        if (problem==1)
            fXg = f3(Xg,fDim,noIntervalVars);
        else if (problem==2)
            fXg = banana(Xg,fDim,noIntervalVars);
        else if (problem==3)
            fXg = Mancino(Xg,fDim,noIntervalVars);
        fx = fValue(fXg);           // function value
        Jfx = JacValue(fXg);  // jacobian value
        //Inversion of the mid of the jacobian
        MatInv(mid(Jfx),Y,Error);

        /*mat MatTemp(fDim,fDim), MatTempInv(fDim,fDim);
        for(int i1=0; i1<fDim; i1++)
            for(int i2=0; i2<fDim; i2++)
                MatTemp(i1,i2)=_double(mid(Jfx)[i1][i2]);
        MatTempInv = inv(MatTemp);*/
        /*double *islam;
        islam = (double*)malloc(fDim*fDim*sizeof(double));
        matrix_inverse(mid(Jfx),islam,fDim);*/
        /*for (int i=0; i<fDim; i++)
        {     for (int j=0; j<fDim; j++)
            {     Y[i][j] =  _real(MatTemp(i,j));
                cout << Y[i][j] << " ";
            }
            cout << endl;
        }
        */
        ivector temp;
        rmatrix eye(fDim,fDim);
        I(eye,fDim);
        bool flag = true;
```

```
      //The main (outer) loop (till convergence or max no. of
iterations)
      do {
            n++;
            //cout << n << endl;
            //The inner loop (p times)
            for(int i=1;i<=p;i++) {
                  iy = mid(X);
                  if (problem==1)
                        fy = f3(iy,fDim,noIntervalVars);
                  else if (problem==2)
                        fy = banana(iy,fDim,noIntervalVars);
                  else if (problem==3)
                        fy = Mancino(iy,fDim,noIntervalVars);
                  if (!Error) {
                        //The main operation (Krawczyk operator)
                        KX = iy - Y*fy + (eye - Y*Jfx)*(X - iy);
                        //Terminate if KX is enclosed in X
                        if ((X & KX) == X)
                        {flag = false; break;}
                        // itersection of X and KX
                        //Update X
                        X = X & KX;
                  }
                  else
                        cout << MatInvErrMsg(Error) << endl;
            }
            //Update Xg
            Xg = GradVar(X);
            //Update fXg
            if (problem==1)
                  fXg = f3(Xg,fDim,noIntervalVars);
            else if (problem==2)
                  fXg = banana(Xg,fDim,noIntervalVars);
            else if (problem==3)
                  fXg = Mancino(Xg,fDim,noIntervalVars);
            fx = fValue(fXg);           // function value
            Jfx = JacValue(fXg);  // jacobian value
            MatInv(mid(Jfx),Y,Error);

            //for(int i1=0; i1<fDim; i1++)
            //    for(int i2=0; i2<fDim; i2++)
            //          MatTemp(i1,i2)=_double(mid(Jfx)[i1][i2]);
            //MatTempInv = inv(MatTemp);
            //for (int i=0; i<fDim; i++)
            //{   for (int j=0; j<fDim; j++)
            //    {     Y[i][j] =  _real(MatTemp(i,j));
            ////        cout << Y[i][j] << " ";
            //    }
            //    //cout << endl;
            //}
            /*if (diam(X[1]) > eps[1])
                  cout << "eps";
            if (!Error)
                  cout << "!Error";
            if (n < nmax)
```

```
                cout << "nmax";
           if (flag)
                cout << "flag";*/
           //matrix_inverse(mid(Jfx),Y,fDim);
      } while ((diam(X[1]) > eps[1]) && (!Error) && (n < nmax)
&& flag);
      clock_t done = clock();
      cout << endl << X[1];
      cout << endl << X[2];
      cout << endl << X[3];
      //cout << endl << X[4];
      //cout << endl << X[5];
      //cout << endl << X[6];
      cout << endl << X[fDim];
      cout << "\nNumber of iterations : " << n;
      double diff = ((double) (done - launch)) / CLOCKS_PER_SEC;

      return diff;
}
```

Similarly, the Krawczyk method for perturbed nonlinear systems can be implemented easily with INTLAB as follows:

```
function [X,TimeElapsed,NoIterations,Error] =
Krawczyk(X0,f,max_no_iter,no_inner_iter,parameters)

X=X0;
x=mid(X);
n=length(X);
format long,
Error = false;
flag=true;
NoIterations=0;
tic

Xg = gradientinit(X);
FXg = feval(f,Xg,parameters);
A = inv(mid(FXg.dx));
r=norm((eye(n,n) - A*FXg.dx),inf);

        ix = midrad(x,0);
        fx = feval(f,ix,parameters);
        Y = x - A*fx + (eye(n,n) - A*FXg.dx) * (X - x);
        Z=intersect(X,Y);
        if (ISNAN(Z(1,1)) | Z==X)
            TimeElapsed = -1;
            return;
        end;

while(flag & NoIterations<max_no_iter)
    NoIterations=NoIterations+1;
    for i=1:no_inner_iter
        if NoIterations==1
            ix = midrad(x,0);
            fx = feval(f,ix,parameters);
            Y = x - A*fx + (eye(n,n) - A*FXg.dx) * (X - x);
            Z=intersect(X,Y);
            if (any(ISNAN(Z)))
                Error = true;
                flag=false;
                break;
            end;
            if(X==Z)
                flag=false;
                break;
            end;
        end;
        X=Z;
        x=mid(X);
    end
    Xg = gradientinit(X);
    FXg = feval(f,Xg,parameters);
    B = inv(mid(FXg.dx));
    s=norm((eye(n,n) - B*FXg.dx),inf);
    if s<=r
        A=B;
        r=s;
    else
        r=norm((eye(n,n) - A*FXg.dx),inf);
    end
end

TimeElapsed=toc;
```

A.4. Hasnen-Sengupta Method for Perturbed Nonlinear Systems

The Hansen-Sengupta method for perturbed nonlinear systems can be implemented easily with INTLAB in the following function:

```
function
[X,TimeElapsed,NoIterations,Error]=HansenSengupta(X0,f,max_no_it
er,A)
X=X0;
Error = false;
n=length(X);
format long,
% NoIterations=0;
tic
for NoIterations=1:max_no_iter
    mid_x=mid(X);
    Xg = gradientinit(X);
    FXg = feval(f,Xg,A);
    Ag = gradientinit(A);
    FAg =feval(f,mid_x,Ag);
    C = inv(mid(FXg.dx));

    Y = mid_x + Gauss_Seidel_image((C*FXg.dx),(-
C*feval(f,mid_x,mid(A))-C*FAg.dx*(A-mid(A))),X-mid_x);

    Z=intersect(X,Y);
    if (any(ISNAN(Z)))
        Error = true;
        break;
    end;
    if(X==Z)
        break;
    end;
    X = Y;

end

TimeElapsed=toc;
```

A.5. Two-Stage Interval Newton Method for Perturbed Nonlinear Systems

The two-stage interval Newton method for perturbed nonlinear systems can be implemented easily with INTLAB in the following function:

```
function [NX_intersect_X, TimeElapsed, NoIterations, Error] =
two_stage_i_newton(X,f,MaxIterations)
%Two-stage interval Newton
%format long;
NoIterations=0;
Error = false;
n = length(X);
NX_intersect_X = X;
tic
while(NoIterations<MaxIterations)
    NoIterations=NoIterations+1;
    y = mid(NX_intersect_X);
    iy = midrad(y,0);
    fy = feval(f,iy);

    % Now compute F'(X) and the preconditioning matrix Y --
    Xg = gradientinit(NX_intersect_X);
    FXg = feval(f,Xg);

    % Compute the initial V --
    V = NX_intersect_X-y;
    % Now, do the Gauss--Seidel sweep to find V --
    [new_V,is_empty,error_occurred] = Gauss_Seidel_image(FXg.dx, -fy,
V);

    NX = y+new_V;
    if(intersect(NX,NX_intersect_X)==NX_intersect_X)
        break;
    end
    NX_intersect_X = intersect(NX,NX_intersect_X);

%Stage 2
    % Compute the initial V --
    V = NX_intersect_X-mid(NX_intersect_X);
    % Now, do the Gauss--Seidel sweep to find V --
    [new_V,is_empty,error_occurred] = Gauss_Seidel_image(FXg.dx, -
feval(f,mid(NX_intersect_X)), V);

    S = mid(NX_intersect_X)+new_V;
    if(intersect(S,NX_intersect_X)==NX_intersect_X)
        break;
    end
    NX_intersect_X = intersect(S,NX_intersect_X);

    if (any(isnan(NX_intersect_X)))
        Error = true;
        break;
    end;
end
TimeElapsed=toc;
```

A.6. Division in Extended Arithmetic

The interval arithmetic in INTLAB does not implement sharp extended arithmetic; division A/B by any interval B that contains zero results in the interval $[-\infty, \infty]$. This is an enclosure of the exact range of the operation but is in general not *sharp*, since it contains many values that are not obtainable as a/b for $a \in A$ and $b \in B$. However, we may use the following function from [35]:

```
function [Y1,Y2,two] = xreciprocal(X)
% [Y1,Y2,two] = xreciprocal(X) returns the extended reciprocal
% of X defined by the three cases. The return value two is set  % to 0
if only one interval is returned and is set to 1
% if two intervals are returned.
% If X does not contain zero, the result of ordinary
% interval division is returned in Y1, and two is set to 0.
% In the case inf(X) = sup(X) = 0, avoided in the text,
% two is set to 1, and two empty intervals are returned.
% (INTLAB represents an empty interval as infsup(NaN,NaN) )
% In cases where there is only one interval, Y2 is set
% to INTLAB's representation of the empty interval.
if (inf(X) > 0) | (sup(X) < 0) % do ordinary interval division
two=0;
Y1 = 1/X;
Y2 = infsup(NaN,NaN);
elseif (inf(X)==0) & (sup(X) > 0) % Case 1 of the text --
two=0;
lower_bound = infsup(1,1) / infsup(sup(X),sup(X));
Y1 = infsup(inf(lower_bound),Inf);
Y2 = infsup(NaN,NaN);
elseif (inf(X)<0) & (sup(X) > 0) % Case 2 of the text --
two=1;
upper_bound = infsup(1,1) / infsup(inf(X),inf(X));
Y1=infsup(-Inf,sup(upper_bound));
lower_bound = infsup(1,1) / infsup(sup(X),sup(X));
Y2 = infsup(inf(lower_bound),Inf);
elseif (inf(X) < 0) & (sup(X) == 0) % Case 3 of the text --
two = 0;
upper_bound = infsup(1,1) / infsup(inf(X),inf(X));
Y1=infsup(-Inf,sup(upper_bound));
Y2 = infsup(NaN,NaN);
else % This is the case where X=0, not covered in the text --
two =1;
Y1 = infsup(NaN,NaN);
Y2 = infsup(NaN,NaN);
end
```

الملخص

هذا البحث العلمي يهتم بدراسة حلول المعادلات غير الخطية و التي تحتوي على بيانات غير دقيقة–والتي يُرمز لها بالمعادلات غير الخطية المضطربة–باستخدام طرق تعتمد على حساب الفترات. و تظهر مثل هذه المسائل المضطربة في العديد من التطبيقات الهندسية لدراسة مدى حساسية عوامل التصميم للتغيرات الناتجة إما أثناء التصنيع أو أثناء أداء الحسابات باستخدام الفاصلة العائمة. إن الإطار الأساسي لهذا البحث هو تقديم طرق شبيهة بطريقة نيوتن لحساب الفترات لحل المعادلات غير الخطية المضطربة عن طريق تعديل الطرق المستخدمة حاليا في حل المعادلات غير الخطية التقليدية لتصبح مناسبة لحل المعادلات غير الخطية المضطربة. يُقدم و يُستنتج في هذا البحث تحليل تواجد حل للمعادلات غير الخطية المضطربة و التقارب إلى هذا الحل.

في سبيل تحقيق هذه الأهداف يتم تقديم مقدمة مختصرة عن حساب الفترات و حلول المعادلات الخطية التي تحتوي على فترات و كذلك حلول المعادلات غير الخطية التقليدية باستخدام حساب الفترات. هناك طرق مشهورة لحل المعادلات غير الخطية يتم تقديمها مثل: نيوتن للفترات و هانسن–سينجوبتا و كراوزيك. و يتم تقديم نسخة مناسبة لحل المعادلات المضطربة من كل طريقة. و كذا يتم تقديم المشاكل المتعلقة بكل طريقة على حدى و دراسة تقاربها. علاوة على ذلك فإنه يتم استنتاج طريقة معدلة لطريقة نيوتن للفترات و التي تُعرف بطريقة نيوتن ذات المرحلتين. و تمتلك الطريقة ذات المرحلتين ميزة تقليل الوقت اللازم لايجاد حل للمعادلات. و يتم اختبار الطرق التي تعتمد على حساب الفترات لاثبات كفاءتها باستخدام مسائل من مصادر علمية سابقة و لكن مع إضافة بعض التغيرات. و لقد لاحظنا أن اتساع حل المعادلات المضطربة يعتمد على مدى اتساع هذه الآضطرابات. و قد تمت مقارنة طريقة نيوتن ذات المرحلتين مع باقي الطرق المذكورة سابقاً. و لقد أثبتت الطريقة تحسن في الوقت المستهلك. و تمت أيضاً مقارنة الحلول الناتجة من الطرق التي تعتمد على حساب الفترات مغ تلك الناتجة من طريقة مونت كارلو. و أثبتت المقارنة تفوق الطرق التي تعتمد على حساب الفترات من ناحية الوقت المستهلك و الدقة المطلوبة لايجاد الحل المنشود.

مهندس:	إسلام رفعت كامل طه
تاريخ الميلاد:	1988\2\1
الجنسية:	مصري
تاريخ التسجيل:	....\....\.........
تاريخ المنح:	....\....\.........
القسم:	الرياضيات و الفيزيقا الهندسية
الدرجة:	ماجستير

ضع صورتك هنا

المشرفون:

ا.م.د. مها أمين حسنين

ا.م.د. حسام علي حسن فهمي

الممتحنون:

أ.د................... (الممتحن الخارجي)

أ.د................... (الممتحن الداخلي)

ا.م.د. مها أمين حسنين (المشرف الرئيسي)

ا.م.د. حسام علي حسن فهمي (عضو)

عنوان الرسالة:

حل المعادلات المضطربة غير الخطية متعددة الأبعاد باستخدام حساب الفترات

الكلمات الدالة:

حساب الفترات، المسائل المضطربة، الأنظمة غير الخطية، طريقة نيوتن للفترات

ملخص الرسالة:

....

....

....

....

....

....

....

حل المعادلات المضطربة غير الخطية متعددة الأبعاد باستخدام حساب الفترات

اعداد

إسلام رفعت كامل طه

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة الماجستير
في
الرياضيات الهندسية

يعتمد من لجنة الممتحنين:

الاستاذ الدكتور:	الممتحن الخارجي
الاستاذ الدكتور:	الممتحن الداخلي
الاستاذ المساعد الدكتور: مها أمين حسنين	المشرف الرئيسى
الاستاذ المساعد الدكتور: حسام علي حسن فهمي	عضو

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـر العربيــة

يونيو – 2013

حل المعادلات المضطربة غير الخطية متعددة الأبعاد باستخدام حساب الفترات

اعداد
إسلام رفعت كامل طه

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة الماجستير
في
الرياضيات الهندسية

تحت اشراف

الاستاذ المساعد الدكتور
مها أمين حسنين
قسم الرياضيات و الفيزيقا الهندسية
كلية الهندسة – جامعة القاهرة

الاستاذ المساعد الدكتور
حسام علي حسن فهمي
قسم الالكترونيات والاتصالات الهندسية
كلية الهندسة – جامعة القاهرة

كليـة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـر العربيــة

يونيو – 2013

حل المعادلات المضطربة غير الخطية متعددة الأبعاد باستخدام حساب الفترات

اعداد

إسلام رفعت كامل طه

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة الماجستير
في
الرياضيات الهندسية

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـر العربيــة

يونيو – 2013