Optimization Using Interval Analysis

Ishan Goswami

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

भारतीय प्रौद्योगिकी संस्थान हैदराबाद

Indian Institute of Technology Hyderabad

Department of Electrical Engineering

June 2018

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Ishan Goswami

(Signature)

Ishan Goswami

(Ishan Goswami)

BB18MTECH11034

(Roll No.)

Approval Sheet

This Thesis entitled Optimization Using Interval Analysis by Ishan Goswami is approved for the degree of Master of Technology from IIT Hyderabad

M. Vidyasagar

(Prof. M Vidyasagar) Examiner
Dept. of Electrical Eng
IITH

(Dr. Sumohana S. Channappayya) Examiner
Dept. of Electrical Eng
IITH

(Dr. Ketan P. Detroja) Adviser
Dept. of Electrical Eng
IITH

Kishalay Mitra

(Dr. Kishalay Mitra) Co-Adviser
Dept. of Chemical Eng
IITH

(Dr. Phanindra Jampana) Chairman
Dept. of Chemical Eng
IITH

Abstract

Optimization is a procedure where we find and compare feasible solutions to a problem until the best solution is found, or in other words until no better solution can be found. Most of the real world problems are multi-objective, i.e. they have more than one conflicting objectives. Due to the presence of multiple conflicting objectives, it becomes difficult to find one best solution among multiple solutions available.

Classical optimization can find at best one solution in one trial. Evolutionary algorithms (EAs), on the other hand, can find multiple solutions in one run. Hence, EAs are best suited to find solutions to multi-objective problems.

Interval mathematics is a generalization of mathematics, in which real numbers are replaced by interval numbers. Interval analysis was developed so as to put bounds on errors, both rounding and measurement, and thus, developing numerical methods to yield better results. In interval analysis we represent a real number x as an interval instead of exact value. The arithmetic is almost similar to real arithmetic, but since it involves intervals, we have to take care of the bounds of all the intervals involved. We begin with the introduction to some of the basic commands and functions that are used in INTLAB, the toolbox that will be used to perform interval analysis. This toolbox is compatible with MATLAB 2016a (the MATLAB version that I have used for this version). The basic commands that are used in MATLAB can also be used in INTLAB. This text explains in fair detail the interval number system. The interval numbers follow the properties of real numbers, although we have to make slight modifications accordingly. When we go through these properties, we make some interesting observations regarding some properties that make interval analysis different, and should be kept in mind while performing operations. Real functions can be extended to their interval counterpart very easily.

Now, the question arises, whether or not, can the interval analysis be applied to solve optimization problems? If yes, then how do we do that? In this text, the different processes involved in performing optimization using interval analysis, which form the prerequisite for the algorithm, are explained. After that, the complete algorithm, called the *Branch and Bound Technique* is explained. Having understood the basics of interval analysis, now the question arises, how do we perform interval analysis on computer? To explain this, some examples are taken, their outputs show the working of interval analysis. Later, some examples are taken for optimization problems and their solutions are produces by the aforementioned technique.

Contents

Chapter 1

Introduction

1.1 Multi-objective Optimization

According to [1], optimization refers to finding one or more feasible solutions corresponding to extreme values of one or more objectives.

The task of finding optimal solution(s) for a physical system with only one objective function is called *single objective optimization*. These problems can be classified into two categories - Evolutionary Algorithms and Simulated Annealing.

The task of finding optimal solution(s) for a physical system with one or more objective functions is called *multi-objective optimization*.

Let's take an example of buying a car, to understand the optimization problem better. We know cars prices range from a few lakhs to a few crores of rupees (depending on the brand and type). Let us take two extreme hypothetical examples of cars (Fig. 1.1, taken from [1]), i.e. car A costing about Rs.10,000 and car B costing around Rs.100,000. Let's say that car A has comfort level of 40% and car B has comfort level of 90%. Here the two objectives are cost and comfort. If cost is the only objective, then without a doubt everyone will buy car A. Similarly, if the comfort is the only objective, then everybody will go for car B. Now, apart from these two solutions, there exist many other solutions as per the needs and capabilities of different people. Thus, among all these solutions, we see that the solution is better only in one of the objectives.

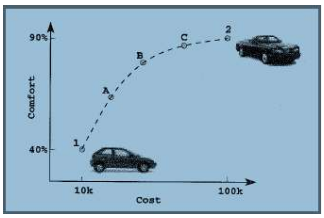

Figure 1.1: Example for Multi-objective optimization

The above problem can be solved by considering either only one of the two objectives individually, or by considering both the objectives simultaneously. The former will be the case for single objective optimization, while the latter will be a problem for multi-objective optimization. As we have seen above, if we solve the problem with single objective, then all we will see around us will be only two

types of cars i.e. car A and car B. But in reality we see that is not the case. And that is why we can say that real world problems mostly are multi-objective in nature.

When we try to solve the problem as multi-objective problem, we see that whenever we optimize one objective, other objectives will be affected for the worse. Thus, we see that in multi-objective optimization problems, if we optimize one objective then we have to compromise on the other objective(s). This is the basic crux of multi-objective optimization problems.

In earlier stages of development of multi-objective optimization problems, people used to solve them by considering them as a cluster of several single objective problems. But over the years it became clear that this was wrong approach.

1.1.1 Approach to solve Multi-objective Optimization

Although a multi-objective problem gives many optimal solutions, as a user we need only one optimal solution, no matter what. Hence, it is needed that after solving the problem and getting many optimal solutions, we use some higher information to get a unique optimal solution. Taking the example of car buying problem once again, we see that apart from the two objectives mentioned above i.e. cost and comfort, there are a number of other objectives that need to be addressed viz. social status, choice of brand, availability of brand, locality, etc. Hence to solve a multi-objective problem, the basic need is to find a trade-off solution by considering all the objectives to be equally important. Hence to find solutions to *ideal multi-objective problem*, we suggest the following principle:

Step 1 Find multiple trade-off solutions for the objectives over a wide range.

Step 2 Use higher level qualitative information to find a single optimal solution.

Hence we see that the classical way of finding optimal solutions follows a procedure based approach.

1.1.2 Evolutionary Algorithms

It was seen in the previous section that solving a multi-objective problem using classical approach requires us to follow a procedure based approach. In classical method, we find many solutions simultaneously. The classical approach was designed keeping in mind the limitations of computing methods at the time which could find only one solution at a time. With the advent of time and advancement of computing facilities it is only justified that the optimization techniques are also improved. Hence, Evolutionary Algorithms (EAs) were designed. EAs mimic the the evolutionary principles of nature. One of the most striking differences of EAs from classical algorithms is that here we have a population of solutions instead of a single solution. Hence, we also get a population of solutions in each iteration. If a problem has a single optimal solution, then we get a single optimal solution by EAs. But, if a problem has multiple optimal solutions, then EAs will give multiple solutions.

1.2 Interval Mathematics

Ever since elementary school, we have been doing mathematics. Our primary objective was always to find out the exact value of the solution. Now, if we are given the exact value of all the quantities, it is alright to perform mathematics in that way. But with the advancement of computers, the

mathematics also evolved to a more complex state. But since computers are also machines and have limitations in the number of digits we can represent, it is difficult to represent certain numbers with accuracy, viz. the irrational numbers.

For example [2], let's take a quadratic equation $x^2 - 2 = 0$. Now we know that the solution is $x = \pm\sqrt{2}$. Now we know that $\sqrt{2}$ is an irrational number, and cannot be represented with high accuracy. Hence, it is difficult to represent these numbers.

Apart form that, mathematical computing errors occur in many ways. Data often contain measurement errors, or otherwise are uncertain due to rounding errors and approximations. Many situations arise where we deal with quantities that are not exactly representable.

The interval analysis was developed in the late 1950s by mathematicians. The purpose of interval analysis is to contain the effect of these errors with lower and upper bounds. As for the example above, it will be way better if we represent the solution as an interval like $x = [1.414, 1.415]$ and $x = [-1.415, -1.414]$. This way we don't have to take care of the number of significant digits as the answer will always lie within the interval.

Interval mathematics is a generalization of mathematics where real numbers are replaced by interval numbers, real arithmetic replaced by interval arithmetic, and real analysis replaced by interval analysis [3].

It is quite alright to represent a quantity in terms of interval, however loose the solution may seem. Since it might be better to know that $y \in [59, 62]$ rigorously than knowing the answer as $y \approx 60$.

1.3 INTLAB Commands and Functions

INTLAB [4], [5] stands for INTerval LABoratory. It is a toolbox that was developed by S.M.Rump so as to perform interval analysis. This version of INTLAB is entirely written in MATLAB and is compatible with MATLAB 2016a, with the exception of some functions that don't work. Here we mention some of the basic INTLAB commands and functions that will be used later to perform some basic operations. Although there are a lot more of these commands and functions that INTLAB incorporates, here only a few are mentioned.

1.3.1 Basic Commands

Let X denote an interval, and a, b, c, and d be real numbers.

- *intvalinit('DisplayInfsup')*: Infimum-Supremum default display

  $X = infsup(a, b)$: Defines X as interval $[a, b]$

- *intvalinit('DisplayMidrad')*: Midpoint radius default display

  $X = midrad(a, b)$: Defines X as $[c, d] = [a - b, a + b]$

- *intvalinit('Display_')*: Uncertainty default display

- INTLAB follows all the basic arithmetic and logical instructions as followed by MATLAB, with interval interpretations

Throughout this work, $X = infsup(a, b)$ has been used.

1.3.2 Special Interval Functions

These are special to interval analysis.

- *diam(X)*: Width of X
- *hull(X,Y)*: Interval hull of X and Y
- *inf(X)*: Returns the lower bound of X
- *sup(X)*: Returns the upper bound of X
- *intersect(X,Y)*: Gives intersection of X and Y
- *union(X,Y)*: Gives union of X and Y
- *mid(X)*: Gives midpoint of X

1.3.3 Plotting in INTLAB

- *plotintval(X)*: If X is an n-dimensional complex interval vector, this plots each component as a circle in complex plane. If X is array of real interval vectors of dimension $(2, N)$, then this plots each interval vector $[X_{1,i}, X_{2,i}]$ as a box in xy-plane.
- *plotlinsol(A,B)*: This plots the solution to the system of linear equation $AX = B$. The entire solution is enclosed in an interval hull.

1.4 Motivation

The motivation for this project comes from the way interval optimization works. It works on the principle of rejection, i.e. it has the ability to reject a particular region that does not fit the required criteria. Hence, when performing optimization we can remove large chunks of search area and hence significantly reduce the region.

1.5 Objectives

The objective of this study is to be able to apply interval mathematics to optimization problems, both single objective and multi-objective.

Chapter 2

Multi-Objective Interval Optimization

From [1] we see that, when we solve a physical system with more than one objective to find it's optimal solution, it is called multi-objective optimization. Almost all the real world problems are multi-objective in nature since they require us to evaluate a problem from more than one perspective.

As we have seen before, for a problem to be multi-objective, all the objectives need to be conflicting in nature. By conflicting we mean that if we try to optimize the problem in one objective, then we won't get optimal solution in the other objective. This is the reason that solving a multi-objective optimization problem is not an easy task.

2.1 Multi-objective Optimization Problem(MOOP)

A problem that deals with multiple objectives is called a Multi-objective optimization problem (MOOP. A MOOP has a number of objective functions which need to be maximized or minimized. The problem usually has a number of constraints which any feasible solution (including the optimal solution) must satisfy. The general form of MOOP can be written as follows:

$$F(x) = (f_1(x), f_2(x), ..., f_m(x))^T \tag{2.1}$$

$$subject\ to \begin{cases} h_i(x) \geq 0\ ,\ i = 1, 2, ..., I \\ g_j(x) = 0\ ,\ j = 1, 2, ..., J \\ x_i^{(L)} \leq x_i \leq x_i^{(U)}, i = 1, 2, ..., N \end{cases}$$

A solution $\mathbf{x}$ is the solution of n variables: $x = (x_1, x_2, ..., x_n)^T$.

Also associated with the problem are I inequality and J equality constraints. The terms $h_i(x)$ and $g_j(x)$ are called constraint functions. A solution that does not satisfy all the $(I + J)$ constraints and all of the $2N$ variable bounds, is called *infeasible solution.* On the other hand,a solution $\mathbf{x}$ satisfying all constraints and variable bounds, is known as *feasible solution.*

In MOOP, there are M objective functions $f(x) = (f_1(x), f_2(x), ..., f_M(x))^T$, each of which has to be either maximized or minimized. A maximization problem can be converted into minimization problem by multiplying the objective function by -1. This is called the Duality principle. This

helps in solving mixed type of problems. Quite a few algorithms are designed to solve only one type of problems like minimization problems. The duality principle comes in handy when we are solving a maximization problem using such algorithms.

2.1.1 Linear and Nonlinear MOOP

If all the objective functions and all the constraints are linear, the resulting MOOP is called *multi-objective linear program* (MOLP).

However, even if one of the objective functions or constraints is nonlinear, the problem is called *nonlinear multi-objective problem.*

2.1.2 Convex and Nonconvex MOOP

Before defining a convex and nonconvex function, we must define what a convex function is.

Definition 1 *A function $f : \Re^n \rightarrow \Re$ is a convex function if for any two pair of solutions $x^{(1)}$, $x^{(2)}$, the following condition is true:*

$$f(\lambda x^{(1)} + (1-\lambda)x^{(2)}) \leq \lambda f(x^{(1)}) + (1-\lambda)f(x^{(2)}) \tag{2.2}$$

for all $0 \leq \lambda \leq 1$

Following properties can be defined from the above definition:

1. The linear approximation of $f(\mathbf{x})$ at any point in the interval $[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]$ always *underestimates* the actual function value.

2. The Hessian matrix of $f(\mathbf{x})$ is positive definite for all $\mathbf{x}$

3. For any convex function, a local minimum is always a global minimum.

A convex function is shown in Fig.2.1.

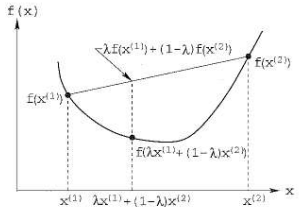

Figure 2.1: A convex function

If, in eq 2.2, we replace the $\leq$ sign with $>$ sign, the function becomes nonconvex. To check if the function is convex, we find the Hessian matrix $\nabla^2 f$ and check if it is positive definite i.e. all eigen values are positive. To check for nonconvex function, we find Hessian matrix $-\nabla^2 f$. If this matrix is positive definite, then the function is nonconvex.

Now we define a convex MOOP:

Definition 2 *A multi-objective optimization problem is convex if all objective functions are convex and the feasible region is convex (or if all inequality constraints are nonconvex and equality constraints are linear).*

2.2 Principles of Multi-objective Optimization

When multiple conflicting objectives are important, there cannot be a single optimum solution which simultaneously optimizes all objectives. The resulting outcome is a set of optimal solutions with a varying degree of objective values.

2.2.1 Pareto-optimal Front

For a non-trivial multi-objective optimization problem, no single solution exists that simultaneously optimizes each objective. In that case, the objective functions are said to be conflicting, and there exists a (possibly infinite) number of Pareto optimal solutions. A solution is called non dominated, Pareto Optimal, Pareto Efficient or non-inferior, if none of the objective functions can be improved in value without degrading some of the other objective values.

2.2.2 Objectives in Multi-objective Optimization

In principle, for the two objective optimization problem, the search space can be divided into two non-overlapping regions, namely optimal and non-optimal.

In the presence of multiple Pareto-Optimal solutions, it is difficult to prefer one solution over the other without any further information. Hence, it is important to find as many Pareto-optimal solutions as possible. Thus we can define two goals in an MOOP:

1. To find a set of solutions as close as possible to the Pareto-optimal front
2. To find a set of solutions as diverse as possible

The first goal is mandatory for any task of optimization, while the second goal is specific to the MOOP.

2.2.3 Non Conflicting Objectives

Multiple Pareto-optimal solutions exist only if the objectives are conflicting to each other.

2.3 Dominance

2.3.1 Concept of Domination

Most MOOPs use the concept of dominance. In these algorithms, two solutions are compared to each other on the basis whether one dominates the other or not.

Definition 3 *A solution $x^{(1)}$ is said to dominate solution $x^{(2)}$ if both the following conditions are true:*

1. *The solution $x^{(1)}$ is no worse than solution $x^{(2)}$ in all the objectives.*

2. *The solution $x^{(1)}$ is strictly better than solution $x^{(2)}$ in at least one objective.*

To understand the concept of dominance properly, let us take us two objective problem with five solutions. The problem is shown in Fig 2.2.

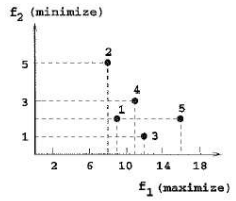

Figure 2.2: Example for Dominance

In the example, we need to maximize objective function 1 while we need to minimize objective function 2. Since both the objective functions are equally important to us, we use the concept of dominance to find optimal solution.

Starting with 1, we compare it with the other solutions. Let's say we first compare it with solution 2. We see that solution 1 is better than solution 2 in objective function 1 as well as 2. So we can say that 1 dominates 2.

Similarly, we see that when solution 1 is compared with solution 5, 5 dominates 1 in objective function 1, but in objective function 2 they are equivalent (i.e. 5 is no worse than 1). Hence, we can say that 5 dominates 1.

2.3.2 Properties of Dominance Relation

From definition 3 we define the concept of dominance. From this we can derive 3 outcomes of dominance between solutions 1 and 2, i.e.

1. Solution 1 dominates solution 2,

2. Solution 2 dominates solution 1, or

3. Solution 1 and 2 do not dominate each other.

The binary relation properties of dominance operator are:

- Reflexive: Dominance solution is not reflexive (as per definition 2.3).

- Symmetric: Dominance relation is also not symmetric since $p \leq q$ doesn't imply $q \leq p$.

- Antisymmetric: Since the dominance relation is not symmetric, it can't be antisymmetric as well.

- Transitive: Dominance relation is transitive as if $p \leq q$ and $q \leq r$, then $p \leq r$.

2.3.3 Procedures for Finding Non Dominance

For finding a non-dominated set, we discuss here three approaches, starting from a naïve and slow approach to an efficient and fast approach.

Approach 1: Each solution is compared to every other solution in the population to check if it is dominated by any solution in the population. If a solution i is found to be dominated by any solution, then there exists at least one solution in the population which is better than i in all objectives. Hence, the solution i cannot belong to the non-dominated set. We mark a flag against solution i to denote that it does not belong to non-dominated set. However, if no solution is found to dominate solution i, it is a member of non-dominated set. This is followed for all the solutions in the population.

The algorithm for Approach 1:

Step 1 Set solution counter $i = 1$ and create an empty non-dominated set P'.

Step 2 For a solution $j \in P (j \neq i)$, check if solution j dominates solution i. If yes, go to Step 4.

Step 3 If more solutions are left in P, increment j by one and go to Step 2; otherwise, set $P' = P' \cup \{i\}$.

Step 4 Increment i by one. If $i \leq N$, go to Step 2; otherwise, stop and declare P' as non-dominated set.

Approach 2-Continuously Updated: This approach is similar to the first one, except that here each solution is checked with a partially filled population for domination.

The algorithm for Approach 2:

Step 1 Initialize $P' = 1$. Set counter $i = 2$.

Step 2 Set $j = 1$.

Step 3 Compare solution i with j from P' for domination.

Step 4 If i dominates j, delete the j^{th} member from P' or update $P = P' \backslash \{P' \{j\}\}$. If $j < |P|$, increment j by one and the go to Step 3. Otherwise, go to Step 5. Alternatively, if the j^{th} member of P' dominates i, increment i by one and then go to Step 2.

Step 5 Insert i in P' or update $P' = P \cup \{i\}$. If $i < N$, increment i by one and go to Step 2. Otherwise, stop and declare P' as the non-dominated set.

Approach 3-Kung et al.'s Efficient Method: This approach first sorts the population according to the descending order of importance to the first objective function value. Thereafter, the population is recursively halved as top (T) and bottom (B) sub-populations. Knowing that the top half is better than the bottom half population in terms of first objective function, the bottom half is checked for domination over top half. The solutions of B that are not dominated by any member of T, are combined with members of T to form a merged population M. The merging and the domination check starts with the innermost case and then proceeds in a bottom-up fashion. This is the most computationally efficient method.

The algorithm for Approach 3:

Step 1 Sort the population according to the descending order of importance in first objective function and rename the population as P of size N.

Step 2, Front(P) If $|P| = 1$, return P as the output of Front(P). Otherwise, $T =$ **Front**$(P^{(1)} - P^{(|P|/2)})$ and $B =$ **Front**$(P^{(|P|/2+1)} - P^{(|P|)})$. If the i^{th} non-dominated solution of B is not dominated by any non-dominated solution of T, create a merged set $M = T \cup \{i\}$. Return M as the output of **Front**(P).

What finally returns from Step 2 is the non-dominated set. It is important to mention that Step 2 uses self-recursion.

The speed comparison of all the three methods are shown in Fig 2.3.

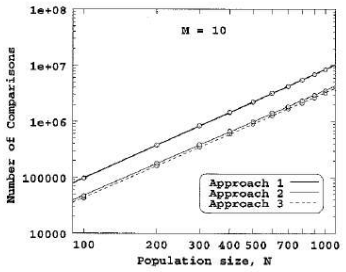

Figure 2.3: Speed Comparison for Three Approaches

Here we see that Approach 1 is the slowest; and, while there is very little difference between their speeds, Approach 3 is still faster than Approach 2.

An example for Approach 2 is given below for better understanding of the concept:

Given the solutions as in Fig 2.4 with five solutions, we need to find the non-dominated set. To solve this two objective optimization problem, we will use Approach 2. As we have seen above, Approach 2 is a five step algorithm. What follows is a step-by-step solution procedure to solve the problem.

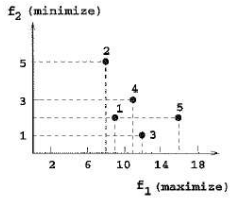

Figure 2.4: Example for Approach 2

Step 1 $P' = \{1\}$ and we set $i = 2$.

Step 2 We set the solution counter of P' as $j = 1$ (which refers to solution 1).

Step 3 We now compare solution 2 ($i = 2$) with the lone member of P' (solution 1) for domination. We observe that solution 1 dominates solution 2. Since the j^{th} member of P' dominates solution i, we increment i to 3 and go to Step 2. This means that solution 2 does not belong to the non-dominated set.

Step 2 The set P' still has solution 1 ($j = 1$) only.

Step 3 Now we compare solution 3 with solution 1. We observe that solution 3 ($i = 3$) dominates solution 1. Thus, we delete the j^{th} (or the first) member from P' and update $P = \emptyset$. Thus, $|P'| = 0$. This depicts that a non-member of the non-dominated set gets deleted from P'. We now move to Step 5.

Step 5 We insert $i = 3$ in P' or update $P' = \{3\}$. Since $i < 5$ here, we increment i to 4 and move to Step 2.

Step 2 We set $j = 1$, which refers to the lone element (solution 3) of P'.

Step 3 By comparing solution 4 with solution 3, we observe that solution 3 dominates solution 4. Thus, we increment $i = 5$ and move to Step 2.

Step 2 We still have solution 3 in P'.

Step 3 Now, we compare solution 5 with solution 3. We observe that neither of them dominates the other. Thus, we move to Step 5.

Step 5 We insert solution 5 in P' and update $P = \{3, 5\}$. Since $i = 5$, we stop and declare $P' = \{3, 5\}$ as the non-dominated set.

By looking at the algorithms of both the approaches we can say, without the loss of generality, that Approach 2 is faster than Approach 1.

2.3.4 Non-Dominated Sorting of a Population

We have seen that many algorithms use the concept of dominance to find optimal solutions. For this they find a non-dominant set and that is the optimal solution. But many evolutionary algorithms classify sets into ranks, i.e. one non-dominated set of level 1 and other non-dominated sets at different levels.

After finding level 1 non-dominated set, to find the other non-dominating sets, we temporarily disregard this set form the list. And this procedure is followed until all the levels are found.

The algorithm for finding levels of non-dominated sorting is as follows:

Step 1 Set all non-dominated $P_j(j = 1, 2, ...)$, as empty sets. Set non-domination level counter $j = 1$.

Step 2 Use any one of the Approaches to find the non-dominated set P' of population P.

Step 3 Update $P_j = P'$ and $P = P'$.

Step 4 If $P' \neq \emptyset$, increment j by one and go to Step 2. Otherwise, stop and declare all non-dominated sets P_i,for $i = 1, 2, , j$.

2.4 Interval Numbers

Taking reference from [3] we define interval numbers.

Consider a closed, real interval $X = [a, b]$. An interval number X is such a closed interval, i.e. it is the set $[x = \{x | a \leq x \leq b\}$ of all real numbers between and including a and b. This process of representing a number or a solution within bounds is called *enclosing* the solution.

A real number x is equivalent to an interval $[x, x]$, which has zero width. Such an interval is said to be *degenerate*.

If the end points of an interval are not representable on a given computer, then lower limit a is rounded down to the largest machine-representable number (a) and upper limit b is rounded up to the smallest machine-representable number (b). This is called *outward rounding*.

2.5 Notation and Relations of Interval Numbers

When a real quantity is expressed in lower case, then interval quantity is expressed in capital letter. For example, if y denotes real variable then Y denotes interval variable. If real quantity is expressed

in capital letters (say matrix $\mathbf{X}$), then corresponding interval variable is expressed with superscript I (matrix $\mathbf{X}^I$). Similarly, for a function $f(y)$ corresponding interval vector is $f(Y)$. An under-bar denotes lower limit while an over-bar denotes upper limit of an interval. For example, if $A = [x, y]$, then $\bar{A} = y$ and $\underline{A} = x$. An interval $A = [x, y]$ is said to be *positive* if $x > 0$ and *non-negative* if $x \geq 0$. It is said *negative* if $y < 0$ and *non-positive* if $y \leq 0$. Intervals $[a, b]$ and $[c, d]$ are *equal* if and only if $a = c$ and $b = d$.

Throughout this text, interval number will be represented by X for any real number x.

2.6 Interval Number System

Let $\cdot$ denote any of the arithmetic operations, i.e. addition $(+)$, subtraction $(-)$, multiplication $(\times)$ and division $(\div)$, on real numbers x and y, then the corresponding operation for arithmetic on interval numbers X and Y is $X \cdot Y = \{x \cdot y | x \in X, y \in Y\}$.

This definition produces the following rules for generating endpoints of $X \cdot Y$ from the two intervals $X = \left[\underline{X}, \overline{X}\right]$ and $Y = \left[\underline{Y}, \overline{Y}\right]$.

$$X + Y = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}] \tag{2.3}$$

$$X - Y = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}] \tag{2.4}$$

$$X \times Y = \begin{cases} [\underline{X}\underline{Y}, \overline{X}\overline{Y}] \; if \; \underline{X} \geq 0 \; and \; \underline{Y} \geq 0 \\ [\overline{X}\underline{Y}, \overline{X}\overline{Y}] \; if \; \underline{X} \geq 0 \; and \; \underline{Y} \leq 0 \leq \overline{Y} \\ [\overline{X}\underline{Y}, \underline{X}\overline{Y}] \; if \; \underline{X} \geq 0 \; and \; \overline{Y} \leq 0 \\ [\underline{X}\overline{Y}, \overline{X}\overline{Y}] \; if \; \underline{X} < 0 < \overline{X} \; and \; \underline{Y} \geq 0 \\ [\overline{X}\underline{Y}, \underline{X}\underline{Y}] \; if \; \underline{X} < 0 < \overline{X} \; and \; \overline{Y} \leq 0 \\ [\underline{X}\overline{Y}, \overline{X}\underline{Y}] \; if \; \overline{X} \leq 0 \; and \; \underline{Y} \geq 0 \\ [\underline{X}\overline{Y}, \underline{X}\underline{Y}] \; if \; \overline{X} \leq 0 \; and \; \underline{Y} < 0 < \overline{Y} \\ [\overline{X}\overline{Y}, \underline{X}\underline{Y}] \; if \; \overline{X} \leq 0 \; and \; \overline{Y} \leq 0 \\ [min(\overline{X}\underline{Y}, \underline{X}\overline{Y}), max(\underline{X}\underline{Y}, \overline{X}\overline{Y})] \; if \; \underline{X} < 0 < \overline{X} \; and \; \underline{Y} < 0 < \overline{Y} \end{cases} . \tag{2.5}$$

$$\frac{1}{Y} = \left[\frac{1}{\overline{Y}}, \frac{1}{\underline{Y}}\right] and \frac{X}{Y} = X \times \left(\frac{1}{Y}\right) \tag{2.6}$$

For division, we exclude division by interval containing 0.

Here, we have to note a very important point regarding subtraction in interval analysis. Now, we know that in regular mathematics, when we subtract a number by itself, we get 0. But, this is not the case in interval analysis. Let's see this with an example.

If we take an interval $X = [\underline{X}, \overline{X}]$, then

$$X - X = [\underline{X} - \overline{X}, \overline{X} - \underline{X}]$$

If, and only if, the interval is degenerate, i.e. $\underline{X} = \overline{X}$, we will get the answer as 0, not in any other case.

Multiplication, in short, can also be represented in a simpler way [9]:

$$X \times Y = \min\{\overline{XY}, \overline{X}\underline{Y}, \underline{X}\overline{Y}, \underline{XY}\} \tag{2.7}$$

2.7 Intersection, Union and Interval Hull

In this section, we define some terms for interval analysis, that we know for real mathematics.

Given two sets $X = [\underline{X}, \overline{X}]$ and $Y = [\underline{Y}, \overline{Y}]$.

Intersection: We know that intersection means taking out the common terms from different sets. In interval analysis also we can define intersection as follows:

$$\begin{aligned} X \cap Y &= \{z : z \in X \; or \; z \in Y\} \\ &= [max(\underline{X}, \underline{Y}), min(\overline{X}, \overline{Y})] \end{aligned} \tag{2.8}$$

But if we have $\underline{Y} < \overline{X}$ or $\overline{X} < \underline{Y}$, then the intersection is an empty set denoted by $\emptyset$, i.e.

$$X \cap Y = \emptyset \tag{2.9}$$

Union: Union of sets is defined as the set containing all the elements of all the individual sets involved in the union. In interval analysis we define union as follows:

$$\begin{aligned} X \cup Y &= \{z : z \in X \; or \; z \in Y\} \\ &= [min(\underline{X}, \underline{Y}), max(\overline{X}, \overline{Y})] \end{aligned} \tag{2.10}$$

Now, a union is an interval only if the individual intervals have a non-empty intersection, i.e. only in the former case. If the intersection is empty then we cannot define a union.

Interval Hull: Now, the union of two intervals defined above is not always an interval. So, to remove this problem, we define an interval hull, as follows:

$$X \underline{\cup} Y = [min(\underline{X}, \underline{Y}), max(\overline{X}, \overline{Y})] \tag{2.11}$$

So, we can say that:

$$X \cup Y \subset X \underline{\cup} Y \tag{2.12}$$

Let's take an example to understand this. Let's say we have two intervals $X = [-2, 0]$ and $[2, 4]$. Now, clearly these two sets are disconnected and so, their intersection is empty. So, here we find interval hull $X \underline{\cup} Y = [-2, 4]$. Although, information is lost while taking interval hull, but most of the times this information is useless.

Since we are using interval analysis for optimization purposes, it is worth mentioning the importance of intersection here. Our task is to find the optimal solution, and for that we need to reach as much close to actual value as possible. And this is where intersection plays a key role. When we perform intersection, we are taking common part of many intervals and then removing all the part that is not common. This way we reduce our search space.

Other way intersection is important is the fact that it ensures the correctness of different measurements. Now, if the solution lies in all two intervals then it must also lie in the intersection of the

two intervals. If the intersection does not contain the solution then either of the solutions is wrong. Let's take an example to understand this. Suppose two people make independent measurements of the same physical quantity q. One finds that $q = 10.3$ with a measurement error less than 0.2. The other finds that $q = 10.4$ with an error less than 0.2. We can represent these measurements as the intervals $X = [10.1, 10.5]$ and $Y = [10.2, 10.6]$, respectively. Since q lies in both, it also lies in $X \cap Y = [10.2, 10.5]$. An empty intersection would imply that at least one of the measurements is wrong.

2.8 Width, Absolute Value, Midpoint

In this section, some other important terms related to interval analysis are defined. Refer to Fig. 2.5 and the definitions below.

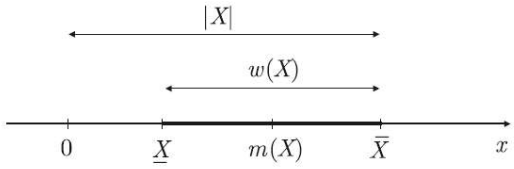

Figure 2.5: Width, Absolute Value, Midpoint

Width: Width of an interval is denoted by the difference of the respective upper and lower bounds. Mathematically, we can say:

$$w(X) = \overline{X} - \underline{X} \tag{2.13}$$

Absolute Value: As the name suggests, it is the magnitude. But, since we are working in intervals, we have to take care of the bounds. So, the absolute value is defined as below:

$$|X| = max\left\{|\underline{X}|, |\overline{X}|\right\} \tag{2.14}$$

Midpoint: As the name suggests, it is the middle point of the interval. It is defined as below:

$$m(X) = \frac{1}{2}\left(\underline{X} + \overline{X}\right) \tag{2.15}$$

A useful formula: Any interval X can be represented as:

$$X = m(X) + [-\frac{1}{2}w(X), \frac{1}{2}w(X)] \tag{2.16}$$

2.9 Interval Vectors and Matrices

An n-dimensional interval vector can be represented as $X = (X_1, X_2, ..., X_n)$. For example, let's take 2-dimensional case, where $X = (X_1, X_2) = \left(\left[\underline{X_1}, \overline{X_1}\right], \left[\underline{X_2}, \overline{X_2}\right]\right)$. It can be represented as a rectangle in the x_1x_2-plane, as shown inf Fig. 2.6

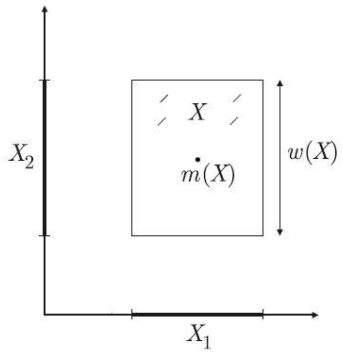

Figure 2.6: Two Dimensional Interval Vector

Now, this can be extended very easily to n-dimensions as well. The properties of simple interval numbers can be extended to interval vectors as well, as shown below:

- If $x = (x_1, x_2, ..., x_n)$ is an n-dimensional real vector, then $X = (X_1, X_2, ..., X_n)$ is n-dimensional interval vector.

- Intersection of two n-dimensional vectors X and Y will be empty if any one of it's components is empty, i.e. $X \cap Y = \emptyset$ if for some i we have $X_i \cap Y_i = \emptyset$. Otherwise, we have

$$X \cap Y = (X_1 \cap Y_1, X_2 \cap Y_2, \ldots, X_n \cap Y_n) \tag{2.17}$$

- Width of $X = (X_1, X_2, ..., X_n)$ is given by the largest of the widths of it's components.

$$w(X) = \max_i w(X_i) \tag{2.18}$$

- Midpoint of vector is given by a set containing the midpoints of all the individual intervals.

$$m(X) = (m(X_1), m(X_2), ..., m(X_n)) \tag{2.19}$$

- The norm of interval vector is given by

$$\|X\| = \max_i |X_i| \tag{2.20}$$

2.10 Properties of Interval Arithmetic

We already saw some of the arithmetic operation of interval analysis previously. In this section, we will see some more properties relating to interval analysis.

2.10.1 Arithmetic Properties

- Interval analysis follows *commutative law*, for intervals X, Y, Z as

$$X + Y = Y + X$$

$$XY = YX \tag{2.21}$$

- Interval analysis also follows *associative law*, for intervals X, Y, Z as

$$X + (Y + Z) = (X + Y) + Z$$

$$X(YZ) = (XY)Z \tag{2.22}$$

- We define *identity elements* for addition and multiplication as, for intervals X, Y as

$$X + 0 = 0 + X = X$$

$$X \times 1 = 1 \times X = X \tag{2.23}$$

  Here 0 and 1 are degenerate intervals.

2.10.2 Inverse Elements

We saw how subtraction is done in interval arithmetic. There we saw that when we subtract an interval X from itself, we DO NOT get 0. Similarly, when we divide an interval X by itself, we DO NOT get 1. This is shown here: Let's take an interval $X = [\underline{X}, \overline{X}]$.

Then when we subtract it from itself, we get

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

It is clear that this will be 0 only when we have $\underline{X} = \overline{X}$.

Similarly, when we divide interval $X = [\underline{X}, \overline{X}]$ with itself, we will get

$$X/X = \begin{cases} [\underline{X}/\overline{X}, \overline{X}/\underline{X}] \; if \; 0 < \underline{X} \\ [\overline{X}/\underline{X}, \underline{X}/\overline{X}] \; if \; \underline{X} < 0 \end{cases} \tag{2.25}$$

2.10.3 Subdistributivity

In normal arithmetic, *distributive law* is defined as

$$x(y + z) = xy + xz \tag{2.26}$$

But in interval mathematics, this is not always possible. Let's take an example to see this. Given

$X = [1, 2]$, $Y = [1, 1]$, $Z = -[1, 1]$. From LHS of 2.26

$$\begin{aligned} X(Y + Z) &= [1, 2] \cdot ([1, 1] - [1, 1]) \\ &= [1, 2] \cdot [0, 0] \\ &= [0, 0] \end{aligned}$$

While the RHS gives

$$\begin{aligned} XY + XZ &= [1, 2] \cdot [1, 1] - [1, 2] \cdot [1, 1] \\ &= [1, 2] - [1, 2] \\ &= [-1, 1] \end{aligned}$$

Hence, the *subdistributive law* is defined for interval mathematics. It is defined as below

$$X(Y + Z) \subseteq XY + XZ \tag{2.27}$$

2.10.4 Symmetricity of Intervals

An interval is called symmetric when it has following property

$$\overline{X} = -\underline{X} \tag{2.28}$$

Use of symmetric intervals helps reduce the complexity of analysis

$$\begin{aligned} &X + Y = X - Y = (|X| + |Y|)[-1, 1] \\ &XY = |X|\,|Y|\,[-1, 1] \\ &X(Y \pm Z) = XY + XZ = |X|\,(|Y| + |Z|)[-1, 1] \end{aligned} \tag{2.29}$$

2.11 Interval Functions

Let f be a real valued function of single variable x. We want to know how the function $f(x)$ varies with variation in x over the range of interval X, i.e. we want to know the image of set X under mapping f:

$$f(X) = \{f(x) : x \in X\} \tag{2.30}$$

Extending it to n-variables, we can write:

$$f(X_1, ..., X_n) = \{f(x_1, ..., x_n) : x_1 \in X_1, ..., x_n \in X_n\} \tag{2.31}$$

Definition 4 *Let $g : M_1 \to M_2$ be a mapping between sets M_1 and M_2, and let $S(M_1)$ and $S(M_2)$ be family of subsets of M_1 and M_2. The united extension of g is the set-valued mapping $\overline{g} : S(M_1) \to S(M_2)$, such that*

$$\overline{g}(X) = \{g(x) : x \in X, X \in S(M_1)\} \tag{2.32}$$

2.12 Elementary Function of Interval Arguments

For some functions, we can easily compute (2.30). For example, let's take

$$f(x) = x^2 \quad x \in \Re \tag{2.33}$$

If $X = [\underline{X}, \overline{X}]$, then

$$f(X) = \{x^2 : x \in X\} \tag{2.34}$$

can be expressed as

$$f(X) = \begin{cases} [\underline{X}^2, \overline{X}^2] \; if \; \underline{X} < 0 < \overline{X} \\ [\overline{X}^2, \underline{X}^2] \; if \; \overline{X} < 0 < \underline{X} \\ [0, max[\underline{X}^2, \overline{X}^2] \; if \; \underline{X} < 0 < \overline{X} \end{cases} \tag{2.35}$$

This can be seen in the Fig. 2.7

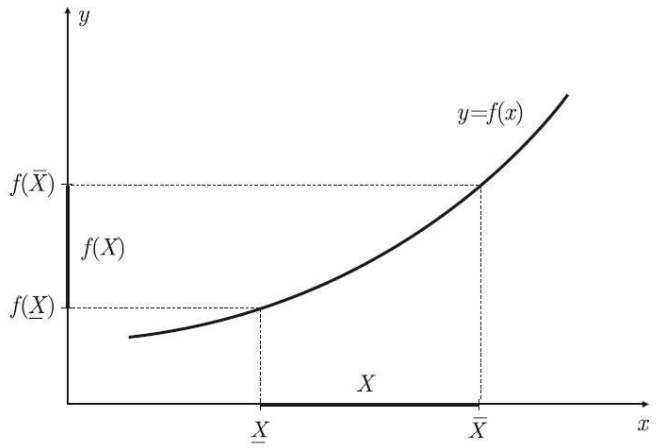

Figure 2.7: Graph for $f(X) = X^2$

2.12.1 Interval Extension for Functions

When the function f is a single interval variable function, we give it's interval extension as follows:

Definition 5 *We say that F is an interval extension of function f, if for degenerate interval arguments, F agrees with f:*

$$F([x, x]) = f(x) \tag{2.36}$$

When the function f is multi-variable, i.e.

$$f = f(X_1, X_2, ..., X_n),$$

we give it's interval extension as follows:

Definition 6 *By an interval extension of function f, we mean an interval-valued function F of n*

interval variables $X_1, X_2, ..., X_n$ *such that for real arguments* $x_1, x_2, ..., x_n$ *we have*

$$F(X_1, X_2, ..., X_n) = f(X_1, X_2, ..., X_n) \tag{2.37}$$

2.12.2 Inclusion Isotonicity

Inclusion Isotonicity is defined as:

Definition 7 *We say that* $F = F(X_1, ..., X_n)$ *is inclusion isotonic if* $Y_i \subseteq X_i$ *for* $i = 1, ..., n \Rightarrow F(Y_1, ..., Y_n) \subseteq F(X_1, ..., X_n)$. *Observe carefully that united extensions, which all have the subset property, are inclusion isotonic. In particular, then, the operations of interval arithmetic must satisfy* $Y1 \subseteq X_1, Y_2 \subseteq X_2 \Rightarrow Y_1 \odot Y_2 \subseteq X_1 \odot X_2$.

2.13 Convergence and Continuity

Given a sequence of intervals X_k, we say that it is converging if there exists and interval X^*, such that for every $\varepsilon > 0$, we get $d(X_k, X^*) < \varepsilon$, whenever $k > N$, where $N = N(\varepsilon)$ is a natural number. We can write

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

Here, d is given by

$$d(X, Y) = \max\{\left|\underline{X} - \underline{Y}, \overline{X} - \overline{Y}\right|\} \tag{2.38}$$

2.14 Refinements

Fundamental theorem of interval analysis gives us

$$f(x_1, x_2, ..., x_n) \subseteq f(X_1, X_2, ..., X_n) \tag{2.39}$$

2.14.1 Lipschitz Interval Extensions

We see a process as to how $f(x_1, x_2, ..., x_n)$ may be approximated as close to desired answer by finite union of intervals.

Definition 8 *An interval extension* F *is said to be Lipschitz in* X_0 *if there is constant* L *such that* $x(F(X)) \leq Lw(X)$ *for every* $X \subseteq X_0$

2.14.2 Subdivisions and Refinements

Definition 9 *By a uniform subdivision of an interval vector* $X = (X_1, X_2, ..., X_n)$, *we mean*

$$X_{i,j} = [\underline{X}_i + (j - i)w(X_i)/N, \underline{X}_i + jw(X_i)/N], \quad j = 1, 2, ...N$$

where N *is a positive integer.*

We have $X_i = \bigcup_{j=1}^{N} X_{i,j}$ *and* $w(X_{i,j}) = w(X_i)/N$. *Furthermore,*

$$X_i = \bigcup_{j_i=1}^{N} (X_{1,j_1}, X_{2,j_2}, ..., X_{n,j_n}) \tag{2.40}$$

with $w(X_{1,j_1}, X_{2,j_2}, ..., X_{n,j_n}) = w(X)/N$

Definition 10 *Let* $F(X)$ *be an inclusion isotonic, Lipschitz, interval extension for* $X \subseteq X_0$*. The interval quantity*

$$F_{(N)}(X) = \bigcup_{ji=1}^{N} F(X_{1,j_1}, ..., X_{n,j_n}) \tag{2.41}$$

is called refinement of F *over* X*.*

If X and Y are intervals such that $X \subseteq Y$, then there is an interval E with $\underline{E} \leq 0 \leq \overline{E}$ such that $Y = X + E$ and $w(Y) = w(X) + w(E)$. If F is an inclusion isotonic interval extension of f with $F(X)$ defined for $X \subseteq X_0$, then $f(X) \subseteq F(X)$ for $X \subseteq X_0$. We have $F(X) = f(X) + E(X)$ for some interval-valued function $E(X)$ with $w(F(X)) = w(f(X)) + w(E(X))$.

Theorem 1 *If* $F(X)$ *is an inclusion isotonic, Lipschitz, interval extension for* $X \subseteq X_0$*, then the excess width of a refinement* $F_{(N)}(X)$ *is of order* $1/N$*. We have*

$$F_{(N)}(X) = f(X_1, ..., X_n) + E_N, \tag{2.42}$$

where $w(E_N) \leq Kw(X)/N$ *for some constant* K

2.15 Finite Convergence and Stopping Criteria

2.15.1 Finite Convergence

Definition 11 *By the finite convergence of a sequence* X_k*, we mean there is a positive integer* K *such that* $X_k = X_K$ *for* $k \geq K$*. Such a sequence converges in* K *steps.*

For example, let's take

$$X_0 = [1, 2], \quad X_{k+1} = 1 + X_k/3, \quad (k = 0, 1, 2, ..)$$

The sequence goes like this

$$\begin{aligned} X_0^* &= [1, 2], \\ X_1^* &= [1.33, 1.67], \\ X_2^* &= [1.44, 1.56], \\ X_3^* &= [1.48, 1.52], \\ X_4^* &= [1.49, 1.51], \\ X_5^* &= [1.49, 1.51], \end{aligned}$$

and $X_k = X$ for all $k \geq 4$. We have finite convergence in four steps.

2.15.2 Natural Stopping Criterion

Any interval sequence that has nested sequence, we have a natural stopping criteria for it. Since the sequence $\{X_k\}$ converges in finite number of steps, we can compute X_k until

$$X_{k+1} = X + k \tag{2.43}$$

If X_k generated are of the form

$$X_{k+1} = F(X_k) \tag{2.44}$$

such that each X_{k+1} depends only on previous X_k , then (2.43) guarantees successful convergence. 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

$$X_{k+1} = F(X_k) \qquad (k = 0, 1, 2, ..) \tag{2.45}$$

is nested with

$$\mathrm{X}_0 \supseteq X_1 \supseteq X_2 \tag{2.46}$$

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

2.15.3 Skelboe's Algorithm

Skelboe [10] introduced an efficient algorithm for refinements. We first find the lower bound for f, then we find lower bound for $-f$, which is upper bound for f. Skelboe shows that, in computing refinements, no subdivision is necessary of argument intervals for arguments occurring only once. Here, a simplified version of this algorithm is presented, called *Skelboe-Moore Algorithm* [2].

Suppose we are given an interval extension $F(X)$ of $f \subset FC_n(X_0)$, and we want to bound the range of values of f in X_0 using refinements of F. We first find a lower bound and then, replacing $F(X)$ by $-F(X)$, repeat the procedure to be described to find an upper bound. If the evaluations are done in IA, the procedure will converge in a finite number of steps. To find a lower bound, we create a list of pairs $(Y, F(Y))$, ordered so that $(Y, F(Y))$ comes before $(Z, F(Z))$ in the list only if $F(Y) \leq F(Z)$. The interval extension used should be having inclusion isotonicity. The better the extension, the fewer the steps to convergence.

Algorithm [2][11]: Given an interval extension F of f , a tolerance ε, and a box X_0, this algorithm returns a lower bound (LB) on the range of f over X that is within ε of the actual minimal value of lower bound as well as lists C and L of boxes that have been produced during the process of computing these, such that $C \cup L = X_0$.

1. Set[#] $f_{ub} \leftarrow f(m(X_0))$, where $f(m(X_0))$ is computed with either upward rounding or interval arithmetic[##] as $\overline{F(m(X_0))}$;

2. Set $X \leftarrow X_0$;

3. Initially, lists L and C are empty;

4. Bisect X in coordinate direction i such that $w(X_i) = w(X) = max_{1 \leq i \leq n} w(X_i) : X = X^{(1)} \cup X^{(2)}$;

5. $f_{ub} \leftarrow min\{f_{ub}, \overline{F(m(X^{(1)}))}, \overline{F(m(X^{(2)}))}\}$;

IF $\max\{\overline{F(X^{(1)})}, \overline{F(X^{(2)})}\} - \min\{\overline{F(X^{(1)})}, \overline{F(X^{(2)})}\} < \varepsilon$, then

(a) Place $X^{(1)}$ and $X^{(2)}$ into C in order

(b) IF L isnotempty $THEN$

(i) remove the first item from L and place its box into X;

(ii) IF $\underline{F(X)} > f_{ub}$ $THEN$

$RETURN$ with LB equal to lower bound on the first box in C and with the lists C and L

END IF

6. $ELSE$

$RETURN$ with LB equal to the lower bound on the first box in C and with list C

END IF

$ELSE$

(a) enter the items $(X^{(1)}, \underline{F(X^{(1)})})$ and $(X^{(2)}, \underline{F(X^{(2)})})$ in proper order in the list L;

(b) set $X \leftarrow$ the argument (first member of the pair) of the first item in the list L (with lowest $\underline{F(X)}$) and remove the item $(X, \underline{F(X)})$ from the list;

END IF

7. IF L is not empty, $THEN$ return to step (4).

Step (4) means, we replace X by two interval vectors,

$$\mathrm{X}^{(1)} = (\mathrm{X}_1,\ldots,\mathrm{X}_{i-1},\mathrm{X}^{(1)}{}_i,\mathrm{X}_{i+1},\ldots,\mathrm{X}_n),$$
$$\mathrm{X}^{(2)} = (\mathrm{X}_1,\ldots,\mathrm{X}_{i-1},\mathrm{X}^{(2)}{}_i,\mathrm{X}_{i+1},\ldots,\mathrm{X}_n),$$

where

$$\mathrm{X}^{(1)}{}_i = [\underline{\mathrm{X}_i}, \frac{1}{2}(\underline{\mathrm{X}_i} + \overline{\mathrm{X}_i})], \quad \mathrm{X}^{(2)}{}_i = [\frac{1}{2}(\underline{\mathrm{X}_i} + \overline{\mathrm{X}_i}), \overline{\mathrm{X}_i}].$$

Note:

- #We use $\leftarrow$ to denote operation of setting a value, while setting a value equal to something is given by $=$
- ## f_{ub} represents a mathematically rigorous upper bound on the lower bound of f over F

2.16 The Newton Method

Let f be a real valued function of real variable x, and is continuously differentiable. By mean value theorem, we can write

$$f(x) = f(y) + f'(s)(x - y) \tag{2.47}$$

for some s between x and y. Now let $[a, b]$ be an interval in which we seek a solution of the equation

$$f(x) = 0 \tag{2.48}$$

If the solution x exists, it should satisfy

$$f(y) + f'(s)(x - y) = 0 \tag{2.49}$$

for any $y \in [a,b]$, in particular for $y = m([a,b]) = (a+b)/2$. Hence,

$$x = y - f(y)/f'(s) \tag{2.50}$$

Let $F'(X)$ be be an inclusion monotonic interval extension of $f'(x)$ and consider the algorithm

$$X^{(k+1)} = X(k) \cap N(X^{(k)}) \qquad k = (0,1,2,...) \tag{2.51}$$

where

$$N(X) = m(X) - f(m(X))/F'(X) \tag{2.52}$$

It follows that x is contained in $N(X)$ if $y = m(X)$, and if x is contained in X, then s is also contained in X.

Theorem 2 *If an interval X^0 contains a zero x of $f(x)$, then so does X^k for all $k = 0,1,2,...$, defined by* (2.50)*. Furthermore, the intervals X^k form a nested sequence converging to x if $0 \notin F'(X^{(0)})$.*

An updated version was presented by Hansen [12]. What he did was that he applied Newton method to f' instead of f. So let X be a closed subinterval of $[a,b]$. Let x be any point in X, it is best if it is the mid point. If $y \in X$ is a stationary point of f, then $f'(y) = 0$ and y solves

$$f'(x) + (y-x)f''(\xi) = 0 \tag{2.53}$$

for some $\xi \in X$. Here $y \in X$ belongs to set

$$S'' = \{y : f'(x) + (y-x)f''(\xi') = 0, \;\; \xi' \in X\}$$

If $0 \notin f''(X)$, then set S'' is contained in

$$S' = x - f'(x)/f''(X) \tag{2.54}$$

If the iterative process is defined as

$$\begin{aligned} &N(X_n) = x_n - f'(x_n)/f''(x_n) \quad n = 0,1,... \\ &X_{n+1} = X_n \cap N(X_n) \end{aligned} \tag{2.55}$$

with x_n the mid point of X_n. This process never fails to converge when $0 \notin f''(X_0)$.

Theorem 3 *Let the extended Newton method be applied to finding zeros of $f'(x)$ in an interval X_0. Assume that, at the i^{th} step, $i = 1,2,3,...$, the method is applied to the largest remaining subinterval. Also assume that $f'(x)$ is continuous and has a finite number of distinct zeros in X_0. Then, for a sufficiently large value of i, the sum of the lengths of the remaining intervals is less than a positive prescribed number ε.*

2.17 The Krawczyk Method

We consider the following finite system of non-linear equations

$$\begin{aligned} &f_1(x_1, ..., x_n) = 0 \\ &. \\ &. \\ &. \\ &f_n(x_1, ..., x_n) = 0 \end{aligned} \tag{2.56}$$

or, in vector form we can write

$$f(x) = 0 \tag{2.57}$$

Suppose that f in (2.57) continuously differentiable in some open domain D. Suppose that we can compute inclusion isotonic interval extensions F and F' for f and f', defined on interval vectors $X \subseteq D$.

Theorem 4 *Let Y be a non-singular real matrix approximating the inverse of the real Jacobian matrix $F'(m(X))$ with elements $F'(m(X))_{ij} = \partial f_i(x)/\partial x_j$ at $x = m(X)$. Let y be a real vector contained in the interval vector $X \subseteq D$. Define $K(X)$ by*

$$K(X) = y - Yf(y) + \{I - YF'(X)\}(X - y) \tag{2.58}$$

If $K(X) \subseteq X$, then (2.57) has a solution in X. It is also in $K(X)$.

2.18 System of Linear Equations

Here we discuss the problem of applying interval mathematics to bound the solution of a non-linear equation [13]. The multi-dimensional Newton method is being extended and implemented in interval analysis.

Let $f(x)$ be a vector having components $f_i(x), (i = 1, ..., n)$, where $f_i(x)$ is a real rational function of a real function of a real vector x. Assume

$$f(y) = 0 \tag{2.59}$$

To obtain the solution y, we do Taylor series expansion of $f(y)$

$$f(x) = f(y) + \sum_{i=1}^{n} (x_i - y_i) \frac{\partial}{\partial x_i} f[y + k_i(x - y)] \tag{2.60}$$

where $k_i \in [0, 1]$. Since $f(y) = 0$, (2.60) becomes

$$J(x - y) = f(x) \tag{2.61}$$

where J is the Jacobian matrix with elements

$$J_{ji} = \frac{\partial}{\partial x} f_i[y + k_i(x - y)] \tag{2.62}$$

Now, let $X^{(0)}$ be an interval vector containing both x and y. Then

$$y + k_i(x - y) \in X^{(0)} \tag{2.63}$$

since $k_i \in [0, 1]$. Let $J(X^{(0)})$ denote the interval matrix obtained from J by replacing $y + k_i(x - y)$ by $X^{(0)}$ for all i. We assume $J(X^{(0)})$ does not contain a singular matrix. Let $V(X^{(0)})$ be an interval matrix containing the inverse of $J(X^{(0)})$. Then $J^{-1} \in V(X^{(0)})$ for all $k_i \in [0, 1]$ and hence the solution vector y is contained in

$$Y^{(0)} = x - V(X^{(0)})f(x) \tag{2.64}$$

Define $X^{(1)} = X^{(0)} \cap Y^{(0)}$. Note $X^{(1)}$ contains the solution Y since $X^{(0)}$ and $Y^{(0)}$ do. We next replace x by the midpoint of $X^{(1)}$, replace $X^{(0)}$ by $X^{(1)}$ and repeat the procedure. In this way, we obtain a sequence of interval vectors $X^{(i)}, (i = 0, 1, 2, ...)$ each containing the solution vector y. Under appropriate conditions, these interval vectors converge to the solution y. Note that every element of J may be an interval.

Chapter 3

Branch and Bound Technique

Interval branch and bound methods [14], [15] are deterministic methods, known to find with certainty, the global optima of a function within a box. With interval arithmetic, it is possible to find all global minima, with certainty, of objective function

$$f(x) = f(x_1, ..., x_n) \tag{3.1}$$

where the following bounds are known

$$\underline{x_i} \leq x_i \leq \overline{x_i} \quad for \quad 1 \leq i \leq n$$

The basic overview of the method is shown below:

1. Begin with box $X^{I^{(0)}}$ in which global optimization is sought.
2. Delete subboxes of $X^{I^{(0)}}$ that cannot contain solution.The methods are:
   (a) Delete subboxes where gradient is non zero.
   (b) Compute upper bounds on f at various sampled points. Smallest computed upper bound $\overline{\mathrm{f}}$ is upper bound on f^*. Delete subboxes wherein $f > \overline{f}$.
   (c) Delete subboxes wherein f is not convex.
3. Iterate step 2 to find small set for x^*. Then find f^* over this set.

We can explain the basic process in the following pictures, although the actual process involves a lot of other steps, which are explained later.

From the overview, we know that we have to initiate with defining a box for search area (Fig. 3.1).

Next, what we do is, we delete the boxes that do not contain the optimal solution (Fig. 3.2).

This is followed for a lot many iterations. And, once the iterations stop, we get the final solution (Fig. 3.3).

Although Fig. 3.3 shows only one optimal solution, there could be more solutions depending upon the function.

The other steps involved in finding the optimal solution are explained henceforth.

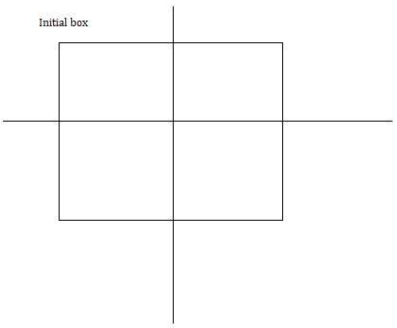

Figure 3.1: Initialising the box

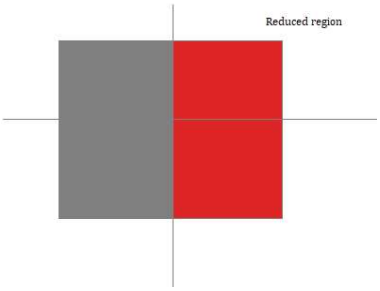

Figure 3.2: Next Steps in Iteration

3.1 The Steps Involved

3.1.1 The Initial Box

The initial box is specified by the user itself, depending upon the needs, as shown in the Fig. 3.1 above. It should be noted that, a user can declare multiple boxes initially, which may be either disjoint or may have some intersection. In both the cases, we have to treat the boxes as separate. Since our aim is to reduce the box size and reach the optimal solution, which is also indeed a box, it is always preferable to start with a box that is as small as possible, but at the same time big enough so that we do not miss the global optimal and end up with local solution.

So, let us take a function $f(X)$ and an initial box $X^{I^{(0)}}$. Our problem now becomes

$$\begin{aligned} &Minimize\ (globally)\ f(X) \\ &Subject\ to\ X \in X^{I^{(0)}} \end{aligned} \tag{3.2}$$

3.1.2 Gradient

We can assume that f is continuously differentiable, which means that we get global minimum where the gradient G of f is zero. But we also get G a zero at other local minima, maxima and saddle points. Our goal is reject any zero that is not a global minimum of f.

Let X^I be a subbox of the initial box $X^{I^{(0)}}$. We evaluate components of G over X^I, i.e. we evaluate $g_i(X^I)$ for $i = 1, ..., n$ and denote the resulting interval by $[\underline{g_i}(X^I), \overline{g_i}(X^I)]$.

Now, if we get either $\underline{g_i}(X^I) > 0$ or $\overline{g_i}(X^I) < 0$, then $g_i(X)$ can never be zero, hence optimal

Figure 3.3: The Final Solution

solution can never be found in X^I and this box should be deleted.

3.1.3 Upper Bound on Minimum

As we proceed with our algorithm, we evaluate f at various points of $X^{I^{(0)}}$. Every upper bound that we calculate is also the upper bound for the global minimum f^* of f. Following this, we use the smallest upper bound $\overline{\mathrm{f}}$.

Again let's take X^I as a subbox of $X^{I^{(0)}}$. Now in every step, X^I is reduced in size or split into more subboxes. Then we evaluate f at the center of each new box. Let's say we get the result as interval $[\underline{f^I}(X), \overline{f^I}(X)]$. Here, $\overline{f^I}$ is the upper bound on $f(X)$ and hence on f^*. If $\overline{\mathrm{f}}$ is the lowest such upper bound, then we can delete any subbox with $f > \overline{\mathrm{f}}$, since $\overline{\mathrm{f}} > f^*$. The method to delete the subbox is follows:

Let's take X^I as a subbox of $X^{I^{(0)}}$. We evaluate f over X^I and obtain $[\underline{f^I}(X), \overline{f^I}(X)]$. If $\underline{f^I}(X) > \overline{\mathrm{f}} > f^*$, then $f(X) > f^*$ for every $X \in X^I$. Therefore, we delete X^I.

3.1.4 Updating the Upper Bound

We already discussed that when we generate new subbox, we evaluate the value of function f at the center of new subbox and then update $\overline{\mathrm{f}} > f^*$. We can infact also search for some point in f where $f < f^*$ and further reduce $\overline{\mathrm{f}} > f^*$. For this, we do a line search for a point where f is small. The procedure is described now.

Suppose we evaluate gradient $G(X)$ of $f(X)$ at a point X. f decreases in the negative gradient direction from X. In a simple way, we can find small values for f if we search along this line. Let X be the center of the current box X^I. We define the half line of points $Y(\alpha) = X - \alpha G(X)$ where $\alpha > 0$.

We first determine the value of α' such that $Y(\alpha') = X - \alpha' G$ lies on the boundary of X^I, and restrict our search region. The procedure is:

If $f(X') < f(X)$, replace X by $(X + X')/2$. Otherwise, replace X' by $(X + X')/2$.

The procedure is applied a few times and the smaller of the final values of $f(X)$ and $f'(X)$ are used to update $\overline{\mathrm{f}}$.

If Newton method succeeds in getting a new box bounding solution for $G = 0$, we evaluate f at the center of the box and use the result to update $\overline{\mathrm{f}}$. Regardless, we always obtain an additional test point $Y = X - BG(X)$ at which we evaluate$f(X)$. Y can be anywhere, be it inside the box $X^{I^{(0)}}$ or outside, it doesn't matter.

3.1.5 Termination

Our algorithm splits and reduces the size of boxes. Thus, making them small. We require a set of conditions which will define the *stop point* for the algorithm, so that it does not go on unnecessary. This set contains two conditions, which when satisfied, we can terminate the algorithm, and take that current value as the final answer. Let's take a box X^I. The conditions applied on this box are:

Condition 1: This condition is on the width of interval X. We can define a number ε_X such that the algorithm stops when the width is either less than or equal to this number.

$$w(X^I) \leq \varepsilon_X \tag{3.3}$$

Condition 2: This condition is on the width of interval function $f(X^I)$ itself. Once again we define a number ε_f such that the algorithm stops when the width is either equal to or less than this number.

$$w[f(X^I)] \leq \varepsilon_f \tag{3.4}$$

that is $\overline{f}(X^I) - \underline{f}(X^I) \leq \varepsilon_f$. Condition (3.4) guarantees that the global minimum f^* of the objective function is bounded to within the tolerance ε_f.

3.1.6 Bounds on Minimum

The algorithm has both lower and upper bounds on the global minimum f^*. After termination, global minimum must lie within the solution box. Let's say that total of s boxes remain in the end, denoted by X^{I^i} $(i = 1, ..., s)$.

The algorithm evaluates f for each i, denoted by

$$f(X^{I^{(i)}}) = [\underline{f}(X^{I^{(i)}}), \overline{f}(X^{I^{(i)}})]$$

Denote

$$\underline{F} = \min_{1 \leq i \leq s} \underline{f}(X^{I^{(i)}})$$

We evaluate f at center of each box, find the upper bound $\overline{\mathrm{f}}$ of f^*, and delete a box if $\underline{f}(X^{I^{(i)}}) \geq \overline{\mathrm{f}}$. Therefore,

$$\underline{f}(X^{I^{(i)}}) \leq \overline{\mathrm{f}} \leq \overline{f}(X^{I^{(i)}}) \tag{3.5}$$

for all $i = 1, ..., s$.

Since the global minimum is within one of the final boxes,

$$\underline{F} \leq f^* \tag{3.6}$$

Let j be an index such that $underline f(X^{I^{(j)}}) = \underline{F}$. Letting $i = j$ in 3.5, we get

$$f^* \leq \underline{f}(X^{I^{(i)}}) \leq \overline{f}(X^{I^{(j)}}) \tag{3.7}$$

From the termination condition 3.4,

$$\overline{f}(X^{I^{(j)}}) - \underline{f}(X^{I^{(j)}}) \leq \varepsilon_f \tag{3.8}$$

From (3.6),(3.7) and (3.8),

$$\underline{F} \leq f^* \leq \underline{F} + \varepsilon_f \tag{3.9}$$

Hence, global minimum is bounded within ε_f. We can conclude that

$$f^* \leq \overline{\mathrm{f}} \leq f^* + \varepsilon_f \tag{3.10}$$

3.1.7 List of Boxes

The algorithm we discussed began with only a single box $X^{I^{(0)}}$. This is generally the case because of it's simplicity. But if the user wants, she can declare more than one box to start with. Now, if this is the case, then we put the initial boxes in a list L_1.

As the algorithm proceeds, $X^{I^{(0)}}$ gets divided into subboxes. Then, any subbox X^I satisfying the following criteria, will be put in list L_2.

$$w[X^I] \leq \varepsilon_X \tag{3.11}$$

$$w[f(X^I)] \leq \varepsilon_f \tag{3.12}$$

3.1.8 Choosing a Box to Proceed

We have seen how to declare initial box(es). Now, whenever a new box is put in list, a cycle of main algorithm begins. It has to decide which box to choose.

Before the choice is made, algorithm evaluates $f(X^I)$ for every box X^I in L_1. Let the result be $[\underline{f}(X^I), \overline{f}(X^I)]$. The box with the smallest $\underline{f}(X^I)$ is chosen. This method is more likely to choose the box containing the point X^* of global minimum than randomly choosing the box.

3.2 Algorithm Used

The algorithm used in this work is a simplified variation of the Box and Bound technique explained in detail above. The steps involved are similar to the main technique, with some differences. The steps involved are mentioned below:

1. Begin with square box X in which global optimization is sought. Define a terminating criteria, i.e. some ε that is the width of the box X.

2. Calculate $y = f(X)$ for the whole box. If the resultant, which is also a box, has width less than ε, then terminate the process. Else, split the box X into Z that has four equal parts, namely $[z_1, z_2, z_3, z_4]$.

3. Now, calculate the function $y = f(Z)$ for all the parts of Z. Store the values in a list $L_1 = [y_1, y_2, y_3, y_4]$. Make a copy L_2 of list L_1.

4. Now, begin the main loop with condition on width.

   (a) Find the position of the minimum value of L_1. Select the value of Z at this corresponding position.

(b) Find $y = f(Z)$ for all parts of Z. Store the solution in the updated list L_1. Plot Z.

(c) Calculate the width of all the solutions in L_1.

(d) If the width of all elements of Z are less than ε, then terminate the process. Else, goto $4(a)$.

5. Come out of the loop when process terminates. Get the final values of L_1 and Z. Also keep the final plot. This is the solution.

Since the solutions are stored in a list, all the solutions will be taken into account and stored simultaneously. The algorithm is applicable for all types of optimization problems i.e. single and multiple objective problems.

The splitting technique was mentioned by E.R.Hansen in [12] for one-dimensional case, and further used for multi-dimensional case in [16]. The version used here is inspired from these but with a slight modification. Hansen suggested that the box be split into two, and the one with the larger width be used for further processing. In this text, the box is split into four equal parts and then further processing is done.

Chapter 4

Results and Discussion

This chapter deals with the working of INTLAB. In here is explained how some of the basic functions are performed in INTLAB toolbox, to get acquainted with the INTLAB toolbox. Then we move on to show some results obtained through the interval optimization algorithm.

Example 1:

$$\phi = 3\pi/180$$

$$Q = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}, \; X = \begin{pmatrix} [1,2] \\ [2,4] \end{pmatrix}$$

$$Y = Q \times X = \begin{bmatrix} -1.1340 & 0.7321 \\ 2.2320 & 4.4642 \end{bmatrix}$$

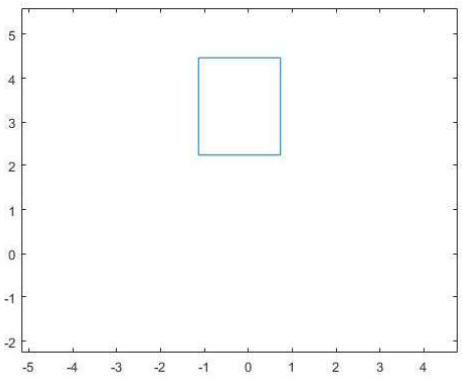

Figure 4.1: Plot for Example 1

The Fig. 4.1 depicts the interval result of the example.

Example 2:

$$\phi = 3\pi/180$$

$$Q = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}, \; X = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 2 & 4 & 2 & 4 \end{bmatrix}$$

$$Y = Q \times X = \begin{bmatrix} -0.1340 & -1.1340 & 0.7321 & -0.2679 \\ 2.2331 & 3.9641 & 2.7331 & 4.4641 \end{bmatrix}$$

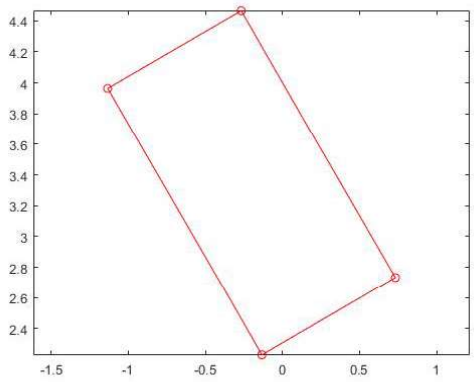

Figure 4.2: Plot for Example 2

Fig. 4.2 shows the true range of the solution.

Example 3:

$$A = \begin{pmatrix} [2,4] & [-1,1] \\ [-1,1] & [2,4] \end{pmatrix}$$

$$B = \begin{pmatrix} [-3,3] \\ [0.8,0.8] \end{pmatrix}$$

Then the plot of A with b is given in Fig. 4.3.

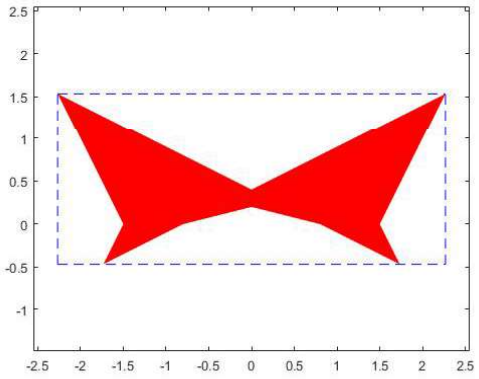

Figure 4.3: Plot for Example 3

Example 4: Given an equation $AX = B$, we need to find the solution for this equation.

$$A = \begin{pmatrix} [1,3] & [-1,2] \\ [-1,0] & [2,4] \end{pmatrix}$$
$$B = \begin{pmatrix} [-2,2] \\ [-2,2] \end{pmatrix}$$

The plot for solution for this is given in Fig. 4.4.

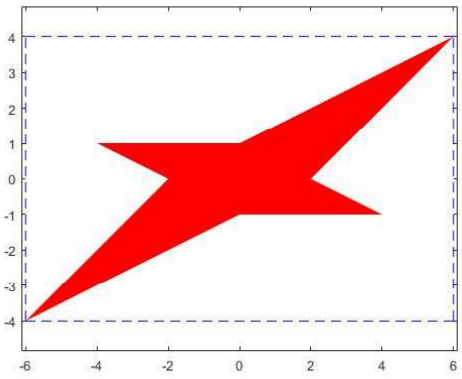

Figure 4.4: Plot for Example 4

Now that we have acquainted ourselves with the basic working of INTLAB, we move on to show the working of the algorithm. It may be noted that although the algorithm is general and applicable for all the cases, the code developed is able to solve only a few test cases.

Function 1: We need to find the global optimum for the function

$$y = X^2$$

The solution for function 1 is shown in Fig. 4.5

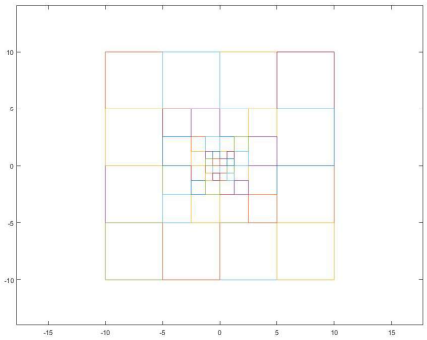

Figure 4.5: Plot for $y = X^2$

Function 2: We need to find the global optimum for the function

$$y = X^2 - 2$$

The solution for function 2 is shown in Fig. 4.6

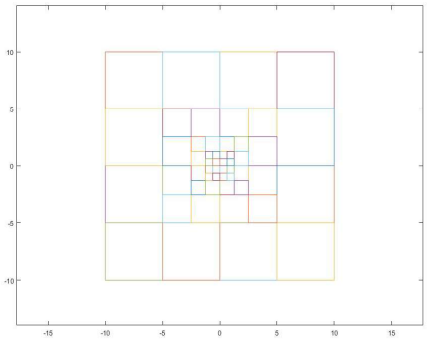

Figure 4.6: Plot for $y = X^2 - 2$

Function 3: We need to find the global optimum for the function

$$y = 3X^2 - 2X + 1$$

The solution for function 3 is shown in Fig. 4.7

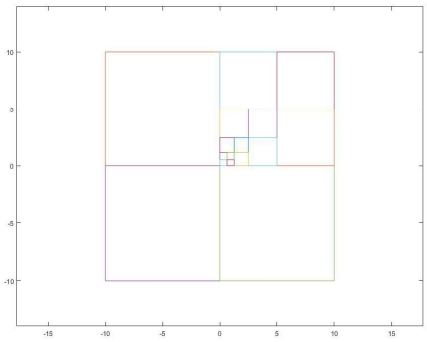

Figure 4.7: Plot for $y = 3X^2 - 2X + 1$

Function 4: We need to find the global optimum for the function

$$y = (X - 1)^2$$

The solution for function 4 is shown in Fig. 4.8

Function 5: We need to find the global optimum for the function

$$y = (X - 2)^2$$

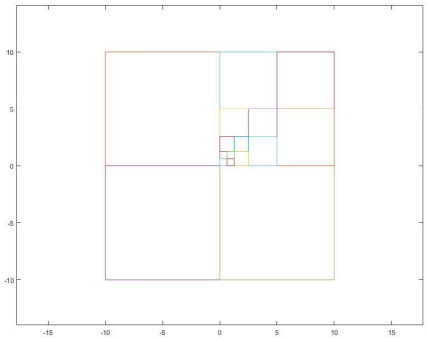

Figure 4.8: Plot for $y = (X - 1)^2$

The solution for function 5 is shown in Fig. 4.9

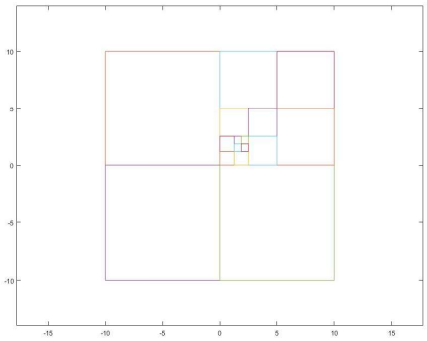

Figure 4.9: Plot for $y = (X - 2)^2$

Chapter 5

Scope for Further Development

As mentioned earlier, the code in its present form is able to solve only some functions of single variable. The algorithm is a general one, and can be extended to multiple variable functions. For solving a multi-variable function, both single and multiple objective, the initial box X has to be defined for multiple variable. Also, all the codes written can be directly used, with slight modifications as required. Then for multi-variable, we can go on solving one objective at a time.

There are a number of applications where Interval Optimization can be applied. This technique can prove quite useful in self drive vehicles. Since the box and bound technique works on rejection principle, it should be able to analyse the surroundings and reject the unfavourable paths, and hence find the optimal path. On similar grounds, it can be used to drive the robots in extremely dangerous environments where human presence is to be avoided.

References

[1] K. Deb. Multi-objective optimization using evolutionary algorithms. Wiley, 2005.

[2] R. E. Moore, R. B. Kearfott, and M. J. Cloud. Introduction to interval analysis. SIAM, 2009.

[3] E. Hansen and G. W. Walster. Global optimization using interval analysis: revised and expanded, volume 264. CRC Press, 2003.

[4] S. Rump. INTLAB - INTerval LABoratory. In T. Csendes, ed., Developments in Reliable Computing, 77–104. Kluwer Academic Publishers, Dordrecht, 1999.

[5] G. I. Hargreaves. Interval analysis in MATLAB .

[6] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation* 2, (1994) 221–248.

[7] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, and A. Fast. Nsga-ii. *IEEE transactions on evolutionary computation* 6, (2002) 182–197.

[8] A. Seshadri. A fast elitist multiobjective genetic algorithm: NSGA-II, Mathlab Central, file exchange, mathworks .

[9] G. Alefeld and G. Mayer. Interval analysis: theory and applications. *Journal of computational and applied mathematics* 121, (2000) 421–464.

[10] S. Skelboe. Computation of rational interval functions. *BIT Numerical Mathematics* 14, (1974) 87–95.

[11] H. Ratschek and J. Rokne. Interval tools for global optimization. *Computers & Mathematics with Applications* 21, (1991) 41–50.

[12] E. R. Hansen. Global optimization using interval analysis: the one-dimensional case. *Journal of Optimization Theory and Applications* 29, (1979) 331–344.

[13] E. R. Hansen. On solving systems of equations using interval arithmetic. *Mathematics of Computation* 22, (1968) 374–384.

[14] R. B. Kearfott. An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization* 2, (1992) 259–280.

[15] P. Nataraj and J. Barve. Reliable and accurate algorithm to compute the limit cycle locus for uncertain nonlinear systems. *IEE Proceedings-Control Theory and Applications* 150, (2003) 457–466.

[16] E. Hansen. Global optimization using interval analysis—the multi-dimensional case. *Numerische Mathematik* 34, (1980) 247–270.