NOVEL VERTEX ENUMERATION ALGORITHMS WITH MECHANICAL ENGINEERING APPLICATIONS

by

Ahmet Ağaoğlu

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Mechanical Engineering

Yeditepe University

2023

NOVEL VERTEX ENUMERATION ALGORITHMS WITH MECHANICAL ENGINEERING APPLICATIONS

APPROVED BY:

Assist. Prof. Dr. Namık Çıplak ....
(Thesis Supervisor)
(Yeditepe University)

Assoc. Prof. Dr. Nezih Topaloğlu ....
(Yeditepe University)

Assist. Prof. Dr. Ali Fethi Okyar ....
(Yeditepe University)

Prof. Dr. Şahin Uyaver ....
(Marmara University)

Assist. Prof. Dr. Murat Tümer ....
(Turkish-German University)

DATE OF APPROVAL: ..... / ..... / 20.....

I hereby declare that this thesis is my own work and that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I have fully cited and referenced all material and results as required by these rules and conduct, and this thesis study does not contain any plagiarism. If any material used in the thesis requires copyright, the necessary permissions have been obtained. No material from this thesis has been used for the award of another degree.

I accept all kinds of legal liability that may arise in case contrary to these situations.

Name, Last Name Ahmet Ağaoğlu

Signature ....

ABSTRACT

NOVEL VERTEX ENUMERATION ALGORITHMS WITH MECHANICAL ENGINEERING APPLICATIONS

This thesis presents the development of three novel vertex enumeration algorithms and their engineering applications. The first algorithm, named the Corridor Method, is introduced as an efficient geometric method for vertex enumeration in simple linear regression problems with bounded error. The method uses bijective maps between lines in measurement space and points in parameter space and shows significantly better performance compared to well-known algorithms in all simulations. The Corridor Method is then extended to cover generic linear optimization problems with two variables, resulting in the Modified Corridor Method. Additionally, a third algorithm, the Edge Tracking Algorithm (ETA), is developed to be used in multidimensional linear problems. ETA tracks the edges of the solution polytope starting from a given vertex to explore the adjacent vertices of the given vertex and shows better performance than well-known algorithms for some of the problems. The thesis then demonstrates the use of these algorithms in mechanical engineering applications. In the regression application, a new regression tool based on the solution polytope obtained by the algorithms is proposed . This tool is compared to other regression techniques, and found to be superior in all experiments by large margins. In the second application, the resulting polytope of the algorithms is used to investigate the feasibility of the assembly. Also, the expected and maximum possible misalignment concepts are discussed based on the resulting polytopes using Monte Carlo simulations.

ÖZET

YENİ KÖŞE SAYMA ALGORİTMALARI VE MAKİNE MÜHENDİSLİĞİNDEKİ UYGULAMALARI

Bu tez, üç yeni köşe sayma algoritmasının geliştirilmesini ve bunların mühendislik uygulamalarını sunmaktadır. Koridor Yöntemi olarak adlandırılan ilk algoritma, limitleri belirli hata içeren basit doğrusal regresyon problemlerinde köşe sayımı için geliştirilmiş geometrik bir yöntemdir . Bu metot, ölçüm uzayındaki doğrular ile parametre uzayındaki noktalar arasında bijektif haritalar kullanır. Yapılan tüm simülasyonlarda iyi bilinen algoritmalara kıyasla daha iyi performans göstermiştir. Devamında Koridor Metodu, iki değişkenli genel doğrusal optimizasyon problemlerini kapsayacak şekilde genişletilmiş ve böylece sunulan ikinci algoritma olan Geliştirilmiş Koridor Metodu ortaya çıkmıştır. Ek olarak, çok boyutlu doğrusal optimizasyon problemlerinde kullanılmak üzere üçüncü bir algoritma olan Kenar İzleme Algoritması (ETA) geliştirilmiştir. ETA, verilen bir köşe noktasından başlayarak çözüm politopunun kenarlarını takip eder ve komşu köşe noktalarını keşfeder. Bazı problemler için bilinen diğer algoritmalardan daha iyi performans gösterdiği saptanmıştır. Sonrasında sunulan algoritmaların makine mühendisliği uygulamalarında kullanımını gösterilmektedir. Regresyon uygulamasında, sunulan algoritmalar tarafından elde edilen çözüm politopuna dayalı yeni bir regresyon tekniği önerilmiştir. Bu teknik, diğer regresyon teknikleriyle karşılaştırılmış ve üstünlüğü deneylerle kanıtlanmıştır. İkinci uygulamada ise sunulan algoritmalarla elde edilen çözüm politopu, bir mekanik montajın uygulanabilirliğinin tespiti için kullanılmıştır. Ayrıca, Monte Carlo simülasyonları ile montajın beklenen ve maksimum olası yanlış hizalama kavramları tartışılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

d	Parameter space dimension
D	Set of extreme rays
d_1, d_2	Extreme rays of R_L
e_{ave}, e_2	Expected misalignment
e_{max}, e_2	Maximum possible misalignment
e_1, e_2	Extreme rays of R_U
H	Hyperplane
h	Line segment
h_L	Set of lower bound line segments
h_U	Set of upper bound line segments
L	Lower bound points
l_{min}	Minimum slope line
l_{max}	Maximum slope line
$\mathbb{M}$	Measurement space
m_{max}	Maximum slope
m_{min}	Minimum slope
N	Number of observations
n_{max}	Maximum intercept
n_{min}	Minimum intercept
N_L	Number of lower bound points
N_U	Number of upper bound points
P	Pivot point
$\mathbb{P}$	Parameter space
Q	Contact point
R_L	Set of lines that are above the lower bound
R_U	Set of lines that are below the upper bound
∂R_L	Boundary set of R_L
∂R_U	Boundary set of R_U
S	Full solution set

S_F	Set of free lines
S_L	Set of lower bound lines
S_P	Final solution polytope
S_U	Set of upper bound lines
S_{UL}	Set of the minimum and maximum slope lines
∂S	Boundary set of S
T_c	Computational time
U	Upper bound points
v	Total number of vertices
V	Set of vertices
V_L	Set of lower bound vertices
V_U	Set of upper bound vertices
x	Predictor variable
y	Response variable
β	Model parameter vector
ε	Error
BENSOLVE	A tool for vector linear programming
CM	Corridor method
DDM	Double description method
ETA	Edge tracking algorithm
FC	Functional constraint
Inf	Possible largest number of the computation platform
lrs	Reverse search vertex enumeration algorithm
MCM	Modified corridor method
MLE	Maximum likelihood estimation
OLS	Ordinary least square
UBBE	Unknown but bounded error case

1. INTRODUCTION

Vertex enumeration problems refer to a class of mathematical optimization problems in which the objective is to determine the vertices of a polytope. Polytopes are geometric objects formed by the intersection of finitely many half-spaces, and their vertices represent the points at which these half-spaces intersect. The study of vertex enumeration problems has a long history, dating back to the early works on linear programming.

Over the years, vertex enumeration problems have been applied to a wide range of fields, including operations research, computer graphics, network flow analysis, and more. Despite the practical importance of these problems, they are computationally challenging; thus, finding efficient algorithms for solving them remains an active area of research.

In this thesis, we aim to address several important aspects of vertex enumeration problems, including the design of efficient algorithms, the analysis of their computational complexity, and the development of new algorithmic techniques for solving these problems. By examining these problems from different angles, we hope to deepen our understanding of the underlying mathematical structure and develop new tools for solving these problems in practice.

This chapter provides a roadmap for the rest of the thesis, highlighting the main contributions and goals of each subsequent chapter.

In various fields such as regression, state-space and time series prediction, parameter and state estimation of dynamic systems, and so on, the objective is to accurately estimate unknown model parameters using observed values of model variables. Nevertheless, measurements inherently possess some degree of uncertainty, which requires appropriate consideration of uncertainty assumptions for solving the problem.

The uncertainty in real-world experiments is mainly related to the accuracy of the sensing device (sensor). Sensor accuracy is typically quantified in terms of additive or multiplicative forms, each of which provides an interval within which the true value is expected to fall.

For example, if a sensor has an additive accuracy of ±2 °C and provides a reading of 35 °C, the actual value could range from 33 °C to 37 °C. Similarly, if the sensor has a multiplicative accuracy of ±2%, the true value range would be 35×(1-0.02)=34.3 °C to 35×(1+0.02)=35.7 °C.

This particular scenario, where only the error bounds are available and no other information is present, is known as unknown but bounded noise (UBBE), see [2].

In the case of multiple observations, an uncertainty corridor that is characterized by a lower and an upper bound within which the true value is expected to fall is established. In Chapter 3, the simplest case, namely the simple linear regression model, is examined using this approach, in which the mathematical model corresponds to a line.

Any line in the uncertainty corridor has the possibility of being the true line. Therefore, the next step involves seeking a solution method to identify all possible lines within the corridor.

Each line within the uncertainty corridor represents a unique point in the parameter space, which is defined by the slope and intercept dimensions. The collection of all lines in the corridor defines a solution region in the parameter space. Therefore, the solution is a set S comprising all possible model parameter vectors that meet the error bounds and the model requirements.

Given that each interval has two inequalities, one from the lower and the other from the upper bounds, S is formed by a set of $2N$ linear inequalities, where N represents the number of observations. This renders the problem a linear programming one, which seeks to determine the full solution set, given the $2N$ linear constraints.

In the case of models that are linear with respect to their parameters, the set S is a convex polygon (2-polytope) in R^d, where d denotes the number of parameters, i.e., the dimensionality of the model parameter space. As the polygon can be defined by its vertices, the problem can be reformulated as a vertex enumeration problem.

In Chapter 3, a novel and fast algorithm (the Corridor Method, or CM) is developed to enumerate all of the vertices with no duplicates. CM is based on bijective maps between lines in the measurement space and points in the parameter space. The worst-case time complexity of CM is determined as $O(N^2)$. However, the median-case complexity of CM appears to be $O(N)$.

Furthermore, CM is compared to two established vertex enumeration algorithms using synthetically generated corridors. Given that all three methods precisely describe S by its vertices, their computational efficiency is compared for various scenarios, in which CM significantly outperforms both.

Although CM is an efficient algorithm, its application is limited to problems with UBBE. However, most linear programming problems involve a set of unrelated individual constraints, which may not have an equal number of lower and upper bound inequalities, or may lack an inequality on one side altogether. To address this, a modified version of CM, the Modified Corridor Method (MCM), is developed in Chapter 4.

MCM is based on the concept of splitting the corridor into two sections, the lower and upper bounds, to obtain individual constraints. By doing so, the solution region is the intersection of two sets, consisting of the lines that remain above the lower bound and below the upper bound. These sets can be obtained by implementing CM with minor adjustments. Subsequently, MCM employs a method of efficiently intersecting both sets that is developed by examining various intersection scenarios.

In this way, MCM is adapted to general linear optimization problems, however, it is not capable of generating solutions for multidimensional spaces, namely $d > 2$. To address this limitation, a new algorithm is developed in Chapter 5. This algorithm is based on the principle of determining adjacent vertices of a solution polytope by following the edges connected to a given vertex.

In a simple polytope, each vertex is at the intersection of d hyperplanes in the parameter space, where each hyperplane corresponds to a constraint in the problem. The intersection of each $d-1$ combination of these d hyperplanes forms an edge of the polytope. By identifying the edges connected to a vertex and intersecting each of them with hyperplanes that do not belong to the original vertex, the other vertices at the other ends of the edges can be found. The algorithm continues until no new vertices can be obtained.

This method is called the Edge Tracking Algorithm (ETA) as it tracks the edges of the polytope to find the adjacent vertices. ETA has been compared to two algorithms that were competed with CM, and its advantages have been evaluated.

Chapter 6 explores the application of ETA to mechanical engineering problems through the use of solution polytopes. The first specific application addressed in this chapter is a regression application.

In the context of UBBE, each measurement in simple or multiple linear regression models can be associated with an upper and a lower bound. For simplicity, we will consider the simple linear model. The solution set for this model consists of all the lines in the uncertainty corridor, and the image of the solution set in the parameter space is a polygon.

Of all the lines in the solution set, only one represents the true line, and the objective is to identify this line or a line that is relatively close to it. Referring back to the polygon in the parameter space, the corresponding line is reduced to a single point. However, it is unclear which point corresponds to the true line, since the nature of the errors is unknown. As a result, all points within the polygon have an equal likelihood of representing the true line.

Several experiments with synthetic data, where the true line is known, demonstrate that the points located near the center of the polygon are more likely to be closer to the true model parameters than those located near the boundary. Consequently, a regression tool is suggested, which utilizes the centroid as the optimal estimate. When comparing the performance of this method with other regression techniques using both synthetic and real experimental data, it is apparent that the proposed technique outperforms the alternatives in all experiments.

In the next application, the use of solution polytopes as a means of testing the feasibility of a mechanical assembly, using a flange assembly consisting of two parts aligned with each other via pins and holes as an example, is explored. We examine the feasibility of the assembly by reducing the mate relations between parts to geometric constraints, taking into account the deviations in part dimensions that may arise due to production tolerances. To ensure that the assembly can be done, one checks whether all constraints satisfy , in which case the solution polytope will not be an empty set.

Further, the solution polytope provides valuable information about possible misalignment. Additionally, a functional constraint, such as a maximum misalignment amount, can be represented as a geometric object in the parameter space where the solution polytope must be included to satisfy this constraint.

Subsequently, Monte Carlo simulations can be used to investigate the effects of different production tolerances on the assembly. These simulations provide a statistical approach to evaluating the feasibility of the assembly and informing about possible misalignment under various tolerance conditions, allowing for more comprehensive analysis of the assembly process.

Overall, these methods provide a robust framework that can have significant practical applications in widely different fields, including engineering and manufacturing.

2. LITERATURE SURVEY

Vertex enumeration is a fundamental problem in many fields, including linear programming, integer programming, optimization, and computer graphics. Vertex enumeration algorithms are a class of computational geometry algorithms that aim to find all the vertices of a polytope.

The first algorithms date back to the 1950s and 60s and were based on the simplex method [3] for linear programming. The primary objective of the first studies was to accurately find the complete set of vertices ([4], [5], and [6]). These early algorithms were limited in their applicability and scalability and had a high time complexity.

Since the 1980s, with the advent of modern computing techniques, the objective has shifted more to enhancing the computational efficiency of existing algorithms or finding faster ones ([7], [8], [9], [10], [11]).

To this end, numerous methods have been proposed for exact description of the solution set as a vertex enumeration problem. In 1980s, Matheiss [12] and Dyer [13] presented surveys on vertex enumeration algorithms. Later in 1997, Avis [14] presented a comparative analysis of the performances of various algorithms. All three papers indicate that there are two types of vertex enumeration algorithms in the literature.

The first type uses pivoting techniques ([15], [16], [17], [18], [7], [19], [9], [20]). Non-pivoting algorithms of the second kind are based on the double description method (DDM) introduced by [6] that intersects a given polytope with one hyperplane at a time ([21], [22], [23], and [24]). Since non-pivoting algorithms can be used recursively to obtain the final polytope S, they are ideal for problems in real-time applications where measurements are taken sequentially in time. Moreover, DDM is considered to be the most efficient algorithm for finding all vertices of a polytope.

The main problem with the exact description methods is their computational complexity, especially when the number of observations or dimensions of the parameter space, or both, are large. In order to circumvent this problem, alternative (non-exact) methods are proposed that approximate S in the parameter space using simpler geometries such as outer bounding ellipsoids, orthotopes, zonotopes, and parallelotopes ([25], [26], [27], [28], [29],

[30], [31], [32], [33], [34], [35], [36]). These methods consist of geometric or analytical approaches based on the minimization of the size of the simpler geometries, or convergence considerations. However, these methods often provide a loose approximation to the exact polytopic region.

In conclusion, vertex enumeration algorithms have a long history and have been widely used in a variety of fields. The development of new algorithms and techniques has made it possible to solve problems that were previously thought to be intractable, and the field is constantly evolving to meet new challenges.

3. CORRIDOR METHOD

Consider the following simple linear regression model.

$$y_i = \beta_0^* + \beta_1^* x_i + \varepsilon_i \tag{3.1}$$

where y_i and x_i are values of response and predictor variables, respectively, measured in the i^{th} observation ($i = 1, \cdots, N$). $\beta_0^*, \beta_1^* \in R$ are the true model parameters to be estimated and $\varepsilon_i \in \left[\varepsilon_i^L, \varepsilon_i^U\right]$ denotes the unknown, but bounded error. Therefore, for each observation the true response is constrained by two inequalities.

$$\beta_0^* + \beta_1^* x_i \geq y_i - \varepsilon_i^U = L_i \tag{3.2}$$

$$\beta_0^* + \beta_1^* x_i \leq y_i - \varepsilon_i^L = U_i \tag{3.3}$$

where L_i and U_i represent the lower and upper limits of the true response.

If multiple observations are considered, then L_i and U_i form an uncertainty corridor as shown in Figure 3.1. Further, $\beta_0^* + \beta_1^* x_i$ now represents the true line which must be in the corridor, assuming that error bounds and the measurements conform to the model. In this way, the problem becomes a membership problem of finding full solution set containing all feasible pairs of (β_0, β_1) given by:

$$L_i \leq \beta_0 + \beta_0 x_i \leq U_i \quad \text{for all } i \tag{3.4}$$

Assumptions:

1. $1 < N < \infty$
2. $x_i \neq x_j$ for $i \neq j$
3. $\varepsilon_i^L, \varepsilon_i^U \neq 0$.
4. The image of S in $\mathbb{P}$ is a non-empty, 2D region.

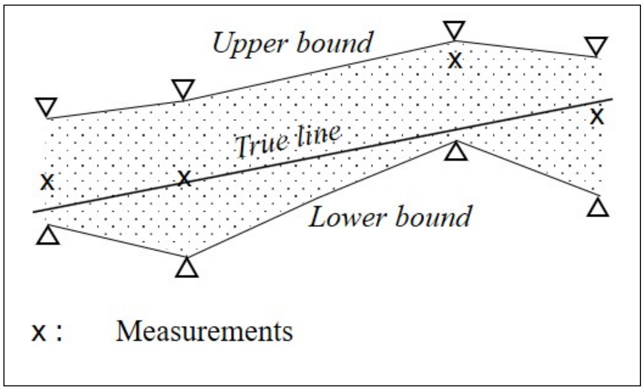

Figure 3.1. The uncertainty corridor

These assumptions and Equation 3.4 force β_0 and β_1 to be bounded. Further, it is well known that the solution set for a system of linear inequalities is always a convex polygonal region. Hence, the following lemma.

Lemma 1. *S is a bounded, convex polygonal region in parameter space, namely space of β_0 and β_1.*

3.1. GEOMETRY OF THE FULL SOLUTION SET

Let the space of (x, y) be denoted by $\mathbb{M}$ (measurement space) and that of (β_0, β_1) by $\mathbb{P}$ (parameter space), both being $\mathbb{R}^2$ spaces. Then, a linear map between $\mathbb{M}$ and $\mathbb{P}$

$$y = \beta_0 + \beta_1 x \tag{3.5}$$

can be interpreted in two ways that are duals of one another:

1. Given (β_0, β_1) in $\mathbb{P}$, Equation 3.5 describes a line in $\mathbb{M}$.
2. Given (x, y) in $\mathbb{M}$, it describes a line in $\mathbb{P}$ ($\beta_0 = y - x\beta_1$).

This correspondence indicates that all lines in S are mapped to a bounded set, S^*, in $\mathbb{P}$.

In this paper, we are interested in translations and rotations of lines in the uncertainty corridor. If a line is moved (translated or rotated) the corresponding point in $\mathbb{P}$ will also move, describing a curve in $\mathbb{P}$.

Consider continuous rotations of a line about a fixed point (x_P, y_P) in $\mathbb{M}$. The resulting set of lines is called a pencil of lines, or a subset of it (partial pencil), with concurrent point (x_P, y_P). This leads to $\beta_0 = y_P - x_P\beta_1$, a line in $\mathbb{P}$. Figure 3.2 illustrates the mapping of a generic partial pencil of lines.

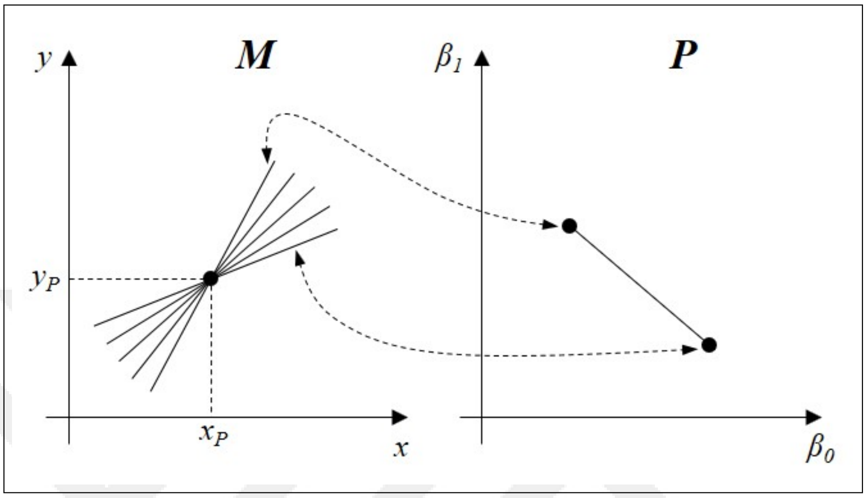

Figure 3.2. A constrained rotation of lines in $\mathbb{M}$ and its image in $\mathbb{P}$.

In this study, it is aimed to fully and efficiently describe S and S^*. Owing to the equivalence of S and S^*, from here on S is used to refer to both sets.

Definition 1. *Define the following subsets of S:*

1. S_F : *The set of all lines through the corridor that do not pass through any lower or upper bound points (free lines).*
2. S_L : *The set of all lines through the corridor that pass through at least one lower, but none of the upper bound points.*
3. S_U : *The set of all lines through the corridor that pass through at least one upper, but none of the lower bound points.*
4. S_{UL} : *The set of all lines through the corridor that pass through at least one lower and at least one upper bound points.*

Theorem 1. *The subsets S_F, S_L, S_U, and S_{UL} form a partition of S.*

Proof. Since all lines are accounted for, one has $S = S_F \cup S_L \cup S_U \cup S_{UL}$. Then the fact that $S_i \cap S_j = \emptyset$ (mutually disjoint subsets) proves the claim. □

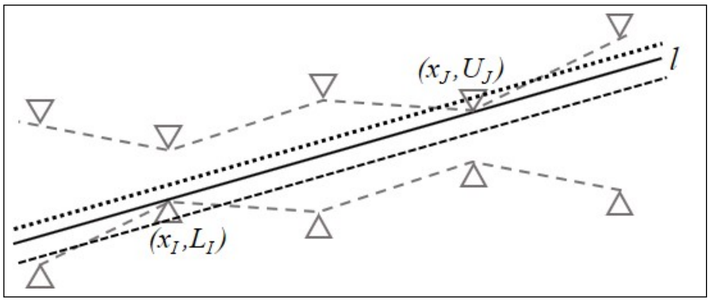

Figure 3.3. Uniqueness of slopes of lines in S_{UL}.

Theorem 2. *S_{UL} is a finite set.*

Proof. This follows from the definition of S_{UL} and the assumption $N < \infty$. □

Corollary 1. *S_F is an open set. The set of all limit points of S_F, $\partial S = S_L \cup S_U \cup S_{UL}$, is the boundary set of S_F. The closure of S_F is $S = S_F \cup \partial S$. Hence, S_F is the interior of S.*

Proof. For any line S_F an appropriate shift downward (upward) would eventually touch a lower (an upper) point yielding a line in S_L (S_U). Similarly, one can show that appropriate and continuous rotations of lines in S_F end up with lines in S_{UL}. Note that these transformations can be regarded as infinite sequences that start in one set yet end, in the limit, in another that is disjoint. In set theoretical terms, the end points are the limit (accumulation) points of the original sets. Since S_F does not contain any of its limit points it is an open set. S_L, S_U, and S_{UL} contain the limit points of S_F. Therefore, their union forms the boundary of S_F. □

Theorem 3. *A line $l \in S_{UL}$ if and only if l has a unique slope among all other lines in S.*

Proof. Let l pass through (x_I, L_I) and (x_J, U_J). Another line with the same slope but greater intercept would violate (x_J, U_J). Similarly, a parallel line with a lesser intercept would violate (x_I, L_I) (see Figure 3.3). Hence, there is no other line in S with the same slope. Conversely, consider $l \in S_L$ (S_U, S_F) any appropriate parallel shift, however small, will still be feasible. Therefore, the slopes of lines in S_L (S_U, S_F) are not unique. Hence, if a line has a unique slope it must belong to S_{UL}. □

Theorem 4. *$S_{UL} = \{l_{\min}, l_{\max}\}$, where $l_{\min}$ and $l_{\max}$ are the lines with the strictly minimum and the strictly maximum slopes in S, respectively.*

Proof. S is a non-empty, bounded, convex polygonal region (Lemma 1) with boundary $\partial S = S_L \cup S_U \cup S_{UL}$ (Corollary 1), a closed, convex polygon. Note that $|S_{UL}| < \infty$ (Theorem 2). For $|S_{UL}| = 0$ or 1, S_L and S_U cease to be disjoint (Theorem 1), see Figure 3.4. For $|S_{UL}| > 2$, at least one element of S_{UL} ceases to have a unique slope (Theorem 3). Therefore, $|S_{UL}| = 2$ is the only viable case that corresponds to the lines with the strictly minimum ($l_{\min}$) and the strictly maximum ($l_{\max}$) slopes, as shown in Figure 3.4. □

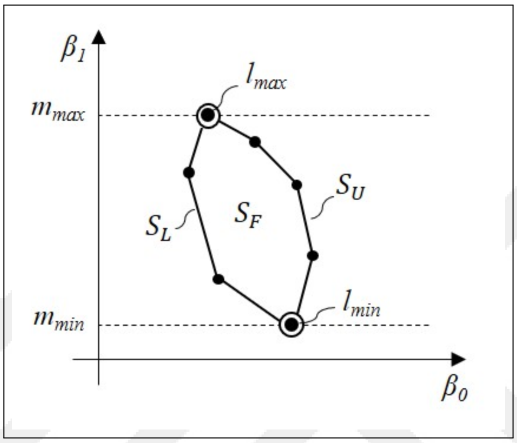

Figure 3.4. Structure of S and partition of ∂S by $l_{\min}$ and $l_{\max}$.

Corollary 2. *$l_{\min}$ and $l_{\max}$ form two vertices of ∂S and partition it into four disjoint components: $\{l_{\min}\}$, $\{l_{\max}\}$, S_L, S_U, the latter two of which are infinite and connected sets (See Figure 3.4).*

The other vertices of ∂S must belong to S_L or S_U. In order to identify all vertices one can employ the following theorem.

Theorem 5. *Any line in the corridor that passes through at least two bound points is a vertex of ∂S.*

Proof. Let $l \in S_L$ pass through two lower bound points, say P and Q. The image of l is a point in $\mathbb{P}$, say Λ. One can rotate l about P by freeing from Q. This yields a line $l' \in S_L$ that passes through only P. The rotation corresponds to a line segment in $\mathbb{P}$, the end point of which is Λ. Now consider the same, but this time by freeing from P and pivoting on Q in the opposite sense, giving $l'' \in S_L$, again a line segment in $\mathbb{P}$ with the common point Λ. However, this time the dependency relation between β_0 and β_1 will be different due to the change in the constraint, i.e. keeping Q fixed instead of P, resulting in distinct line segments

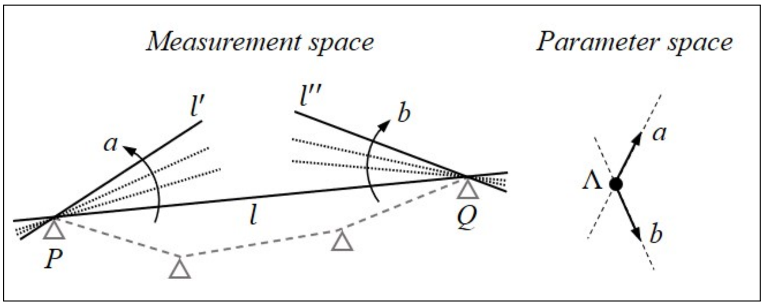

Figure 3.5. Formation of a vertex in $\mathbb{P}$ by a line in $\mathbb{M}$ contacting two points in the corridor.

in $\mathbb{P}$. Therefore, l represents the join of two line segments in $\mathbb{P}$, a *vertex* Λ (Figure 3.5). Similar arguments hold for S_U. Further, if P and Q are considered as a lower point and an upper point, then one can show that $l_{\min}, l_{\max}$ are also vertices as shown before. Therefore, any line in the corridor that passes through at least two bound points is a vertex of ∂S. □

Some of the findings in this section are well known facts. What is important here is the geometrical method employed that forms the basis of an efficient algorithm described next.

3.2. TRACING THE BOUNDARY OF THE FULL SOLUTION SET

Define $V_L = \{l_i\} \subset S_L$ and $V_U = \{u_j\} \subset S_U$ as finite sets composed of the vertices of S, called the lower and upper bound vertices, respectively. The vertices in $S_{UL} = \{l_{\min}, l_{\max}\}$ will be called as the minimum and the maximum slope vertices. Thus, the totality of the vertices is given by

$$V = \{l_{\min}\} \cup \{l_{\max}\} \cup V_L \cup V_U \tag{3.6}$$

Consider $l \in S_L$ that passes through a lower bound point P_0. Image of l corresponds to a point in $\mathbb{P}$, say B. Since the boundary is a polygon, one can trace the boundary by starting from B and returning back to B, in one full cycle. A visual illustration of this cycle is depicted in Figure 3.6.

In order to reach $V_{\min}$, one rotates l by keeping P_0 fixed and decreasing the slope. This transformation traces the image of S_L in $\mathbb{P}$. The rotation continues until either of two things happens: i) the rotated line contacts one or more lower bound points, say P_1, without contacting the upper bound, marking an element of V_L (Theorem 5) or, ii) the line contacts

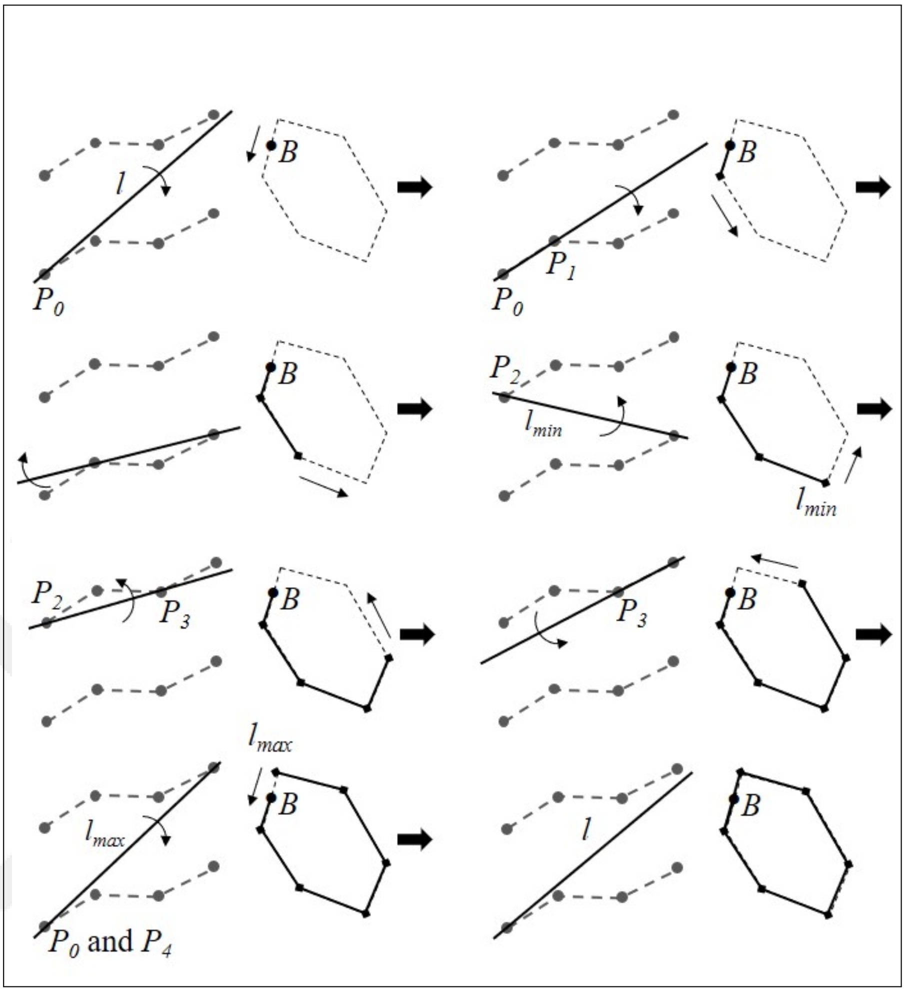

Figure 3.6. Illustration of the cycle to determine the vertices sequentially.

one or more upper bound points, say P_2, giving $l_{\min}$ (since the slope is decreased).

In both cases no further rotation in the same sense is possible. Note that since the slope is decreased, P_1 and P_2 can only be the points succeeding and preceding the pivot point, respectively. If case (i) occurs, then one continues the rotation in the same sense by freeing P_0 and pivoting on P_1. Thus, tracing the image of S_L in $\mathbb{P}$ continues. Again, either $l_{\min}$ is reached (case ii) or a new element of V_L is found (case i). The process continues until $l_{\min}$ is reached eventually.

After $l_{\min}$ is reached, one continues rotation by freeing the lower contact point and pivoting on the upper contact point P_2. Further, since $V_{\min}$ is the point with the minimum slope in $\mathbb{P}$, slope is to be increased. Rotations about P_2 trace the image of S_U in $\mathbb{P}$ that continue until either of two things happens: a) the rotated line contacts one or more upper bound points P_3, without contacting the lower bound, marking an element of V_U or, b) the rotated line contacts one or more lower bound points P_4, giving $l_{\max}$. The rest is similar.

The process is terminated when P_0 becomes the pivot again. In this way, the vertices are determined sequentially as follows.

$$V_L \rightarrow l_{\min} \rightarrow V_U \rightarrow l_{\max} \rightarrow V_L \tag{3.7}$$

In case of multiple new contact points, the pivoting should continue with the furthest one, since they lie on the same line and would result in duplicate vertices.

Note that $l_{\min}$ and $l_{\max}$ are adjacent to the first and the last vertices on S_L and S_U segments. Thus, by employing these vertices directly, one can determine $l_{\min}$ and $l_{\max}$ using a more efficient approach . The following theorem is used to this end.

Theorem 6. *Let* $V_L = \left\{l_1, l_2, \cdots, l_p\right\}$ *and* $V_U = \left\{u_1, u_2, \cdots, u_q\right\}$ *be ordered with respect to* β_1 *values, the former in descending and the latter in ascending order. Also, let* (x_a, L_a) *be the left contact point of* l_1 *and* (x_b, L_b) *be the right contact point of* l_p*. Similarly, let* (x_c, U_c) *be the left contact point of* u_1 *and* (x_d, U_d) *be the right contact point of* u_q*. Then,*

1. $l_{\min}$ *passes through* (x_b, L_b) *of* l_p *and* (x_c, U_c) *of* u_1 *such that* $x_c < x_b$ *(first from upper),*
2. $l_{\max}$ *passes through* (x_a, L_a) *of* l_1 *and* (x_d, U_d) *of* u_q *such that* $x_a < x_d$ *(first from lower).*

Proof. Due to the ordering, $l_{\min}$ is adjacent to and between l_p and u_1. The edge between l_p and $l_{\min}$ in $\mathbb{P}$ is formed by rotations in $\mathbb{M}$ about (x_b, L_b), whereas the edge between u_1 and $l_{\min}$ is formed by rotations about (x_c, U_c) (Figure 3.7). At the intersection of these two edges the rotated lines become coincident, yielding $l_{\min}$. Therefore, $l_{\min}$ has to pass through (x_b, L_b) and (x_c, U_c). The case for $l_{\max}$ is similar. Let $m_{\min} = \frac{U_c - L_b}{x_c - x_b}$ and $m_{\max} = \frac{U_d - L_a}{x_d - x_a}$ be the slopes of $l_{\min}$ and $l_{\max}$, respectively. Since both are finite (Lemma 1), one has $x_c \neq x_b$

and $x_d \neq x_a$. The corridor conditions for $l_{\min}$ at $l_{\max}$ contact points yield

$$m_{\min}(x_a - x_b) + L_b \geq L_a \tag{3.8}$$

$$m_{\min}(x_d - x_b) + L_b \leq U_d \tag{3.9}$$

which, after eliminations, leads to

$$m_{\min}(x_d - x_a) \leq U_d - L_a = m_{\max}(x_d - x_a) \tag{3.10}$$

since $m_{\min} < m_{\max}$, this can be true only if $x_a < x_d$. The case for $x_c < x_b$ is similar. □

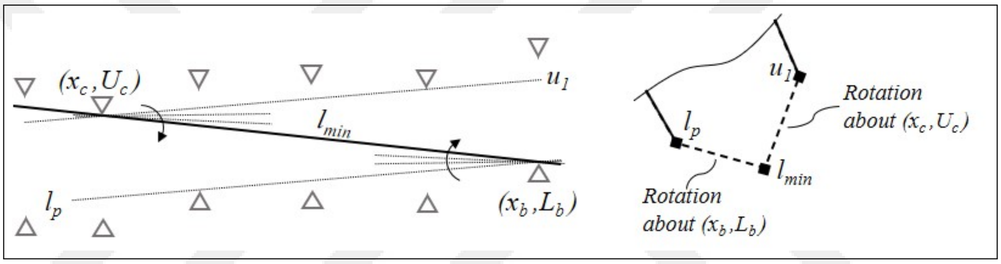

Figure 3.7. Demonstration of the adjacency relations of $l_{\min}$.

Consider the case $V_L = \emptyset$, in which the image of S_L in $\mathbb{P}$ is a single line segment. This case is possible only if $V_U \neq \emptyset$. Had we had $V_U = \emptyset$ simultaneously, the images of S_L and S_U would become identical line segments connecting $l_{\min}$ and $l_{\max}$ in $\mathbb{P}$, giving $S_F = \emptyset$, contradicting Assumption 4. Therefore, V_L and V_U cannot simultaneously be empty sets. In what follows the cases of $V_L = \emptyset$ or $V_U = \emptyset$ are handled separately.

1. If $V_L \neq \emptyset$ and $V_U = \emptyset$, then S_U is a line segment in $\mathbb{P}$ connecting $l_{\min}$ to $l_{\max}$. In this case, $l_{\min}$ and $l_{\max}$ are adjacent and connected by the edge that is formed by a rotation about a common upper bound point (x_w, U_w) (see Figure 3.8). Therefore, $l_{\min}$ passes through (x_b, L_b) and (x_w, U_w) whereas $l_{\max}$ passes through (x_a, L_a) and (x_w, U_w). By Theorem 6, $a < w < b$, which eliminates some upper bound points to be checked and improves the computational efficiency.

2. The case of $V_L = \emptyset$ and $V_U \neq \emptyset$ is handled similarly. Let (x_z, L_z) be the common lower bound point. Therefore, $l_{\min}$ and $l_{\max}$ are formed by (x_z, L_z), (x_c, U_c) and (x_z, L_z), (x_d, U_d), respectively, where $c < z < d$.

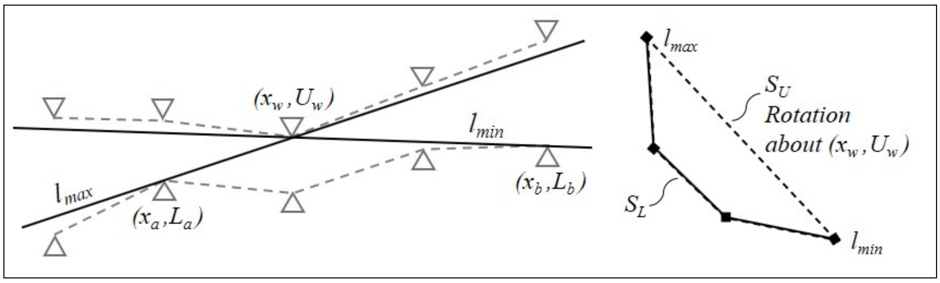

Figure 3.8. Demonstration of the adjacent $l_{\min}$ and $l_{\max}$ vertices case.

3.3. AN ALGORITHM TO ENUMERATE VERTICES OF THE FULL SOLUTION SET

In what follows, we propose an algorithm based on the method outlined in Subsection 3.2. The algorithm expects on input the lower and upper bound data, $L_i\,(x_i)$ and $U_i\,(x_i)$, sorted with respect to $\{x_i\}$ in ascending order. The algorithm first determines vertices in V_L and V_U, then $l_{\min}$ and $l_{\max}$.

The algorithm starts with (x_1, L_1) as the first pivot, say P. In order to find the other contact point, say Q, which would form a vertex together with P, the lines that pass through P and all succeeding points are created. The point Q is determined by the line with the maximum slope, since the other points would stay below this line and so are not feasible (Figure 3.9). If the maximum slope line touches more than one point, excluding P, then the furthest is employed as Q (see Subsection 3.2).

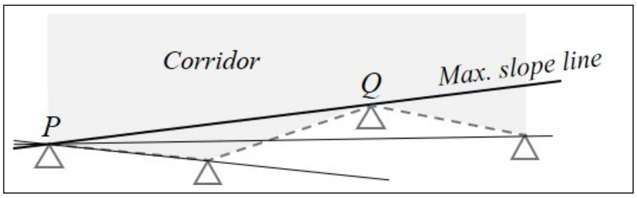

Figure 3.9. Determination of an element of S_L

Subsequently, the line passing through P and Q is tested against the upper bound. If the line is *below* the upper bound then it is a lower bound vertex. In the next iteration, Q becomes the new pivot point regardless of whether a vertex is present in the previous iteration. The same operations continue until the index of the pivot point reaches N.

When a vertex is found, it is recorded with β_0 and β_1 values. Further, indices a and b are also recorded to employ later in determination of $l_{\min}$ and $l_{\max}$. The implementation of this method is presented in Algorithm A.1.

A similar algorithm, with minor modifications, can be written to determine vertices in V_U. This time, the algorithm operates on upper bound points, where Q is determined by the minimum slope line, and checks the feasibility against the lower bound points. The output would be V_U, c, and d.

After V_L and V_U are obtained, $l_{\min}$ and $l_{\max}$ are determined by another algorithm, an implementation of which is given in Algorithm A.2. The slope and intercept of $l_{\min}$ are denoted by $m_{\min}$ and $n_{\min}$, respectively, and those of $l_{\max}$ are denoted by $m_{\max}$ and $n_{\max}$, respectively.

3.4. AN EXAMPLE

The following example has been used as a benchmark in [37]. A set of three observations, sorted with respect to $\{x_i\}$ in ascending order, are given as follows.

$$y_1 = 1.5 = \beta_0 - \frac{1}{2}\beta_1 + \varepsilon_1 \tag{3.11}$$

$$y_2 = 3.5 = \beta_0 + \frac{1}{8}\beta_1 + \varepsilon_2 \tag{3.12}$$

$$y_3 = 3.5 = \beta_0 + \frac{1}{2}\beta_1 + \varepsilon_3 \tag{3.13}$$

where $-0.5 \le \varepsilon_i \le 0.5$ for $i = 1, 2, 3$. Using equations 3.2 and 3.3, one finds the lower and upper bounds: $L = \begin{bmatrix}1 & 3 & 3\end{bmatrix}^T$ and $U = \begin{bmatrix}2 & 4 & 4\end{bmatrix}^T$ with $x = \begin{bmatrix}-\frac{1}{2} & \frac{1}{8} & \frac{1}{2}\end{bmatrix}^T$. Then, V_L is determined as follows.

In the first iteration, the pivot point P is the first lower bound point $\left(-\frac{1}{2}, 1\right)$ and the slope of lines passing through the pivot and succeeding lower bound points are calculated as $m^L_{12} = 3.2$ and $m^L_{13} = 2$. Since m^L_{12} is the maximum, the contact point Q is determined as the second point and the line passing through the first and second lower bound points, $y = 3.2x + 2.6$, is tested against the upper bound, in which it fails due to $y_3 = 4.2 \nleq U_3 = 4$.

Next, the pivot becomes Q, the second lower bound point. In the same manner, $m^L_{23} = 0$ and it is the maximum since there is only one line. The line passing through the second and third lower bound points, $y = 3$, violates the upper bound, since $y_1 = 3 \nleq U_1 = 2$. Then, P becomes the third lower bound point and the procedure is terminated since there are no succeeding points left. Thus, there are no lower bound vertices, $V_L = \emptyset$, $a = \emptyset$, $b = \emptyset$.

In order to determine the elements of V_U, one fixes the first upper bound point $\left(-\frac{1}{2}, 2\right)$ as P and calculates slopes: $m^U_{12} = 3.2$ and $m^U_{13} = 2$. The third point becomes the candidate in this case, Q, since m^U_{13} is the minimum. The line passing through the first and third upper bound points, $y = 2x + 3$, is above the lower bound and, thus, is feasible and recorded as ($\beta_0 = 3$, $\beta_1 = 2$). Then, P becomes next pivot and the procedure is terminated since there are no succeeding points left. As a result, one upper bound vertex is found with indices $c = 1$ and $d = 3$.

$V_L = \emptyset$ and $V_U \neq \emptyset$ corresponds to Case 3 (line 14) of Algorithm A.2. Therefore, $l_{\min}$ and $l_{\max}$ are formed by (x_z, L_z), (x_1, U_1) and (x_z, L_z), (x_3, U_3), respectively. The operations given in Case 2 are not necessary to perform since $1 < z < 3$ and, therefore, $z = 2$. Thus, $l_{\min}$ is formed by (x_2, L_2) and (x_1, U_1) where $m_{\min} = 1.6$ and $n_{\min} = 2.8$. Consequently, $l_{\max}$ is formed by (x_2, L_2) and (x_3, U_3) where $m_{\max} = 2.666$ and $n_{\max} = 2.666$. The solution region is plotted in Figure 3.10, along with the one found in [37].

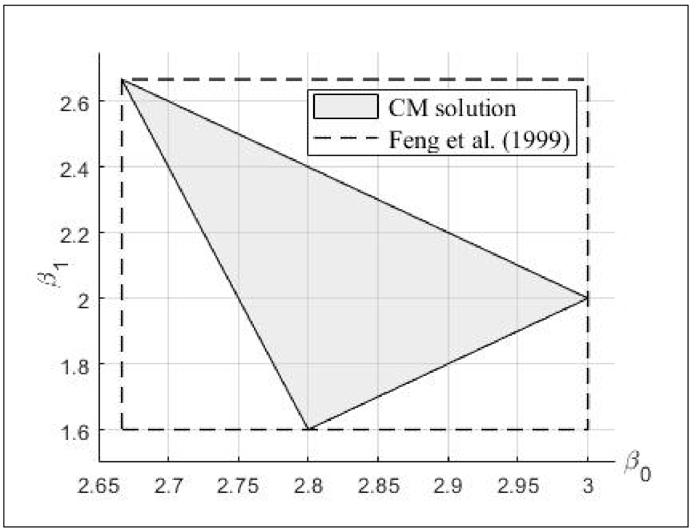

Figure 3.10. The solution region of the example.

3.5. AN APPLICATION

A second simulation with synthetic datasets is performed to observe CM performance for large N values. In this simulation, each experiment is repeated 100 times, in which N ranges from 10^3 to 10^5. If the minimum and maximum T_c values for each N value are considered as the limits of the true T_c, then an uncertainty corridor can be formed using these limits, as demonstrated in Figure 3.11.

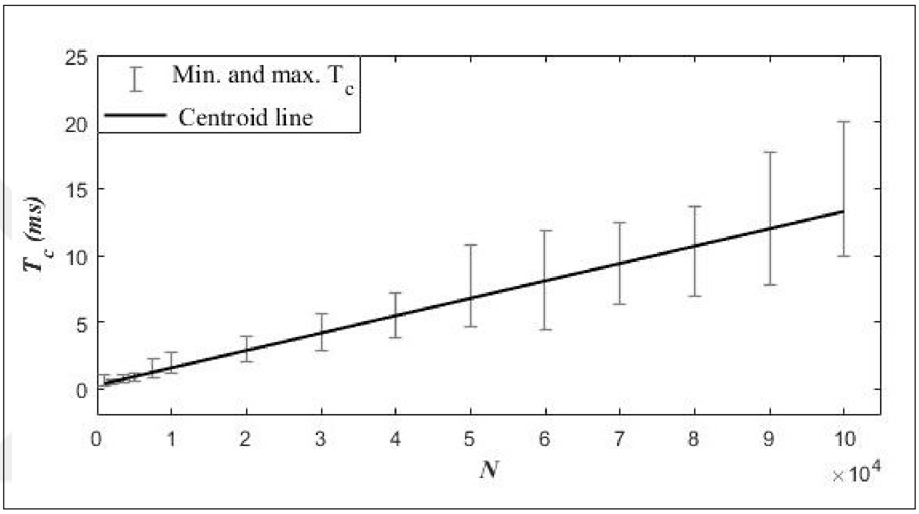

Figure 3.11. The limits of the T_c values in the simulation with respect to N

Assuming a linear relationship between the T_c and N, the proposed method can be applied to estimate this relationship. The solution region of the corridor is shown in Figure 3.12. If the centroid, $\beta_0 = 2.68 \times 10^{-4}$, $\beta_1 = 1.30 \times 10^{-7}$, of the solution polygon is considered as an estimator for the parameters of the simple linear model, the relation between T_c and N is given as follows.

$$T_c = 1.30 \times 10^{-7} N + 2.68 \times 10^{-4} \tag{3.14}$$

The reader should be cautioned that although a linear fit is obtained for CM computational time, even when the number of observations are extended to 10^5, essentially, the algorithm does not run in linear time.

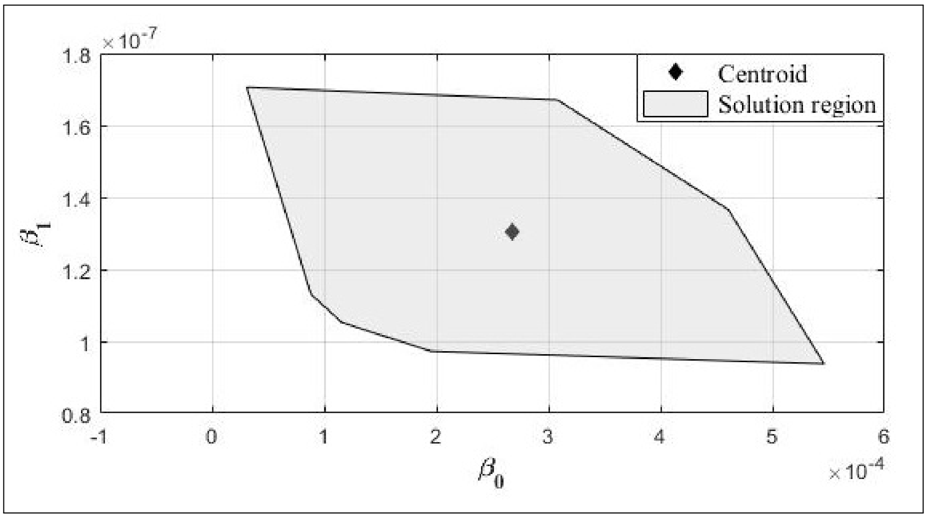

Figure 3.12. The solution region of the application

3.6. COMPLEXITY OF CM

CM moves on the lower and upper bound points by jumping from a vertex to the next, skipping the intermediate points. Thus, the total number of vertices, $v = |V|$, plays a critical role in the computational time T_c. For each vertex the maximum number of operations is proportional to N. Hence, the actual complexity can be estimated to be $O(vN)$.

Nevertheless, v varies depending on the form of the corridor. Therefore, though being very unlikely, the worst-case scenario should also be considered, in which every adjacent pair of lower and upper points forms a vertex. In this case, $|V_L| = |V_U| = N - 1$, and $v = 2N$, including $l_{\min}$ and $l_{\max}$. Therefore, the time complexity can be estimated to be $O(N^2)$. However, a much closer estimation can be made by counting the total number of operations as follows.

For V_L, starting in the first lower bound point, CM creates $N - 1$ lines through all succeeding points. Since the line passing through the first and second points would form a vertex, the process continues with pivoting on the second point and $N - 2$ lines are created this time. Therefore, the number of required operations can be given as $\frac{1}{2}N(N - 1)$. These vertices are than checked against N upper bound points, giving $N(N - 1)$ more operations that brings the total to $\frac{3}{2}N(N - 1)$. The same is true for V_U. Hence, the overall total is $3N(N - 1)$. Since the determination of $l_{\min}$ and $l_{\max}$ does not depend on N, its contribution to T_c is negligible.

As a result, $T_c \sim 3N(N-1) \rightarrow O(N^2)$ in the worst-case scenario.

Nevertheless, since this scenario is extremely improbable, a better assessment of complexity could be based on median T_c values. To this end, two MATLAB simulations are performed: one for the worst-case scenario and one for the median-case. In both, the number of observations N were taken from 1000 to 50000, in increments of 1000. For the median-case, experiment was repeated 100 times for each N with randomly generated data.

Results strongly validate the claim that the complexity of the worst-case is $O(N^2)$ (Figure 3.13). More importantly, however, the same for the median T_c seems to be $O(N)$. A second order polynomial fit on the worst-case data and a linear fit on median T_c data yielded correlation coefficients of 0.9998 and 0.9971, respectively. For $N = 50000$ the run times were about 43 s and 7 ms for the worst and median-cases, respectively.

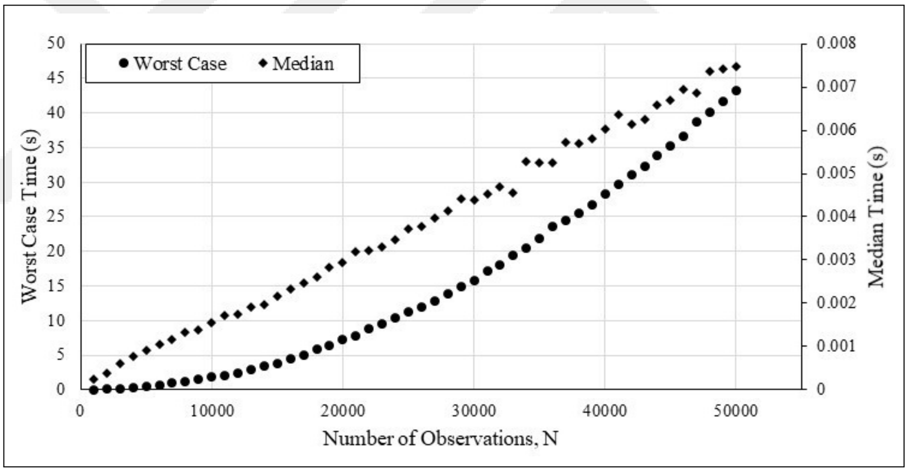

Figure 3.13. Time complexity simulations of the worst-case scenario and random data.

3.7. PERFORMANCE COMPARISON OF CM

In this section, the performance of the proposed algorithm is compared to one pivoting and one non-pivoting algorithms, both well-known.

The first competitor is the lexicographic reverse search algorithm proposed by [38], *lrs*, a revised version of the reverse search vertex enumeration algorithm proposed by [19]. Although a C language implementation is available in [39], the algorithm is implemented in MATLAB to provide a common platform for the algorithms to compete.

The reader should note that lrs requires an initial vertex of S as an input. Normally, the time required to find this vertex would need to be added to the lrs time to be fair to the other algorithms. However, there are various approaches to finding the initial vertex and each performs differently. In order not to shade the lrs performance, this preprocessing time is not included in the lrs times. Nevertheless, it should be emphasized that in some experiments initial vertex search takes much longer than the lrs time.

The second competitor is a variant of the double description method (DDM) ([8]), available in BENSOLVE, a tool for vector linear programming. A MATLAB implementation of BENSOLVE is available in [40], with further improvements in [41], which was used in this study directly.

T_c performances of the algorithms are tested in MATLAB using synthetically generated datasets. The experimental design criterion in these simulations is the number of observations N, where N is assigned 9 different values ranging from 5 to 5000. For each N, the experiment is repeated 100 times.

Table 3.1. Median run times (ms) of the three algorithms

N	5	10	25	50	100	250	500	1000	5000
lrs	3.4	3.7	3.9	6.0	10.2	69.8	331	2200	187221
DDM	11.0	11.6	13.1	14.4	18.3	33.4	41.7	69.1	331
CM	0.09	0.10	0.11	0.17	0.16	0.16	0.18	0.26	1.37

The median performances are given in Table 3.1. For a better comparison, the ratios of median T_c values of the competing algorithms to that of CM are presented in Table 3.2.

Table 3.2. Ratios of median run times with respect to CM

N	5	10	25	50	100	250	500	1000	5000
DDM/CM	125	115	118	83	117	204	234	270	242
lrs/CM	39	37	35	35	65	426	1855	8603	136783

It is observed that up to around $N = 100$ lrs performs better than DDM. However, for $N > 100$ performance of lrs quickly deteriorates. In contrast, the performance of DDM compared to CM remains more or less the same. The median time complexities of both CM and DDM seem to be $O(N)$, whereas that for lrs is very close to $O(N^2)$.

4. EXTENSION OF CM FOR LINEAR OPTIMIZATION PROBLEMS WITH TWO PARAMETERS

Recall that the uncertainty corridor is formed by observations,where each observation is associated with two inequalities, one from the lower and the other from the upper bound.

$$L_i \leq \beta_0 + x_i\beta_1 \quad \text{and} \quad \beta_0 + x_i\beta_1 \leq U_i \quad \text{for } i = 1, \cdots, N \tag{4.1}$$

This is a consequence of the UBBE case. However, in the vast majority of the optimization problems, a set of independent linear inequalities is given. In such problems, the numbers of lower and upper inequalities may not be equal, or there may not even be an inequality from one side. In order to modify CM to cover such problems, the corridor is split independent lower and upper bounds.

Let the numbers of lower and upper bound points be denoted by N_L and N_U, respectively. As they are independent, they are no longer associated with common x values. We will denote these points by (x_i^L, L_i) for the lower and (x_j^U, U_j) for the upper bound, where $i \in [1, N_L]$ and $j \in [1, N_U]$. In this case, one should update the assumptions made in the UBBE case as follows.

Assumptions:

1. All lower bound points are at distinct x values, $x_i^L \neq x_k^L$, for $i \neq k$ where $k \in [1, N_L]$.
2. All upper bound points are at distinct x values, $x_j^U \neq x_z^U$, for $j \neq z$ where $z \in [1, N_U]$.

With the separated bounds, the feasible lines in $\mathbb{M}$ can be defined in two ways: The ones above the lower and the ones below the upper bound.

Definition 2. *R_L is the set of all lines in $\mathbb{M}$ that stay above the lower bound. R_U is the set of all lines that stay below the upper bound.*

R_L is constituted by the elements that conform to $L_i \leq \beta_0 + x_i^L\beta_1 \quad \text{for} \quad i = 1, \cdots, N_L$. Similarly, R_U is constituted by the elements that conform to $\beta_0 + x_j^U\beta_1 \leq U_j \quad \text{for} \quad j = 1, \cdots, N_U$. The intersection of R_L and R_U defines S.

$$S = R_L \cap R_U \tag{4.2}$$

Since the lines in $R_L(R_U)$ are not constrained by an upper(lower) bound this time, one can state the following lemma.

Lemma 2. *R_L and R_U are unbounded convex sets.*

As R_L and R_U are unbounded, they are described by their vertices in addition to extreme rays. Moreover, it is not guaranteed that the resulting S will be bounded. In such a case, S is also to be described by its vertices and extreme rays.

Again, one can employ the boundary sets of R_L and R_U in determination of the vertices and the extreme rays. Let the boundary sets of R_L and R_U be $\partial R_L \subset R_L$ and $\partial R_U \subset R_U$, respectively. One can define these sets as follows.

Definition 3. *∂R_L is the set of all lines in $\mathbb{M}$ that stay above the lower bound and contact at least one lower bound point. ∂R_U is the set of all lines that stay below the upper bound and contact at least one upper bound point.*

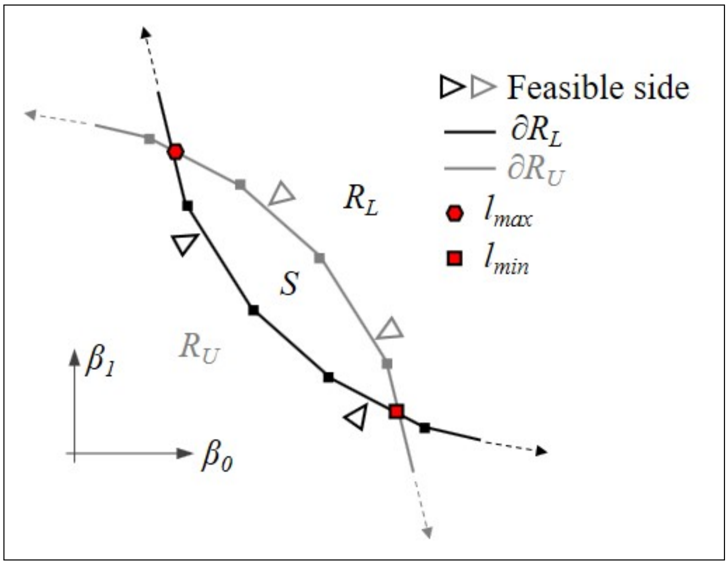

Figure 4.1. Intersection of R_L and R_U

This definition provides adequate information about the correspondence of ∂R_L and ∂R_U with ∂S. Based on this definition, $S_L \cup S_{UL} \subset \partial R_L$ and $S_U \cup S_{UL} \subset \partial R_U$, which can also be observed in Figure 4.1. The image of S_L is a part of the image of ∂R_L, similarly, the image of S_U is a part of the image of ∂R_U in $\mathbb{P}$. Note also that the elements of S_{UL} $(l_{\min}, l_{\max})$, if any, are at the intersection of ∂R_L and ∂R_U.

4.1. DETERMINATION OF R_L AND R_U

Consider the images of ∂R_L and ∂R_U in $\mathbb{P}$. They are composed of line segments that join by vertices, both *polygonal chains*. Recall that rotation of a line about a lower(upper) point in $\mathbb{M}$ corresponds to a line segment of $\partial R_L(\partial R_U)$ in $\mathbb{P}$. In order to better grasp the geometry of the chains, we will first take a closer look at what happens to the line segment if position of the lower(upper) point changes. Consider a point (x_p, y_p) in $\mathbb{P}$ where rotation of a line about (x_p, y_p) forms a line in $\mathbb{M}$. The relation is given by

$$\beta_0 = y_P - x_P\beta_1 \tag{4.3}$$

Here, y_P represents the intercept of β_0-axis whereas $-x_P$ represents the slope measured from β_1-axis in CW direction (see Figure 4.2).

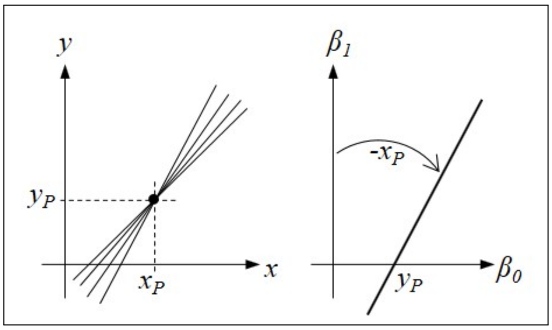

Figure 4.2. Definition of slope and intercept in $\mathbb{P}$

Corollary 3. *By Equation 4.3, if y_P is increased without a change in x_P, then the line in $\mathbb{P}$ is parallel shifted along the β_0-axis and if x_P is increased without a change in y_P, then the line rotates about the intercept point $(y_P, 0)$ in CCW direction (see Figure 4.3). Additionally, $(x_P = 0, y_P)$ corresponds to a vertical line passing through $(y_P, 0)$, and $(x_P = \infty, y_P)$ corresponds to a horizontal line in $\mathbb{P}$.*

Corollary 4. *Based on $\beta_0 \geq L_i - x_1\beta_i$, the right hand side of ∂R_L segments is always feasible. If the segment were horizontal, then one would better describe the feasible side as above or* *below. However, this is not possible due to $x_1^L, x_{N_L}^L \neq \infty$, Corollary 3. One can also assert that the left hand side of ∂R_L segments is always feasible, by $\beta_0 \leq U_j - x_j\beta_1$.*

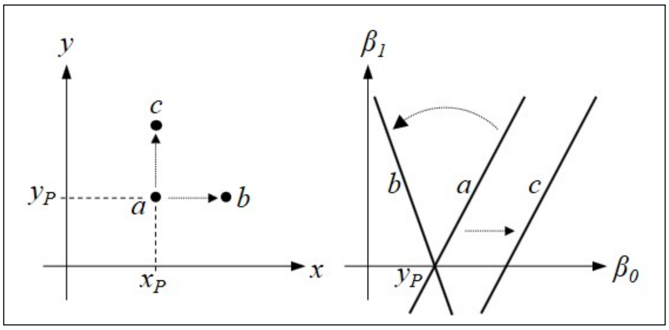

Figure 4.3. Consequences of the rotation point translation in $\mathbb{P}$

We will first study R_L and then R_U. Consider ∂R_L. Each lower bound point represents a line in $\mathbb{P}$ where the right sides of these lines are feasible by Corollary 4. Let each line be denoted by H_i where i represent the lower point index. The lines intersect and define ∂R_L, where the intersection points are the vertices. When a line intersects another, a portion of it that stays at the infeasible side of the other becomes infeasible. In this way, the lines become the line segments of ∂R_L, see Figure 4.4.

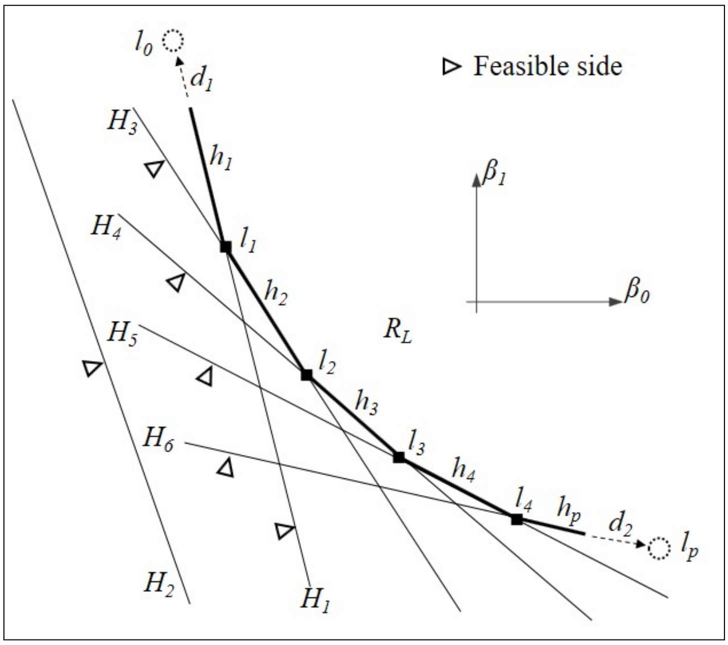

Figure 4.4. Demonstration of R_L formed by the lower bound half-spaces.

Note that some of the lines are redundant and has no effect on the formation of R_L, e.g. H_2 in Figure 4.4 is a redundant line. The redundant lines correspond to the lower bound points that are skipped during the enumeration of the lower bound vertices by Algorithm A.1, inactive constraints.

Consider the first and the last segments in the chain. They are half-open line segments with one end connected to a vertex and the other located at infinity. We will consider the end points at the infinity, say $l_0, l_p \in \mathbb{P}$ as imaginary vertices of R_L. Let $V_L = \{l_0, l_1, \cdots, l_p\}$ be the set of vertices ordered in descending order with respect to β_1-axis. Note that normally l_0 and l_p would not be in V_L that will be used to describe R_L. However, we will employ them in the method that will be proposed to be used in finding the intersection R_L and R_U. Their existence is helpful in cases of $|L| = 1$ and/or $|U| = 1$, where the chains have a single line segment with no vertices, and in cases of where the intersection is observed at the first or the last segments. For now, we allow l_0 and l_p in V_L. After the intersection, if they are still in the vertices list of S, they will be removed.

The remaining intermediate segments are closed, with both ends connected to vertices. Let $h_L = \{h_1, h_2, \cdots, h_p\}$ be the set of the line segments of ∂R_L and ordered with respect V_L, i.e., h_1 connects l_0 and h_p connects l_p. Keep in mind that there are $p + 1$ vertices and p line segments.

Consider l_1, a line in $\mathbb{M}$ that touches at least two lower bound points, namely a left and a right contact points. Since the upper bound is not considered at this time, it is clear that the first lower bound point (x_1^L, L_1) is always the left contact point of l_1. If ones frees the right contact point and pivots on (x_1^L, L_1) in CCW direction, then this rotation traces h_1 in $\mathbb{P}$ by starting at l_1 and running in the direction of β_1-axis, since the slope is increased. Thus, the line equation of h_1 is $\beta_0 = L_1 - x_1^L \beta_1$. Afterwards, when the line becomes vertical, l_0 is reached. Note that further rotation in the same sense would violate the lower bound.

On the other hand, if one frees (x_1^L, L_1) and pivots on the right contact point of l_1, say (x_z^L, L_z), in CW direction (slope decreases this time), then h_2 is traced by starting at l_1 and moving in the direction of negative β_1-axis, giving the line equation of h_2 as $\beta_0 = L_z - x_z^L \beta_1$. Note that the line equations of the segments can simply be determined by $\beta_0 = L_z - x_z^L \beta_1$ where z denotes the index of the pivoting point in $\mathbb{M}$.

Eventually, the rotating line contacts a lower bound point succeeding (x_z^L, L_z), and l_2 is reached. By repeating the same process, freeing the left and pivoting on the right, one can trace all remaining intermediate line segments and reach l_{p-1}. Similar to the case of l_1, the right contact point of l_{p-1} is always the last lower bound point $(x^L_{N_L}, L_{N_L})$. If the process is continued by pivoting on $(x^L_{N_L}, L_{N_L})$, then one traces h_p by starting at l_{p-1}. Again, when the line becomes vertical, no further rotation is possible, and the vertex l_p is reached. This procedure determines V_L in an efficient way. Note that, in this procedure the pivoting point of each segment has a larger x value than the preceding one. Therefore, the following corollary.

Corollary 5. *The slopes of ∂R_L segments decrease monotonically from h_1 to h_p.*

In order to complete the description of R_L, one should determine the extreme rays of R_L. The set of extreme rays is constituted by rays parallel to the segments h_1 and h_p in the correct sense, see Figure 4.4. Let $\overrightarrow{d_1}, \overrightarrow{d_2} \in R^2$ be vectors with the same orientation of h_1, h_p in the correct sense, respectively. The extreme rays can be obtained by $\gamma_1\overrightarrow{d_1}$ and $\gamma_2\overrightarrow{d_2}$ where $\gamma_{1,2} \geq 0$. In what follows, for convenience in the methods to be introduced, $\overrightarrow{d_1}$ and $\overrightarrow{d_2}$ will be used to represent the extreme rays of R_L. Although the orientations of $\overrightarrow{d_1}, \overrightarrow{d_2}$ can be obtained using the line equations of h_1 and h_p, their senses are still to be determined and which is straightforward.

Since l_0 is at positive infinity with respect to β_1-axis, let $P_1 \in h_1$ be a point above l_1 with respect to β_1-axis. Therefore, $\overrightarrow{d_1} = \overrightarrow{l_1 P_1}$ will be in the correct sense. Similarly, l_p is at negative infinity with respect to β_1-axis. Let $P_2 \in h_p$ be a point below l_p with respect to β_1-axis. Thus, $\overrightarrow{d_2} = \overrightarrow{l_{p-1} P_2}$.

Let the coordinates of l_1 and that of P_1 be $(\beta_0^{l_1}, \beta_1^{l_1})$ and $(\beta_0^{P_1}, (\beta_1^{l_1} + \delta))$ in $\mathbb{P}$, respectively, where δ is an arbitrary positive number. Recall that both l_1 and P_1 lie on h_1 and therefore, must satisfy $\beta_0 = L_1 - x_1^L \beta_1$. Thus, one determines $\beta_0^{P_1}$ as follows

$$\beta_0^{l_1} = L_1 - x_1^L \beta_1^{l_1} \tag{4.4}$$

$$\beta_0^{P_1} = L_1 - x_1^L (\beta_1^{l_1} + \delta) \tag{4.5}$$

by subtracting the first from the second

$$\beta_0^{P_1} = \beta_0^{l_1} - x_1^L \delta \tag{4.6}$$

In a similar way, let the coordinates of l_{p-1} and P_2 be $(\beta_0^{l_{p-1}}, \beta_1^{l_{p-1}})$ and $(\beta_0^{P_2}, (\beta_1^{l_{p-1}} - \delta))$, respectively. Both points are on $\beta_0 = L_{N_L} - x_{N_L}^L \beta_1$. Thus,

$$\beta_0^{P_2} = \beta_0^{l_{p-1}} + x_{N_L}^L \delta \tag{4.7}$$

Therefore, the extreme rays are

$$\overrightarrow{d_1^L} = \overrightarrow{l_1 P_1} = \begin{bmatrix} \beta_0^{l_1} - x_1^L \delta \\ \beta_1^{l_1} + \delta \end{bmatrix} - \begin{bmatrix} \beta_0^{l_1} \\ \beta_1^{l_1} \end{bmatrix} = \begin{bmatrix} -x_1^L \delta \\ \delta \end{bmatrix} \tag{4.8}$$

$$\overrightarrow{d_2^L} = \overrightarrow{l_{p-1} P_2} = \begin{bmatrix} \beta_0^{l_{p-1}} + x_{N_L}^L \delta \\ \beta_1^{l_{p-1}} - \delta \end{bmatrix} - \begin{bmatrix} \beta_0^{l_{p-1}} \\ \beta_1^{l_{p-1}} \end{bmatrix} = \begin{bmatrix} x_{N_L}^L \delta \\ -\delta \end{bmatrix} \tag{4.9}$$

Or, by taking $\delta = 1$

$$\overrightarrow{d_1} = \begin{bmatrix} -x_1^L \\ 1 \end{bmatrix} \quad \text{and} \quad \overrightarrow{d_2} = \begin{bmatrix} x_{N_L}^L \\ -1 \end{bmatrix} \tag{4.10}$$

After exact description of R_L by V_L and $\overrightarrow{d_1}, \overrightarrow{d_2}$, we will proceed with the description of R_U.

Consider the image of ∂R_U in $\mathbb{P}$, another convex polygonal chain. Let $V_U = \{u_0, u_1, \cdots, u_q\}$ be the set of vertices ordered in descending order with respect to β_1-axis. Further, let $h_U = \{k_1, k_2, \cdots, k_q\}$ be the set of the line segments and ordered with respect to V_U, i.e., k_1 connects to u_0 and k_q connects to u_q.

This time, the lines in $\mathbb{M}$ must be below the upper bound. The procedure introduced in determination of V_L is executed in reverse. u_0 corresponds to a vertical line that passes through the last upper bound point $(x_{N_U}^U, U_{N_U})$. One rotates this line about $(x_{N_U}^U, U_{N_U})$ in CW direction (slope decreases). This traces k_1 in $\mathbb{P}$ by starting at u_0 and moving in the direction of negative β_1-axis. When the line contacts another upper bound point, $u_1$is reached. Clearly, the new contact point is preeceding the pivot. Therefore, the pivoting point of each segment has a lesser x value than the preceding one.

Lemma 3. *The slopes of ∂R_U segments increases monotonically from u_1 to u_q, .*

The process continues by freeing the right and pivoting on the left. In this way, all elements of V_U are enumerated in desired order. When the pivot becomes the first point (x_1^U, U_1), rotation continues until the line becomes vertical that corresponds to u_q.

Similar to h_1 and h_q, u_0 and u_q are always at positive and negative infinity with respect to β_1-axis, respectively. Let $\overrightarrow{e_1}, \overrightarrow{e_2} \in R^2$ be vectors with the same orientation of k_1, k_p in the correct sense, respectively. Similar to the calculations made in determination of the extreme rays of V_L, by employing two points that are above u_1 and below u_{q-1}, one can determine the extreme rays of R_U

$$\overrightarrow{e_1} = \begin{bmatrix} -x_{N_U}^U \\ 1 \end{bmatrix} \quad \text{and} \quad \overrightarrow{e_2} = \begin{bmatrix} x_1^U \\ -1 \end{bmatrix} \tag{4.11}$$

Implementations of the method to describe $R_L(R_U)$ are introduced below. In addition to the vertices and the extreme rays, in order to employ later in the intersection algorithm, the set of line segments $h_L(h_U)$ are also desired. Recall that the line segments can be represented by their pivot points.

Further, the vertices l_0, u_0, l_p, u_q are at infinity and their positions in an implementation can be determined by the extreme rays $\overrightarrow{d_1}, \overrightarrow{e_1}, \overrightarrow{d_2}, \overrightarrow{e_2}$, respectively, and the possible largest number of the computation platform, say Inf. For instance, the position of l_0 is given by $Unit(\overrightarrow{d_1}) \times Inf$.

The proposed method to determine V_L, r_L and V_U, r_U is presented respectively in Algorithms A.3 and A.4 as pseudocodes.

4.2. DETERMINATION OF S BASED ON INTERSECTION OF R_L AND R_U

The problem can be described as a polygonal chain intersection problem. Despite the existence of numerous algorithms in the literature, new solutions with increased efficiency can be developed using the geometric characteristics of the chains. The following will go over some topics to better understand the structure of chains:

i) How the location of the boundary affects the position and orientation of the chains.

ii) The fundamentals of chain intersection to develop a successful algorithm.

We explore a scenario in which a bound travels in $\mathbb{M}$ to see its consequences in $\mathbb{P}$. Although y-axis translation does not contribute to our analysis and will thus be disregarded, an example will be looked at to provide the necessary evidence. ∂R_L has two vertices in this scenario, l_1 and l_2. The lower bound is deliberately constructed in concave down form to have positive and negative β_1 values for l_1 and l_2, respectively.

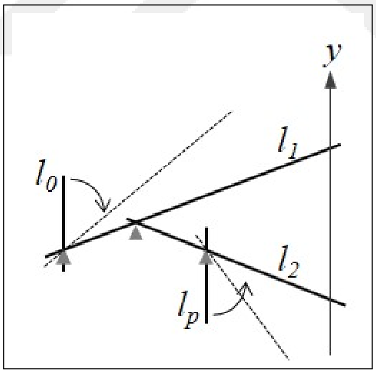

Figure 4.5. The lower bound case A where the lower bound stays left hand side of the y-axis

In first place, we consider a lower bound that is to the left of the y-axis, $x_1^L, x_{N_L}^L < 0$. By $\overrightarrow{d_1}$, l_0 is at (∞, ∞) in $\mathbb{P}$. This can also be observed in Figure 4.5. If one rotates l_0 in $\mathbb{M}$ about its contact(pivot) point in CW direction, β_0 and β_1 values get smaller starting from positive infinity. Similarly, if one rotates l_p in CCW direction, β_0 and β_1 values become larger starting from negative infinity. Use of $\overrightarrow{d_2}$ can simply achieve the same. Thus, $l_p : (-\infty, -\infty)$ in $\mathbb{P}$. In this case, ∂R_L is a concave down chain (A) as demonstrated in Figure 4.6.

Based on Corollary 3, parallel shifts of the bound through the y-axis result in translation of ∂R_L along the β_0-axis (see Figure 4.7). They will not be discussed in any of the following

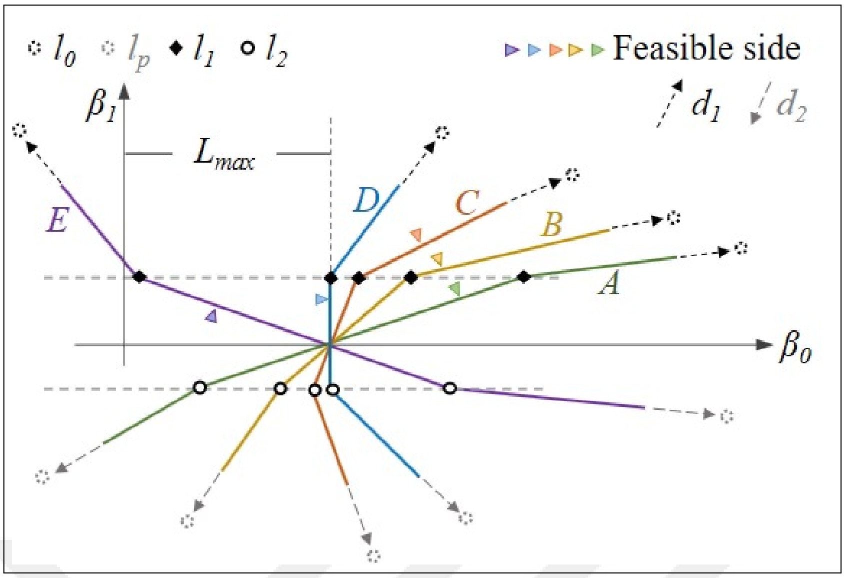

Figure 4.6. Illustration of different cases created for the example scenario

cases, as it does not provide the necessary information for our analysis.

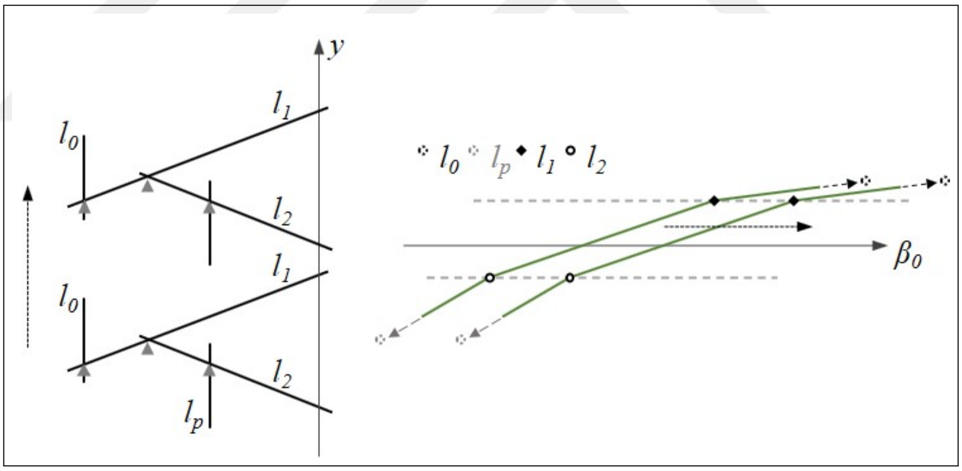

Figure 4.7. Parallel shifting of the bound through the y-axis

On the other hand, if one moves the bound to the right, then $x_1^L, x_{N_L}^L$ approaches zero and all line segments of ∂R_L rotates in CCW (Corollary 3) as demonstrated in Figure 4.6 as case B. Furthermore, β_1 values of l_1 and l_2 do not change during this shifting. In this case, elements of V_L move horizontally in $\mathbb{P}$. Based on Equation 4.3 and Figure 4.8, those with a positive β_1 have a decreasing β_0 and are to move to the left, while those with a negative β_1 will have an increased β_0, yielding a move to the right. Since this applies to all of the cases in the scenario that follow, it won't be brought up again.

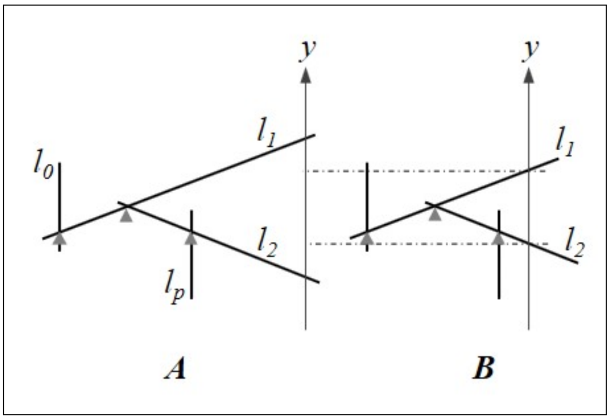

Figure 4.8. Demonstration of the lower bound in the cases A and B of the scenario

When $x^L_{N_L} > 0$ (see Figure 4.9 case C), the rotating line of l_p starts intersecting $y-$axis from above its pivot point, instead of below, giving a positive β_0 for l_p. Thus, $l_p : (\infty, -\infty)$. At this point, ∂R_L is a right-facing c-shaped chain as depicted in Figure 4.6 case C. In continuation, when the common point of l_1 and l_2 in $\mathbb{M}$, namely the right and the left contact points of two respectively, lies on the y-axis, then β_0 values of both become equal (see Figure 4.9 case D). This forms a vertical line segment between between l_1 and l_2 in $\mathbb{P}$, see Figure 4.6 chain D.

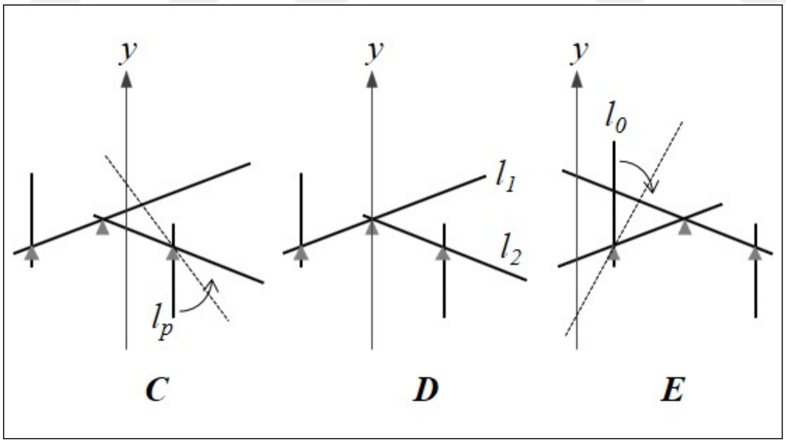

Figure 4.9. Demonstration of the lower bound in the cases A, B and C of the scenario

Lastly, when the first bound point stays to the right of y-axis, $x^L_1 > 0$, the rotating line of l_0 starts intersecting y-axis from below its pivot point, instead of above, (see Figure 4.9 case E). This means a sign change in β_0 of l_0. Another inevitable conclusion that $\overrightarrow{d_1}$ could draw, $l_0 : (-\infty, \infty)$. ∂R_L forms a concave up chain in this state, see Figure 4.6 chain E.

In sum, when the bound advances to the right, R_L rotates in the CCW direction (Corollary 3) on a specific point $(L_{\max}, 0)$, leaving the region to the right of ∂R_L feasible, by Corollary

4. The rotation point $(L_{\max}, 0)$ refers to a common element of all different ∂R_L that are generated during shifting, which is a flat line passing through the highest point of the bound.

Similar arguments hold for ∂R_U with some exceptions. Recall that the line segments are associated by the x-values of the upper points in reverse order. In addition, the feasible region is always left to ∂R_U. Further, when one starts to shift the upper bound through x-axis, ∂R_U starts to rotate about $(0, U_{\min})$ in CCW direction, where $U_{\min}$ is the elevation of the lowest point of the bound. Different ∂R_U that are formed by shifting of the upper bound through x-axis are depicted in Figure 4.10.

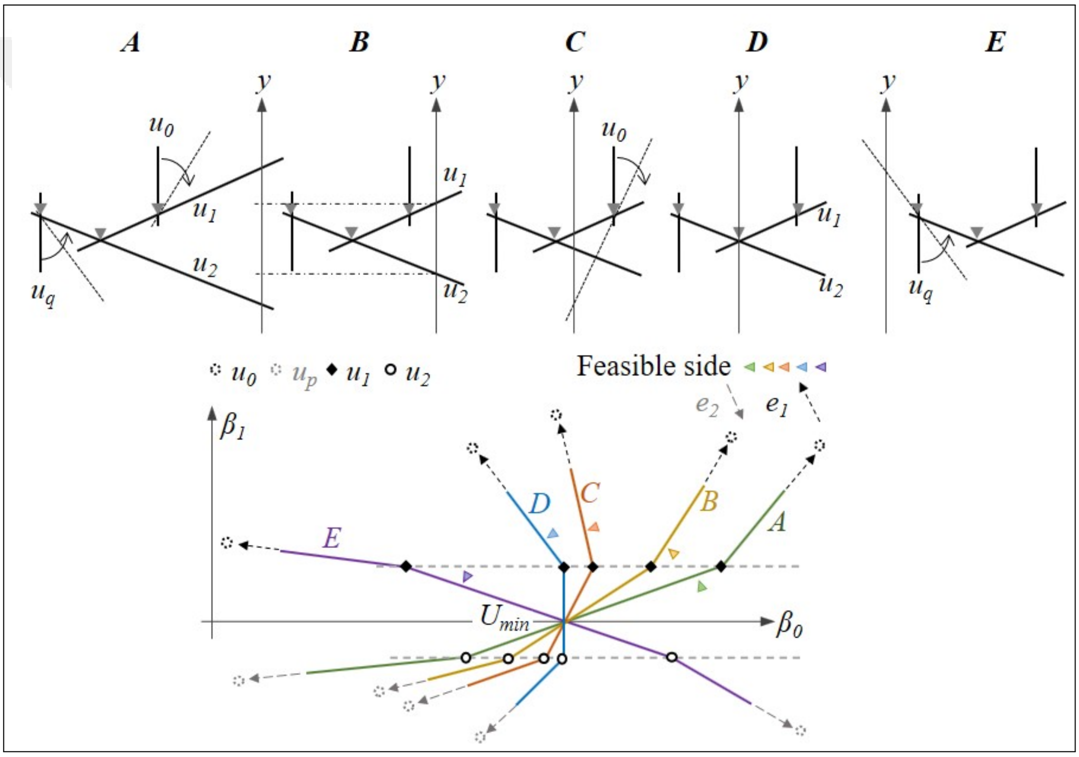

Figure 4.10. The same scenario for the upper bound

After the investigation of ∂R_L and ∂R_U, we proceed with the intersection of the chains. Three possible generic examples of the intersection are demonstrated in Figure 4.11. In the first example, l_0 and u_0 are feasible. We will call this case as *open top end.* The chains must intersect at the bottom by forming $l_{\min}$. We will prove this by the following Proposition. Therefore, the bottom end is closed, or *closed bottom end*. The feasible region is always unbounded where the extreme rays are $\overrightarrow{d_1}$ and $\overrightarrow{e_1}$. In order to find $l_{\min}$, one is to determine the intersecting line segments. Obviously, starting from the bottom will be more efficient to find these segments.

Proposition 1. *If the top end is open, then the bottom end is closed. Similarly, if the bottom end is open, then the top end is closed.*

Proof. In order to have an open top end, the slope of k_1 must be greater than or equal to the slope of h_1, in other words $-x_1^L \leq -x_{N_U}^U$. Therefore $x_1^L \geq x_{N_U}^U$ for an open-top. Similarly, an open bottom requires the slope of k_q to be less than or equal to the slope of h_p, namely $x_{N_L}^L \leq x_1^U$.

Consider the open top end where $x_1^L \geq x_{N_U}^U$. Since $x_1^L < x_{N_L}^L$ and $x_1^U < x_{N_U}^U$, $x_{N_L}^L > x_1^U$ must hold, yielding a close bottom. One can show that if the bottom end open, then the top end must be closed in a similar way. □

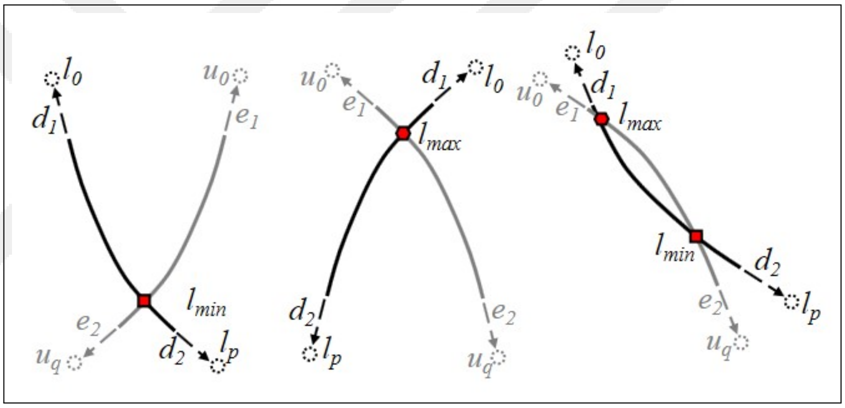

Figure 4.11. The three generic intersection cases

In the next example, l_p and u_q are feasible. Thus, an open bottom and a closed top ends (Proposition 1) in which $l_{\max}$ exists this time. Again, S is unbounded and described by the rays $\overrightarrow{d_2}$ and $\overrightarrow{e_2}$. One searches for the intersecting line segments by starting from the top. In the last example, both ends are closed, providing that $l_{\min}$ and $l_{\max}$ exist. Moreover, S is bounded. The intersecting segments should be determined starting from either end. Note that this case is also observed in case of no solution, but we will discuss this later.

In what follows, we will introduce the intersection algorithm by parts for convenience. Based on Proposition 1, one first determines the status of the ends by Algorithm A.5. We will use 0 and 1 to represent an open and a closed end, respectively.

Note that the case of open top and bottom can only be observed, if each of ∂R_L and ∂R_U have one line segment h_1 and k_1 that are parallel to each other, $x_1^L = x_1^U$, see Figure 4.12.

In this case:

i) If k_1 is to the right of h_1, $U_1 > L_1$, then S is a strip, therefore unbounded.

ii) If k_1 is coincident with h_1, $U_1 = L_1$, then S is a line.

iii) Finally, if k_1 is to the left of h_1, $U_1 < L_1$, then $S = \emptyset$.

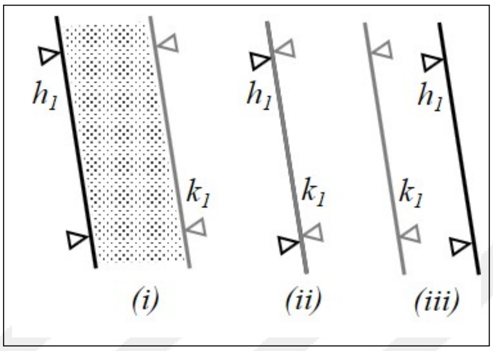

Figure 4.12. Possibilities if there are one observation from each side and $x_1^L = x_1^U$

Since this case is very improbable and also straightforward to solve even manually, it is excluded from the intersection analysis.

Consider the intersection of the top ends where the intersecting segments can be determined by the first feasible vertices, say l_i, u_j, of both chains from the top. Since l_i is feasible and h_{i-1} is not, the line segment between them h_i must be the intersecting segment of ∂R_L. In the same way, Since u_j is feasible and u_{j-1} is not, the line segment between them k_i must be the intersecting segment of ∂R_U.

One can determine $l_i(u_j)$ by testing all $V_L(V_U)$ elements against all segments of $\partial R_U(\partial R_L)$. However, this is a computationally expensive task. Instead, a time efficient method is proposed as follows. One starts to examine the vertices in V_U one by one from u_1 to u_q against the first line segment h_1 of R_L. If the vertex is to the right of h_1, then feasible.

Subsequently, the vertices in V_L are tested against the segment of ∂R_U that is determined by the feasible vertex of the previous iteration. The same process continues as one after another until a feasible vertex found in the previous iteration is encountered again.

See the example demonstrated in Figure 4.13. The first feasible vertex of ∂R_U against h_1 is u_3. This test can be performed by $\beta_0 \geq L - x\beta_1$ where x and L are the coordinates of

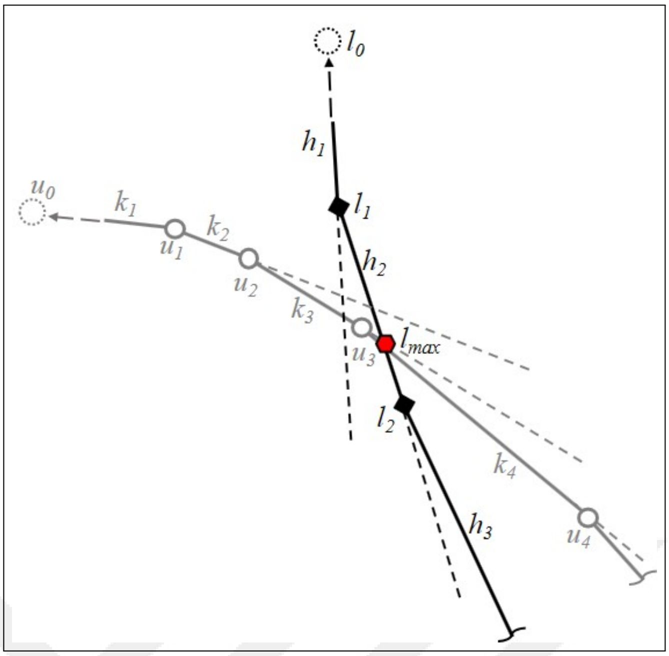

Figure 4.13. An example of closed top end

the corresponding lower bound point of h_1 and β_0, β_1 are the coordinates of the vertices in $\mathbb{P}$. Note that each test is terminated once a feasible vertex is encountered. Additionally, the vertices are tested starting from u_1 instead of u_0. The reason is to have a closed top end where both l_0, u_0 are always known to be infeasible.

The segments of infeasible vertices u_1 and u_2 (k_1 and k_2) cannot be the intersecting segment, however, k_3 can be. In other words, h_1 intersects k_3. The process then continues with examination of V_L elements against ∂R_U to find the intersecting segment of ∂R_L. One employs k_3 and seeks for the first feasible in V_L starting from l_1. This test is performed by $\beta_0 \leq U - x\beta_1$ in which l_2 is found as the first feasible. Similarly, h_1 cannot be the intersecting segment any longer. The examination continues with V_U elements against h_2. Note that the test starts with u_3 since the preceding ones were already found infeasible. This test yields u_4. Then, starting from l_2 one checks feasibility of V_L elements by k_4 where the first feasible is l_2 again. Since its segment h_2 was found in the previous iteration, at this point, the operation is terminated as it converges to the result. Otherwise, it would result in the continual repetition of the last two iterations, i.e., tests by h_2 yields k_4 and k_4 yields h_2 again. Consequently, the last segments h_2 and k_4 are determined as the intersecting segments.

One may question the necessity of l_p and u_q. However, in case of intersection of the last segments, the chains without l_p and u_q would not have any feasible vertex. The proposed top intersecting method is implemented in Algorithm A.6. I_T and J_T are the outputs and correspond to indices of the intersecting segments of ∂R_L and ∂R_U, respectively.

The same method can also be employed in determination of the intersecting segments of the closed bottom end. This time the procedure is applied in reverse order. See the example shown in Figure 4.14.

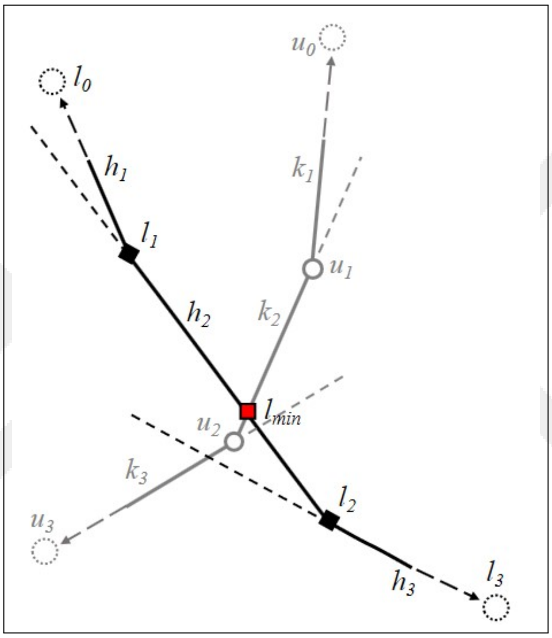

Figure 4.14. Closed bottom end example

The process starts with examination of V_U elements in reverse order, from u_2 to u_0, against the last segment h_3 of ∂R_L. Again, u_3, l_3 are not examined as they are infeasible in the closed bottom case. The first test gives u_2 as the first feasible vertex. Since u_3 is infeasible and u_2 is feasible, the segment between them k_3 may be the intersecting segment. Note that the segment index in reverse order is one more than the vertex index. Next, V_L elements, from l_2 to u_0, are examined against k_3, which yields l_1. Therefore, the segment between l_1 and l_2 may be the intersecting segment. Next, the test against h_2 results in u_1. In the final test, one examines V_L elements against k_2 starting from l_1 (as l_2 was found infeasible in the previous iteration) and again l_1 is obtained. The iterations are terminated with the intersecting segments h_2 and k_2.

The implementation of the bottom intersecting method is presented in Algorithm A.7. I_B and

J_B are the outputs and correspond to indices of the intersecting segments of ∂R_L and ∂R_U, respectively.

In cases where both ends are closed, upper and lower intersecting algorithms are employed simultaneously. The intersecting segments of the top are reported by one, and those of the bottom by the other. The vertex list V is then updated with the insertion of $l_{\min}$ and $l_{\max}$. What is important here is to maintain the correct vertex order in V.

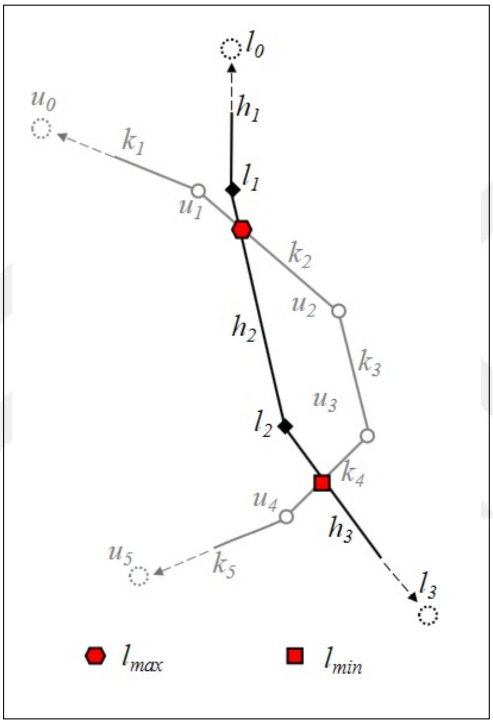

Figure 4.15. Both ends closed example

See the example of both ends being closed in Figure 4.15. The top intersecting algorithm outputs h_2 and k_2. The vertices that stays above h_2, l_0 and l_1, are removed from V_L. In the same way, the vertices above k_2 that are u_0 and u_1 are deleted from V_U. The bottom intersecting algorithm outputs h_3 and k_4. The vertices below these segments are removed from their corresponding vertex lists, l_3 and u_4, u_5. Thus, $V_L = \{l_2\}$ and $V_U = \{u_3, u_2\}$. Note that V_U is always reordered in reverse for the sake of preserving the ordering in V.

Subsequently, one intersects h_2, k_2 and h_3, k_4 to find $l_{\max}, l_{\min}$, respectively. The vertex list in desired order is formed as $V = \{V_L, l_{\min}, V_U, l_{\max}\} = \{l_2, l_{\min}, u_3, u_2, l_{\max}\}$. The cases of closed top-open bottom and open top-closed bottom are handled similarly.

Now, we look at a few particular examples to demonstrate that the proposed method excels in very special cases as well. However, some intersections may result in duplicate vertices. Moreover, some gives no feasible region $S = \emptyset$.

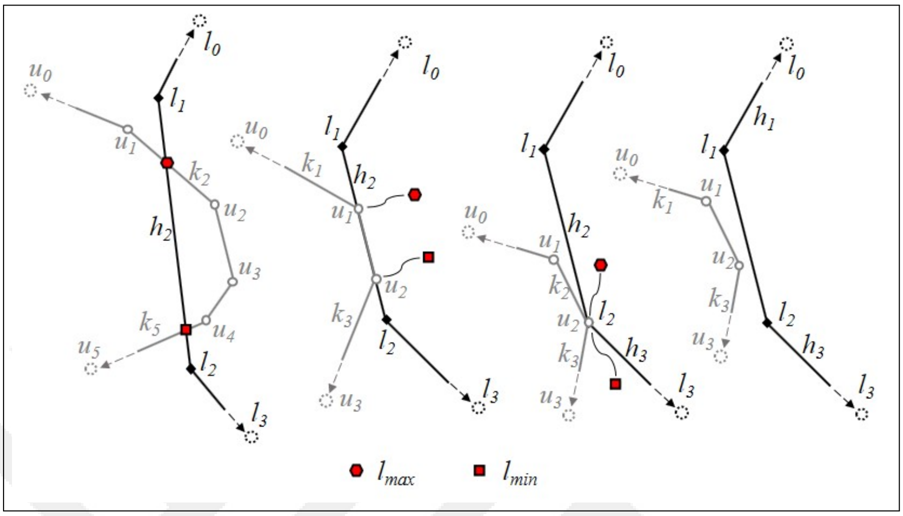

Figure 4.16. Four special intersection examples

In the first example demonstrated in Figure 4.16, both ends of ∂R_U intersect the same segment h_2 of ∂R_L. This also corresponds to the case in which ∂R_L has a single line segment, $N_L = 1$. The top and the bottom intersecting algorithms output h_2, k_2 and h_2, k_5, respectively. One removes the vertices that are above and below h_2 from V_L, namely l_0, l_1 and l_2, l_3, giving $V_L = \emptyset$. In the same manner, the elements of V_U that stays above k_2 and below k_5 are removed. Thus, $V_U = \{u_4, u_3, u_2\}$. Next, one determines the positions of $l_{\min}$ and $l_{\max}$, by intersecting h_2, k_2 and h_2, k_5, to add them into list $V = \{V_L, l_{\min}, V_U, l_{\max}\} = \{l_{\min}, u_4, u_3, u_2, l_{\max}\}$.

In the second example, one line segment from each chain is coincident. The algorithms return h_2, k_1 and h_2, k_3. After the removal of the vertices $V_L = \emptyset$ and $V_U = \{u_2, u_1\}$. $l_{\min}$ and $l_{\max}$ are determined and appended to V, thus $V = \{l_{\min}, u_2, u_1, l_{\max}\}$. Care must be taken since the first two and the last two in the list are duplicates. In this case, one should check the list and free from the duplicate vertices, $V = \{l_{\min}, u_1\}$.

In the next example, intersection is observed at a single point, necessarily a vertex. The outputs are h_2, k_2 and h_3, k_3. After elimination $V_L = \{l_2\}$ and $V_U = \{u_2\}$. Therefore, $V = \{l_2, l_{\min}, u_2, l_{\max}\}$, this time all duplicates. Deletion of the duplicates result in $V = \{l_2\}$.

In the last example, algorithms output h_3, k_3 and h_1, k_1, indicating no intersection. Therefore, one concludes that if the indices of top segments are greater than those of the bottom, then $S = \emptyset$ yielding $V = \emptyset$.

The proposed method only includes the cases in which the number of observations is more than one, $N_L > 0$ and $N_U > 0$. The remaining cases are clarified as follows.

i) $N_L = 0$ and $N_U = 0$ is a trivial case. S is represented by the full parameter space, $S = \mathbb{P}$. Therefore, an arbitrary point in $\mathbb{P}$ (origin) can be given as a vertex, $V = \{(0,0)\}$. Any pair of orthogonal vectors can define extreme rays, $D = \{\begin{bmatrix}1 & 0\end{bmatrix}^T, \begin{bmatrix}0 & 1\end{bmatrix}^T\}$.

ii) If $N_L = 0$ and $N_U > 0$, then $S = R_U$, therefore $V = V_U$ and $D = \{\overrightarrow{e_1}, \overrightarrow{e_2}\}$.

iii) If $N_L > 0$ and $N_U = 0$, then $S = R_L$, therefore $V = V_L$ and $D = \{\overrightarrow{d_1}, \overrightarrow{d_2}\}$.

The implementation of the proposed method is presented in Algorithm A.8.

4.3. LINEAR PROBLEMS WITH TWO PARAMETERS

In general linear optimization problems, a set of linear constraints determines S. Let the generic constraint be

$$a_{i0}\beta_0 + a_{i1}\beta_1 \leq b_i \tag{4.12}$$

where $a_{i0}, a_{i1}, b_i \in R$ are given variables in the i^{th} observation for $i = 1, \cdots, N$ and $\beta_0, \beta_1 \in R$ are the model parameters to be estimated. Although this model contains three variables (a_{i0}, a_{i1}, b_i), the dimensionality of the parameter space is still two (β_0, β_1). Therefore, one can express this in the form of a simple linear regression model by a simple division. In order to eliminate a_{i0}, Equation 4.12 is divided by a_{i0}, giving

$$\beta_0 + \frac{a_{i1}}{a_{i0}}\beta_1 \leq \frac{b_i}{a_{i0}} \quad \text{if} \quad a_{i0} > 0 \tag{4.13}$$

$$\beta_0 + \frac{a_{i1}}{a_{i0}}\beta_1 \geq \frac{b_i}{a_{i0}} \quad \text{if} \quad a_{i0} < 0 \tag{4.14}$$

which resembles $\beta_0 + \beta_1 x_j^U \leq U_j$ or $\beta_0 + \beta_1 x_i^L \geq L_i$. In this way, $\frac{a_{i1}}{a_{i0}}$ corresponds to x_i^L, x_j^U and $\frac{b_i}{a_{i0}}$ represents L_i, U_j. Therefore, one can classify these constraints as the lower and the upper bound constraints, respectively. These constraints form a feasible convex polygonal region in $\mathbb{P}$, if any, say $S \in \mathbb{P}$, with the set of vertices V and the set of the extreme rays D, both can be determined by the Algorithm A.8.

However, for any i, if $a_{i0} = 0$, Equation 4.12 becomes

$$\beta_1 \leq \frac{b_i}{a_{i1}} \quad \text{if} \quad a_{i1} > 0 \tag{4.15}$$

$$\beta_1 \geq \frac{b_i}{a_{i1}} \quad \text{if} \quad a_{i1} < 0 \tag{4.16}$$

which constraints β_1 from above or below in $\mathbb{P}$ by a horizontal line, a horizontal constraint. A new type of observation, since segments of $\partial R_L(\partial R_U)$ cannot be horizontal by Corollary 3. Note that if a_{1i} is also zero, then the observation should be excluded, since it is trivial by $0 \leq b_i$.

Consider $\beta_1 \leq \frac{b_i}{a_{i1}}$. If there are multiple constraints in this form, then the one with the smallest $\frac{b_i}{a_{i1}}$ value constraints β_1 from above. Let this horizontal line be $H^U \in \mathbb{P}$. The rest are redundant. Similarly, if the form is $\beta_1 \geq \frac{b_i}{a_{i1}}$, then the one with the greatest value of $\frac{b_i}{a_{i1}}$ will constraint β_1 from below. Let this horizontal line be $H^L \in \mathbb{P}$. Further, let $S_H \in \mathbb{P}$ be the region defined by H^L and H^U. Therefore, the final solution region is given by $S_P = S_H \cap S$.

Since determining H^L and H^U will take less than S, one first controls the emptiness of S_H and then the other possibilities.

1. If $S_H = \emptyset$, then there is no need to determine S, since $S_P = \emptyset$ for any S. In order $S_H = \emptyset$, H^U and H^L must exist and H^L must be above H^U. For all remaining cases S is to be determined.

2. If $S = \emptyset$, then $S_P = \emptyset$.

3. If $S_H = \mathbb{P}$, then $S_P = S$. A case with no horizontal observation.

4. If $S = \mathbb{P}$, then $S_P = S_H$. A case with no lower and upper observation.

5. If S_H and S are neither empty nor the full parameter set, then one is to intersect S_H and S, which can be analyzed by three sub-cases. Let V_P be the set of the vertices of S_P and let d_P be the set of the extreme rays of S_P.

i) If H^L exists H^U not, then one finds the feasible vertices of S that stay above H^L. If all vertices of S (including $l_{\min}$ and $l_{\max}$) are infeasible, then $S_P = \emptyset$. On the other hand, if they are all feasible, then $S_P = S$, therefore, $V_P = \{V_L, V_{\min}, V_U, V_{\max}\}$ in sequential order and $D_P = D$. Note that not all vertices of an open-top S can be infeasible due to l_0 and u_0. Note also that not all vertices of an open-bottom S can be feasible due to l_p and u_q.

Furthermore, if some vertices are feasible, then the intersecting segments, say h_i and k_j, are determined by identifying the segments between the feasible and the infeasible vertices. Then, h_i and k_j are intersected by H^L and new vertices are formed, say $l^L_{\min}$ and $l^U_{\min}$, respectively. Note that $l^L_{\min}$ and $l^U_{\min}$ are the vertices with the minimum slope now. Also, if S is closed top and $l_{\max}$ is on H^L, then $l_{\max}, l^L_{\min}$ and $l^U_{\min}$ will be duplicates, the two of which should be eliminated. Next, the feasible vertices are appended to V_P with the new vertices $l^L_{\min}$ and $l^U_{\min}$ in sequential order. Thus, $V_P = \{V^*_L, l^L_{\min}, l^U_{\min}, V^*_U, V_{\max}\}$ where V^*_L and V^*_U are reduced sets of V_L and V_U, respectively. Finally, d_P is determined by the existence of l_0 and/or u_0 in V_P. We will discuss this by the following example.

Assume that $V_L = \{l_0, l_1, l_2, l_3\}$, $V_U = \{u_0, u_1, u_2\}$, $V_{\min} = \{l_{\min}\}$, $V_{\max} = \emptyset$ and $l_3, l_{\min}, u_2$ are infeasible against H^L. Therefore, the intersecting segments are h_3 and k_2. In this case, one intersects h_3 and k_2 by H^L and finds $l^L_{\min}$ and $l^U_{\min}$. Thus, $V_P = \{l_0, l_1, l_2, l^L_{\min}, l^U_{\min}, u_1, u_0\}$. Since l_0, u_0 are in V_P, then one sets $D_P = \{\overrightarrow{d_1}, \overrightarrow{e_1}\}$ and removes l_0, u_0 from V_P. Thus, $D_P = \{\overrightarrow{d_1}, \overrightarrow{e_1}\}$.

Consider the same example with no upper bound observation. $V_P = \{l_0, l_1, l_2, l^L_{\min}\}$ and since no u_0 is observed, the set of extreme rays $D_P = \{\overrightarrow{d_1}, \begin{bmatrix} 0 & 1 \end{bmatrix}^T\}$. Conversely, if there was no lower bound observation, there would be no l_0 this time and D_P would be $\{\overrightarrow{e_1}, \begin{bmatrix} 0 & -1 \end{bmatrix}^T\}$. Additionally, if S is closed top, then $D_P = \emptyset$.

ii) If H^U exists H^L not, then one finds the feasible vertices of S that stay below H^U, and the procedure proposed in Case 2 is repeated in a similar way. This time the intersection gives $l^L_{\max}$ and $l^U_{\max}$. The extreme rays are determined by l_p and u_q.

iii) If H^L and H^U both exist, then the procedures proposed in Case 2 and 3 are applied simultaneously. Note that S_P is always bounded, thus, $D_P = \emptyset$.

5. A METHOD FOR MULTIDIMENSIONAL SOLUTION SPACE

This chapter focuses on exact description of the full solution set S that is defined by linear constraints with more than two variables. First, a method called edge tracking will be introduced. Its implementation will also be provided. Then, its performance will be compared by lrs and DDM.

5.1. THE EDGE TRACKING ALGORITHM (ETA)

Consider the following generic linear constraint with d variables.

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i \tag{5.1}$$

where a_{ij} (for $j = 1, \cdots, d$) and b_i represent the constraint coefficients in the i^{th} observation, for $i = 1, \cdots, N$, where N denotes the number of constraints. Further, $x_1, \cdots, x_d$ are the decision variables to be estimated. The set of constraints can be expressed in canonical form as

$$Ax \leq b \tag{5.2}$$

where

$$A_{N\times d} = \begin{bmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{Nd} \end{bmatrix} \quad x_{d\times 1} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \quad b_{N\times 1} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} \tag{5.3}$$

S refers to the set of all feasible x vectors that satisfy $Ax \leq b$. Let the space of x be denoted by $\mathbb{X} \subseteq \mathbb{R}^d$. Each constraint defines a closed half-space $\{x_1, \cdots, x_d \in \mathbb{X} | a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i\}$ in $\mathbb{X}$, the hyperplane of which $\{x_1, \cdots, x_d \in \mathbb{X} | a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d = b_i\}$ divides $\mathbb{X}$ into two parts, a feasible and an infeasible side. S is defined as the intersection of N half-spaces.

$$S = \bigcap_{i=1}^{N} \{x_1, \cdots, x_d \in \mathbb{P} | a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i\} \tag{5.4}$$

The image of S in $\mathbb{X}$ is a hyper-volume now, which is a convex *polytope* rather than a convex polygonal region. The complexity of the problem increases dramatically with the space dimension d.

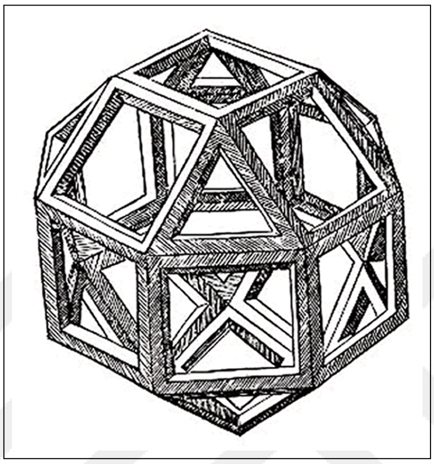

Figure 5.1. A skeletal polyhedron drawn by Leonardo da Vinci to illustrate a book by Luca Pacioli [1]

Definition of a polytope is not fully consistent across different resources. In certain fields of mathematics, the terms polytope and polyhedron are used in different senses. In this thesis we define a polytope as the generalization of three-dimensional polyhedra to any number of dimensions. A polytope is comprised of several dimensional components including vertices, edges, faces, cells, and so on. The ones we will utilize are listed with their dimensionalities as follows.

1. Facet with $d - 1$ dimension. Note that facets correspond to hyperplanes.
2. Edge with 1 dimension.
3. Vertex with 0 dimension.

If $d - 1$ hyperplanes are intersected in $\mathbb{X}$, then the dimensionality of the resulting element is reduced to one, $d - (d - 1) = 1$, giving an edge. Further, if d hyperplanes are intersected, then the dimensionality is reduced to zero, $d - d = 0$, which gives a vertex. For a system with N constraints, the maximum number of vertices that can be produced is $N!/(d!(N - d)!)$.

Among them, the ones that stay at the feasible side of N half-spaces will be feasible and therefore, the vertices of S. In other words, a vertex is feasible if it satisfies the remaining $N - d$ constraints.

The d constraints of a vertex can be expressed as $A^*x = b^*$ where $A^*_{d\times d}$ and $b^*_{d\times 1}$ are reduced matrices of A and b, respectively, that contain the corresponding d rows of A and b. The point coordinates $\overline{x}$ of the vertices are determined by

$$\overline{x} = A^{*-1}b^* \tag{5.5}$$

Note that this requires matrix inverse operation, hence, A^* has to be a non-singular matrix where $rank(A^*) < d$ indicates a degenerate case. Degenerate cases are the exceptional cases in which the object's (or a portion of it) regular dimension changes. For example, a triangle is an object of dimension two whereas a degenerate triangle that is contained in a line is dimension of one.

The degeneracy in our problem can be well illustrated by the coincident, parallel, etc. hyperplanes. In these cases at least two of the constraints in A^* are linearly dependent. Since one can simply determine and eliminate these dependent constraints as a prior stage of any method, the degenerate cases are excluded from our analysis. This exclusion makes S a simple polytope, meaning that each vertex of S is adjacent to exactly d vertices by d edges.

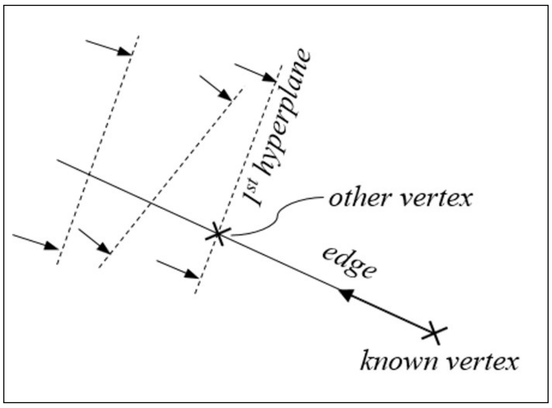

Figure 5.2. Demonstration of the first vertex encountered

Now consider two adjacent vertices of S, say $v_1, v_2 \in S$. They are linked each other by an edge, say $e_1 \in S$. One can reach v_2 by starting from v_1 and tracing e_1. Although there are finitely many hyperplanes in the direction of e_1, the first encountered one determines v_2 (see Figure 5.2). The reason for this is that the latter hyperplanes stay behind the first one ,hence, there can be no point on them satisfying the constraint provided by the first hyperplane. In order to determine the first hyperplane encountered the following series of operation are followed.

1. Get a hyperplane(constraint), say c_i.
2. Intersect c_i by e_1 and find a vertex.
3. Test the vertex against all constraints.
4. If the vertex is feasible then stop.
5. Otherwise, proceed with the next hyperplane c_{i+1} and return Step 2.

Note that the $d-1$ constraints of e_1 are common for v_1 and v_2, since they are on e_1. The remaining constraint of v_2 corresponds to the first encountered hyperplane.

Now, assume that v_1 is at the intersection of the first d constraints, $v_1 = \{c_1, c_2, \cdots, c_{d-1}, c_d\}$. Since S is a simple polytope, v_1 is connected to d vertices, say $v_2, v_3, \cdots, v_{d+1}$, by d edges, say $e_1, e_2, \cdots, e_d$, respectively. Each $d-1$ combination of $\{c_1, c_2, \cdots, c_{d-1}, c_d\}$ corresponds to an edge connected to v_1. Assume $e_1 = \{c_1, c_2, \cdots, c_{d-1}\}$. In order to determine the remaining constraint of v_2, one performs the operations given above.

In step one, c_i cannot be any of $\{c_1, c_2, \cdots, c_d\}$, since $\{c_1, c_2, \cdots, c_{d-1}\}$ are already included in e_1 and c_d defines v_1. In Step 3, there is no need to test the vertex against $\{c_1, c_2, \cdots, c_{d-1}\}$, since it is known that all will be satisfied. These eliminations are vital for the sake of the computational time.

Once v_2 is determined, the other $d-1$ vertices are explored in the same way. Call them primary vertices. Next, the search is continued by exloring the adjacent vertices of the primary vertices. Each primary vertex is connected to v_1 and other $d-1$ vertices. So, there are actually $d-1$ promising edges to be tracked. Therefore, $d(d-1)$ secondary vertices are obtained in the second iteration. Note that these edges are the feasible edges of S, since their one end is connected to a feasible vertex. If the intersection of these edges by the hyperplanes do not give a feasible vertex (an open end), then it is an extreme ray of S.

In continuation of the iterations, the new links(edges) multiply very rapidly at first. Afterwards, it is seen that some of the new links are connected to the vertices that are already found. These links are not tracked and the growth rate of the number of new links in each iteration starts to decrease. At some point, the number of new links becomes constant and then starts to decrease. Eventually, a point is reached where all new links begin to produce vertices that are already in the vertex list. The iterations are terminated at this point.

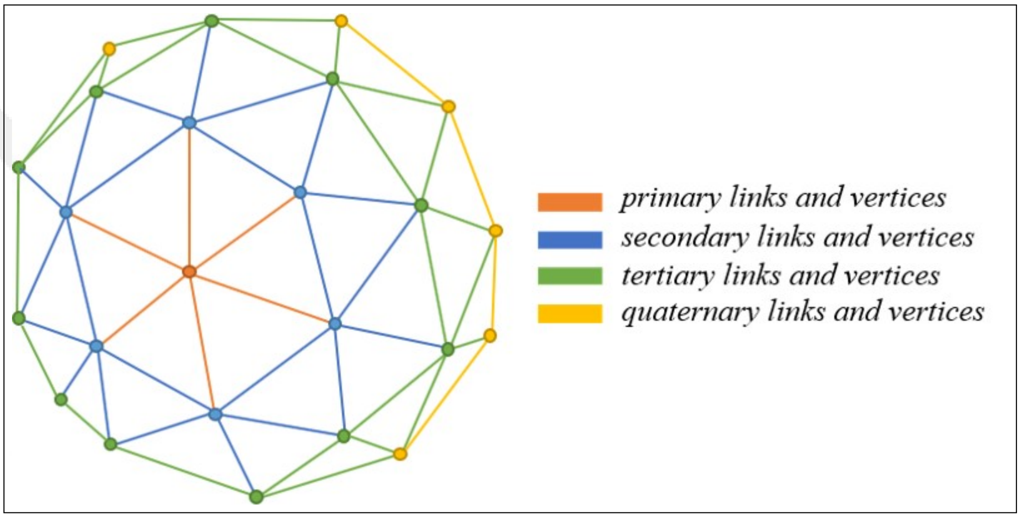

Figure 5.3. An illustration of vertex enumeration by ETA

The mathematical operations of the proposed method are explained by the following example. Consider a set with 8 constraints where $d = 3$. Let A_i and b_i correspond to the i^{th} rows of A and b, respectively. Therefore, a constraint is denoted by $c_i = \{A_i, b_i\}$. Let the initial vertex be $v_1 = \{\{A_1, b_1\}, \{A_3, b_3\}, \{A_4, b_4\}\}$. Thus, the primary edges are $e_1 = \{\{A_1, b_1\}, \{A_3, b_3\}\}$, $e_2 = \{\{A_1, b_1\}, \{A_4, b_4\}\}$, and $e_3 = \{\{A_1, b_1\}, \{A_4, b_4\}\}$. The first iteration starts by the first edge. e_1 is intersected by c_2 where the coordinates of the resulting vertex is determined by

$$\overline{x} = \begin{bmatrix} A_1 \\ A_3 \\ A_2 \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_3 \\ b_2 \end{bmatrix} \tag{5.6}$$

Note that the first two rows are fix and represent e_1. The vertex is tested against c_i for

$i = 4, 5, 6, 7, 8$ by

$$0 \leq b_i - A_i\bar{x} \tag{5.7}$$

If the inequality above holds for all i values, then the vertex is feasible. Assume that it is infeasible. The next hyperplane is $\{A_4, b_4\}$, but it is skipped since $\{\{A_1, b_1\}, \{A_3, b_3\}, \{A_4, b_4\}\}$ defines v_1. Therefore, one continuous with $\{A_5, b_5\}$ and repeats the same process. Assume this time it is feasible, therefore $v_2 = \{\{A_1, b_1\}, \{A_3, b_3\}, \{A_5, b_5\}\}$ registered. The iteration proceeds with e_2 and e_3 and ends up by 3 primary vertices, each connected to 2 secondary vertices.

Note that since $\{\{A_1, b_1\}, \{A_3, b_3\}\}$ would yield $v_1 = \{\{A_1, b_1\}, \{A_3, b_3\}, \{A_4, b_4\}\}$ again, the promising edges of v_2 are $\{\{A_1, b_1\}, \{A_5, b_5\}\}$ and $\{\{A_3, b_3\}, \{A_5, b_5\}\}$ for the second iteration. In other words, one end of e_1 is connected to the given vertex v_1 and the other end is already found to be connected to v_2. Since e_1 cannot be connected to another vertex, it should not be tracked in the future iterations. Therefore, one should keep the record of the edges that are tracked before. In this way, the edges of a newly explored vertex are compared by the previously processed ones. The ones that are already in the list are skipped. When there is no new edge to follow, iterations are terminated.

Algorithm A.9 is first introduced in order to simplify the implementation of an algorithm that uses the proposed method to find vertices V and extreme rays D of S. This subroutine accepts a vertex v, arrays of A and b, and the tracked edges list E and outputs the new vertices connected to the given vertex v and updated form of E.

The main algorithm of the proposed method is implemented in Algorithm A.10 where v_i denotes the initially given vertex.

5.2. PERFORMANCE COMPARISON OF ETA

The competitors are again lrs and DDM. T_c performances of the algorithms are tested in MATLAB by synthetically generated datasets where experimental design criteria are N and d. N is assigned 9 different values ranging between 10 to 500 where d takes 5 different values between 2 and 6, providing 54 experiments in total. Each experiment is repeated 100 times in order to better observe distribution of T_c values of the algorithms. For a better comparison, the ratios of median T_c values of lrs and DDM to that of ETA are demonstrated in Figure 5.4.

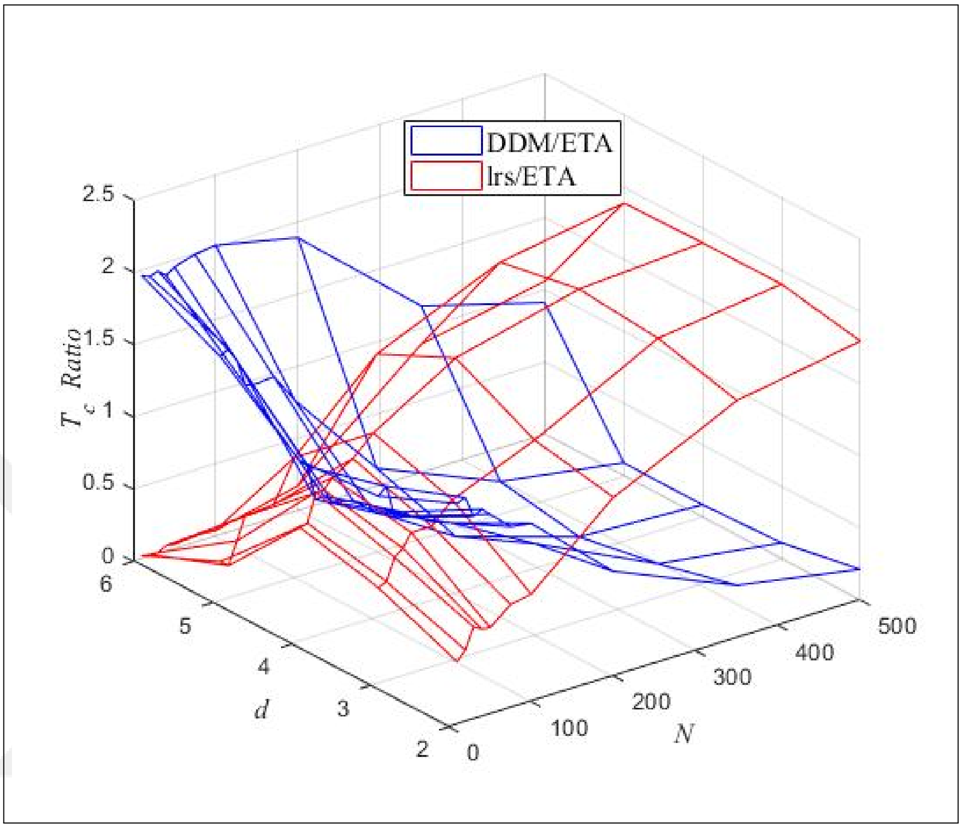

Figure 5.4. Performances of lrs and DDM with respect to ETA

When comparing the performance of ETA and DDM, it is observed that ETA outperforms DDM for small values of N. Specifically, when $N = 10$ and $d = 6$, the performance of ETA is found to be twice that of DDM. However, as N increases to approximately 200, the performance ratio between the two algorithms is equalized. For N values greater than 200, the performance of DDM becomes better, with the degree of superiority depending on the value of d. It should be noted that the influence of d on performance cannot be ignored, as increasing d generally results in an improvement in the performance ratio for a majority of N values. In conclusion, it can be stated that while DDM performs better for high N values, ETA performs better for high d values.

The comparison between the performances of lrs and ETA reveals a roughly inverse relationship. For N values up to approximately 200, lrs provides a significant advantage over ETA. As N increases, ETA becomes the better performer, with the ratio reaching 2 for the largest N value considered. The effect of d on the performance ratio is variable, with a decrease observed for d values of 2, 5 and 6. However, the impact of d on the performance ratio may not be straightforward to interpret, as the ratio is larger for other values of d. Despite this, it can be concluded that ETA performs better than lrs for high N values.

6. MECHANICAL ENGINEERING APPLICATIONS

This chapter presents two engineering applications. The first application investigates how to utilize the full solution set S to obtain the best regression estimates. In order to provide a more comprehensive regression tool, the exact description of S for the multiple linear regression model is studied, as it was done for the simple linear model with bounded error, presented in Chapter 3. The performance of this tool is then evaluated against well-known regression techniques in both real-world and synthetically generated experiments, which are undertaken separately for both models.

The second application involves conducting a geometric tolerance analysis on an assembly flange model, which is comprised of two components with pin and hole connections. The geometry of the assembly is first modeled in such a manner that the pins are compatible with the holes, ensuring a functional assembly. Subsequently, virtual simulations of the model are performed using various tolerance specifications, in order to assess the impact of tolerances on the assembly.

6.1. THE REGRESSION APPLICATION

Consider the following multiple linear regression model.

$$y_i = \beta_0^* + \beta_1^* x_{i1} + \cdots + \beta_p^* x_{ip} + \varepsilon_i \tag{6.1}$$

where y_i and x_{ij}, for $j = 1, \cdots, p$, are respectively response and predictor variables measured in the i^{th} observation ($i = 1, \cdots, N$). $\beta_k^* \in R$ are the true model parameters to be estimated ($k = 0, \cdots, p$) and $\varepsilon_i \in [\varepsilon_i^L, \varepsilon_i^U]$ denotes the unknown but bounded error. Again, for each observation the true response is constrained by two inequalities.

$$\beta_0^* + \beta_1^* x_{i1} + \cdots + \beta_p^* x_{ip} \geq y_i - \varepsilon_i^U = L_i \tag{6.2}$$

$$\beta_0^* + \beta_1^* x_{i1} + \cdots + \beta_p^* x_{ip} \leq y_i - \varepsilon_i^L = U_i \tag{6.3}$$

where L_i, $U_i \in R$ represent the lower and the upper bound points, respectively. Further, this time the uncertainty corridor is multidimensional and $\beta_0^* + \beta_1^* x_{i1} + \cdots + \beta_p^* x_{ip}$ represents the true hyperplane which must be in the corridor, assuming that the error bounds and measurements conform to the model. Therefore, the corridor condition is given by

$$L_i \leq \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} \leq U_i \tag{6.4}$$

where $[\beta_0, \beta_1, \cdots, \beta_p] \in R^{p+1}$ represent all feasible regression parameter vectors that constitutes S, the image of which in $\mathbb{P}$ is a bounded $(p+1)$-polytope.

Further, higher degree polynomial regression models can be expressed in the form of the multiple linear regression model, since, although this model is not linear in its variables, it is linear in its parameters. Consider the following n- degree polynomial regression model.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_n x_i^n \tag{6.5}$$

If one takes

$$x_{i1} = x_i, x_{i2} = x_i^2, \cdots, x_{in} = x_i^n \tag{6.6}$$

Equation 6.5 becomes identical with the multiple linear regression model. For both models, one can employ ETA to find all possible parameter vectors, the one of which is known to be the true solution.

6.1.1. The Centroid of S as the Best Estimate

Since we consider that there is no information about the error except the error bounds, all elements of S have the same probability of being the true solution. However, a single solution is to be returned as the result of the regression analysis. At this point, a particular point that is more likely than others to be the closest to the true solution may be found by examining many candidates. Here, we design an experiment in computer environment that makes use of the simple linear model since it is extremely challenging to determine most of the special points for higher dimensional polytopes. The special points for S as the convex polygon are listed as follows.

1. The midpoint of the diagonal where the diagonal is defined as the line segment that connects $l_{\min}$ and $l_{\max}$.
2. The centroid of the polygon.
3. Mean of the vertex coordinates.
4. Geometric median of the vertices, which corresponds to the point minimizing the sum of distances to the vertices.
5. Medoid of the vertices, which is a vertex minimizing the sum of distances to other vertices.
6. Chebyshev center of the polygon, which is the center of the largest inscribed circle of the polygon.

Note that due to the convexity of S, all these points are guaranteed to be in the solution region. The performances of the candidates are evaluated by several simulations. In each simulation, first, the true line parameters (β_0^*, β_1^*) are generated randomly, then, a set of N random x values is generated. Next, $y_i = \beta_0^* + \beta_1^* x_i + \varepsilon_i$ is calculated for $i = 1, \cdots, N$, where ε_i is randomly generated in the interval of the predetermined bounds $[\varepsilon^L, \varepsilon^U]$. In this way, a set of x and y values that conform to the simple linear model and error bounds is obtained where the true model parameters are known. The simulations are then repeated with different N and $[\varepsilon^L, \varepsilon^U]$ values where the distribution type of ε_i also varies in different simulations.

Table 6.1. Regression best estimate candidate performances

Candidate	Mean of the normalized MRSS	Standard deviation of the normalized MRSS
The diagonal midpoint	0.063	0.120
The centroid	0.058	0.118
Mean of vertices	0.061	0.126
Geometric median	0.062	0.128
Medoid	0.232	0.443
Chebyshev center	0.079	0.183

After the calculation of the bound points L_i, U_i with respect to $[\varepsilon^L, \varepsilon^U]$, the dataset becomes ready to be input to CM that will give the polygon S. Finally, the coordinates of each candidate (β_0, β_1) are calculated as the estimates of the regression analysis. In order to determine the best candidate, we employ a metric that slightly differs from the residual sum of squares (RSS). Since the true model is known in our simulations, the total error of each estimate can be better determined with respect to the true model as follows

$$MRSS = \sum_{i=1}^{N}(\beta_0^* + \beta_1^* x_i - (\beta_0 + \beta_1 x_i))^2 \tag{6.7}$$

where $MRSS$ stands for modified residual sum of squares. In order to see the overall performances in the simulations, one averages the $MRSS$ values after normalization. The overall performances of the candidates are presented in Table 6.1.

The centroid comes forward by a slight difference whereas the medoid shows the worst performance. This is reasonable since its estimate is always on the boundary. Therefore, one can conclude that the true solution has a higher probability of being at the center of the polygon, which can also be verified by the other candidates. Note that for higher dimensional polytopes, determining the centroid is not only challenging but also time consuming. For this reason, one may consider to use mean of vertices that only requires basic algebraic operations.

6.1.2. Other Techniques As Competitors

Numerous methods have been developed for parameter estimation in linear regression. These methods differ in computational simplicity of algorithms, presence of a closed-form solution, robustness with respect to heavy-tailed distributions, and theoretical assumptions needed to validate desirable statistical properties such as consistency. One can classify these methods as:

1. Least-squares estimation and related techniques: Ordinary least squares (OLS), weighted least squares, generalized least squares.

2. Maximum-likelihood estimation and related techniques: Maximum likelihood estimation (MLE), ridge regression, least absolute deviation regression.

3. Other estimation techniques: Bayesian linear regression, principal component regression, least-angle regression, the Theil–Sen estimator, interval valued regression methods, linear regression using gradient descent (GDR), and so on.

Recall that from the early beginning of this study, it is assumed that no error information is available except the bounds of the error. Since the generated datasets are known to conform to the model and the error bounds, the following assumptions are considered in the synthetic experiments.We will investigate the listed methods in this context.

- The datasets have no outliers.

- The number of model variables are known so that underfitting and overfitting are not to be considered.

- All predictor variable values are generated randomly so that no correlation among them can be observed.

Consider the first regression group, OLS and its derivatives, the only viable of which seems to be OLS, considering our synthetic simulations. Undoubtedly, OLS is the most widely used regression techniques and the reasons for that can be listed as follows.

i) OLS estimates are unbiased: In other words, the expected value of the OLS estimate is equal to the true value of the parameter being estimated. This is an important property because it means that, on average, the OLS estimates will be close to the true values of the parameters.

ii) OLS estimates are consistent: As the sample size increases, the OLS estimates converge to the true values of the parameters. This is also an important property because it means that the estimates will become more accurate as more data is collected.

iii) OLS estimates are efficient: Among all unbiased estimators, the OLS estimates have the smallest variance. This means that they are less sensitive to random fluctuations in the data, and are therefore more reliable.

iv) OLS estimates have a closed-form solution: This means that there is a simple formula for calculating the OLS estimates, which makes them easy to compute.

Therefore, OLS is determined as the first competitor. Now, consider the the second group. The ridge regression deliberately incorporates bias into the estimation of the parameters. If multicollinearity is present or overfitting is an issue, the resulting estimates typically have lower mean squared error than the OLS estimates. However, this will not be the case in our experiments. On the other hand, the least absolute deviation regression is less efficient than OLS when no outliers are present. Hence, MLE remains as the only candidate.

MLE requires the PDF of the error distribution. At this point, one may assert to use normal distribution, as we will explain shortly, as this is justifiable in many real-world situations where regression analysis is applied. The measurement errors can be classified as random errors that include measurement device resolution, operator bias, environmental factors, etc. With this definition, it can be stated the error is composed of a linear sum of errors from various sources. The Central Limit Theorem comes into play here.

The theorem states that, in many situations, when independent random variables are added, their properly normalized sum tends toward a normal distribution even if the original variables themselves are not normally distributed. In other words, even though the individual constituent errors or deviations may not be normally distributed, the combined error is approximately so. This can also be better comprehended by a simple example.

Consider a dice game played by rolling two dices simultaneously and the sum of the two outcomes is recorded. Each die has six identical faces numbered by integers from one to six. Therefore, each integer has the same probability of coming and the expected distribution is uniform (rectangular). However, when the sums are calculated as outcome, the distribution becomes normal. This explains why the outcome 7 is very common in two dice rolling games.

One may also consider the uniform distribution for the error distribution, a well-known application of which is digital resolution uncertainty. Digital resolution uncertainty originates from the resolution of a digital sensor. For instance, consider a sensor capable of indicating a three-digit readout. If the reading is 32.115, it is known that, considering the device employs the standard round-off practice, the displayed number is derived from a sensed value that lies between 32.1145 and 32.1155. Also, it can be asserted to a very high degree of validity that the value has an equal probability of being between these two values.

Since the probability density function of the uniform distribution is a flat line which means that all response values have the same probability to occur, the density values of responses are equal to each other in the limits of the distribution. In this case, likelihood function yields the same value for any N observations and, therefore, loses its functionality. This can also be given as another reason to comprehend employment of normal distribution in general. However, use of the normal distribution in MLE results in OLS and for this reason, we will not include MLE in our analysis.

The ridge regression is similar to OLS, but it enhances the objective function with a penalty term to lower the coefficient estimates to zero. This can lessen the risk of overfitting and increase the model's generalizability. Since the number of variables are known in our experiments, we also skip this method. Further, least absolute deviation regression is less sensitive to the outliers than OLS whereas it is less efficient than OLS in presence of no outliers.

Consider the last group. The Bayesian linear regression assumes the parameters to be random variables with a specified prior distribution. In addition, the Bayesian estimation process produces not a single point estimate, rather an entire posterior distribution. The next is principal component regression employs principle component analysis when strong correlations exist among the predictor variables. The correlations are first identified by principle component analysis and redundant variables are eliminated. Then, the procedure proceeds with OLS to find the parameter estimates. Least-angle regression was developed to tackle high-dimensional covariate vectors that may contain more covariates than observations. The next candidates are interval valued regression methods. In this context, three well-known candidates are examined. These are Center Method [42], MINMAX Method [43] and lastly, Center and Range Method [44].

All three methods consider error in both variables, namely, uncertainties in both x and y variables. However, in the present study, we only consider uncertainty in the predictor variable, thus requiring the elimination of the response uncertainty term from these models. As a result, all three methods revert back to ordinary least squares (OLS), and thus will not be considered further in this study.

Next, Theil-Sen estimator is a simple linear regression technique for robustly fitting a line to sample points in the plane by selecting the median of all line slopes via pairs of points. The slope β_1 is calculated by the median of the slopes $(y_j - y_i)/(x_j - x_i)$ ($i \neq j$ and $i, j = 1, \cdots, N$) determined by all pairs of sample points. Once the slope is determined, β_0 is determined by the median of the values $y_i - \beta_1 x_i$. Further, the Theil-Sen estimator is a non-parametric method whereas OLS is parametric. However, Theil-Sen estimator is not functional for multidimensional cases.

In the context of linear regression, the gradient descent method can be used to find the values of the coefficients that best fit the data. In order to find the optimal coefficients the process starts by initializing the coefficients to some random values. Then, one iteratively updates the coefficients in the direction that reduces the error between the predicted values and the observed values. The error is typically measured using a loss function, such as mean squared error, which calculates the average squared difference between the predicted and observed values.

At each iteration, the gradient of the loss function with respect to the coefficients is computed and used to update the coefficients in the opposite direction. The process continues until the coefficients converge to values that minimize the loss function. The algorithm of this method is given as follows:

1. Compute the predicted values of the dependent variable using the current values of the coefficients.

2. Compute the error between the predicted values and the observed values.

3. Compute the gradient of the loss function with respect to the coefficients.

4. Update the coefficients using the gradient and a learning rate (a value that determines the size of the step taken in the direction of the gradient).

5. Repeat the process until the coefficients converge to values that minimize the loss function.

Undoubtedly, GDR requires more computational time and may not be as efficient as OLS. However, it does not require some assumptions that may not hold in all cases i.e., OLS relies on a normally distributed random errors. In addition, GDR does not provide a closed-form solution and requires iterations to converge to the best estimate. At this point, one may asses GDR as a good alternative to OLS. However, the number of operations and the step size in each iteration determines the accuracy of the method. These parameters may vary in a large scale.

Using appropriate values for those parameters, we first tested GDR versus OLS in the synthetic tests. OLS outscored GDR in every simulation, leading us to run more iterations and smaller steps. In this way, despite a significant increase in processing time, an accuracy that is comparable to OLS was achieved. Since GDR estimates still have some room for improvement over OLS estimates at this level of accuracy, we forego this approach as well because it would require processing time hundreds of times longer than OLS to achieve roughly the same accuracy. This explains why GDR is not included as a competitor in this study.

As a result of the examination, a number of candidates were evaluated against the proposed method. Ordinary Least Squares (OLS) emerged as a clear front-runner due to its distinctive features. Furthermore, most of the other candidates were eliminated as their estimates were equivalent to the OLS estimates under certain assumptions, such as Maximum Likelihood Estimation (MLE) with the assumption of a normal distribution. Additionally, some of the candidates were derived from the OLS model by incorporating additional features. However, given the assumptions made for the present case, these additional features were unnecessary, resulting in OLS being the only remaining candidate in the experiments.

6.1.3. Synthetic Experiments

The performances of the OLS and the proposed method are evaluated in this section. Both provides fast results and hence, the computational times in this part are not considered. What is considered here is the accuracy of the estimates. Since the true model parameters are known in each trail, we employ normalized MRSS again to represent the accuracy of the method. The accuracy of each method is calculated and recorded over a large set of simulations. In these simulations, the experimental design criteria are N, d, $\varepsilon_{\max}$ and the distribution type of the random error term ε_i. Here, $\varepsilon_{\max}$ represents signal to noise ratio.

Employing a multiplicative error term rather than additive can be explained by the nature of system used in the real-life experiments where the most of the sensor accuracies are defined by a percentage term. Although there are very accurate devices that have accuracies of less than $\%0.1$, our goal here is to develop a method that can also work with devices of low accuracy ($\%1 - \%3$). It is clear that the former would obviously provide very good results, anyway.

The distribution types will cover the normal and the uniform distributions, for the reasons and examples already discussed. The third type can be a skewed distribution where the limits are asymmetric for the most of the cases. For example, consider the manufacturing process of an inner (or outer) ring of a bearing. This part is circular and manufactured by turning process. Its outer diameter is defined by a tolerance since it is not possible to have an exact value for a dimension in real world. Assume that the nominal dimension is 98 mm where the limits are specified by a symmetric tolerance ± 0.05 mm. Therefore, the dimension after the manufacturing has to be between 97.95 mm and 98.05 mm.

An extruded raw material with a circular profile is suitable for this manufacturing. These extruded rods are manufactured in some certain dimensions called basic sizes. As turning process is based on removing material, a basic size that is greater than the upper tolerance limit is selected.

Assume that the rod is selected as it has a basic size of 100 mm. The second assumption is that the ring is manufactured by a human controlled turning machine. In this case, at least three passes (considering time efficiency) are generally required in which the first one is called rough pass and the others are intermediate and finish passes, respectively. The operator usually tries to approach a value close to (greater than) the upper limit 98.05 mm by the first two passes. In the last pass, the operator has to put the dimension in the tolerance limits.

Care must be taken during this operation, since, if excessive amount of material is removed then the part cannot be used any longer. In order to manage the risk, the operator tries to remove material as less as possible but still guaranteed to be in the limits. Hence, the final dimension mostly stays on the positive side of the nominal value (greater than the nominal value and close to the upper limit). It is obvious that it causes the distribution to be skewed that can be represented by Weibull distribution.

In the simulations, N ranges between 10 and 50, whereas d is assigned to be between 2 and 6. On the other hand, $\varepsilon_{\max}$ is assigned the values $\%1$, $\%2$ and $\%3$. The normal, the uniform, and the Weibull distributions are used to generate the random error. In this way, the performances of both methods are examined over a total of $5 \times 5 \times 3 \times 3 = 225$ simulations. Each simulation is repeated 100 times and overall performances of both methods are recorded using the means of normalized MRSS values. Since the differences can be large, in order to better comprehend the performances we preferred to use a ratio, namely, the performance ratio of the OLS to the proposed method. Note that a ratio of 1 indicates the same performances whereas less than 1 and greater than 1 indicate a worse and better performance of the proposed method, respectively.

The datasets in each simulation are generated by the following operations.

1. Depending on N and d values a set of random x values generated.

2. Random true parameters β_j^* are then generated.

3. True response values are computed by $y_i^* = \beta_0^* +_{j=1}^{d} \beta_j^* x_i$ for $i = 1, \cdots, N$.

4. Random error term ε_i for each observation is generated with respect the distribution

type and $\varepsilon_{\max}$. The noisy response values are calculated $y_i = y_i^* + \varepsilon_i$.

Determination of ε_i is as follows. Let $r \in \mathbb{R}$ be a random variable and bounded by -1 and 1. Therefore, $\varepsilon_i = y_i^* \varepsilon_{\max} r$, where the only remaining control parameter is the distribution of r. MATLAB has a random number generator function called "random". This functions requires an object that defines the distribution as input. In order to create this probability distribution object, makedist function of MATLAB can be used.

1. If the distribution is normal, then a normal distribution object with $\mu = 0$ and $\sigma = 1$ is defined for the makedist. This is the simplest case and known as the standard normal distribution. Then, it is modified to be truncated where the tails of the distribution are located at -1 and 1 so that $r \in [-1, 1]$.

2. If the distribution is uniform, then an uniform distribution object with $[-1, 1]$ limits is defined for the makedist.

3. The probability density function of a Weibull random variable is

$$f(r) = \begin{Bmatrix} \frac{k}{\lambda}(\frac{r}{\lambda})^{k-1} e^{-(r/\lambda)^k} & r \geq 0 \\ 0 & r < 0 \end{Bmatrix} \tag{6.8}$$

where $k > 0$ is the shape parameter and $\lambda > 0$ is the scale parameter of the distribution.

We assign these values as $k = 1.483$ and $\lambda = 4.459$, both are attained from an example bearing life calculation [45]. The normalized PDF with the specified parameters is plotted in the Figure 6.1.

In order to make r bounded between -1 and 1, the following scaling and shifting operations are performed.

$$r = 2\left(\frac{r^*}{\Delta r} - \Delta s\right) \tag{6.9}$$

In this formula, r^*denotes the random variable generated under the Weibull distribution with the specified k and λ values, Δr and Δs are respectively the scaling and shifting factors. $\Delta r = 28.15$ and $\Delta s = 0.5$ provides r to be distributed in the desired interval.

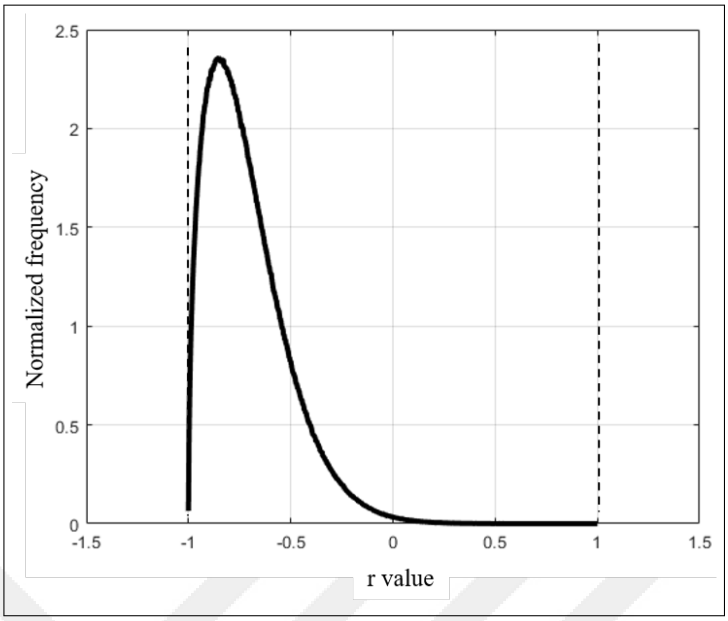

Figure 6.1. Normalized probability density function of the bearing life example

The performances of the methods for 225 different experiments are demonstrated in Figures 6.2 to 6.5. The first three figures show the performances under different distributions and sensor accuracy levels whereas the last figure shows the effect of the distributions on the same graph where the sensor accuracy is fixed to $\varepsilon_{\max} = 0.02$.

In every simulation, the proposed technique has been found to be unquestionably better than OLS. The following interpretation can be drawn about how design parameters affect the performances.

- The number of model parameters d has negligible impact. Even though the proposed method is always better, the performance gap changes slightly as d is increased. Therefore, one can conclude that the performances are not significantly impacted by various models.

- The impact of the number of observations N is striking. The performance of the proposed method compared to OLS clearly improves when N is increased. Therefore, it can be said that the greater N, the better performance of the proposed method compared to OLS. This is reasonable since it is expected that the volume of the solution polytope gets smaller when additional constraints are added to the system. When the polytope gets smaller, the estimates of the proposed method gets closer the to the true solution

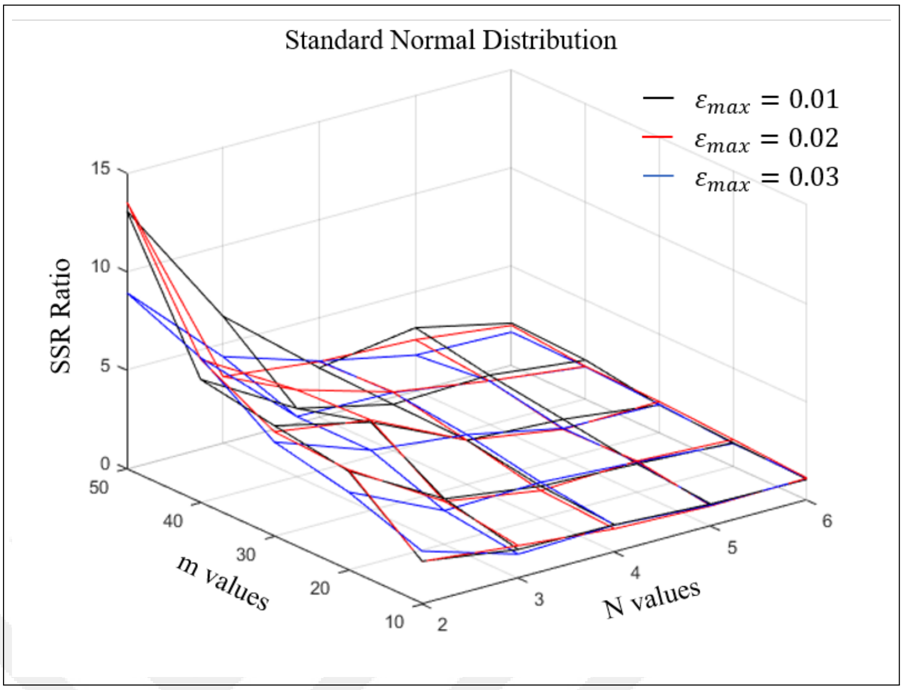

Figure 6.2. Performance comparison under normal distribution

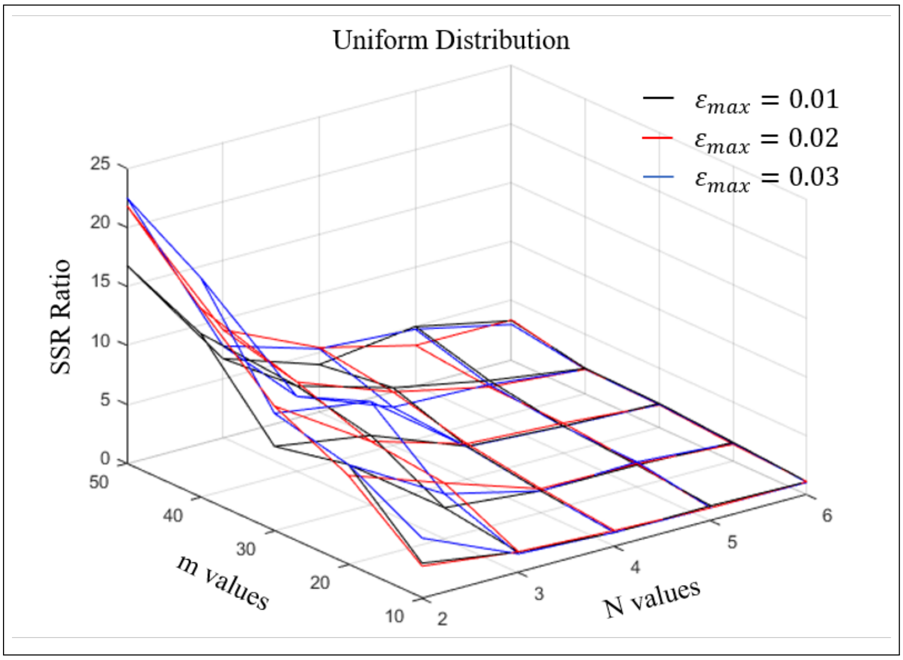

Figure 6.3. Performance comparison under uniform distribution

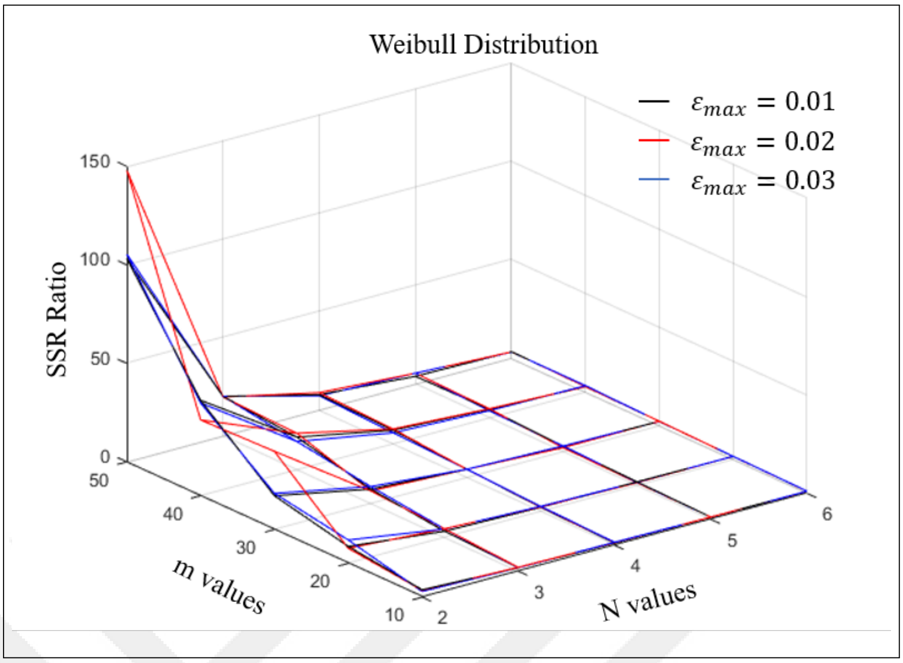

Figure 6.4. Performance comparison under weibull distribution

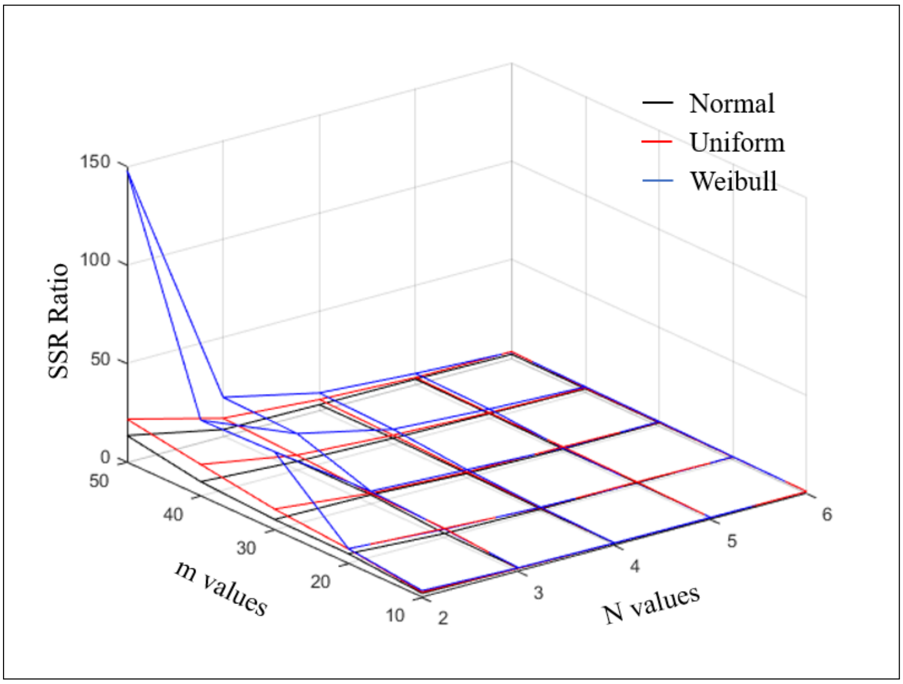

Figure 6.5. Performance comparison under different distributions for a fixed accuracy value of $\varepsilon_{max} = 0.02$

since it is known that the true solution must be in the solution region.

- The sensor accuracy $\varepsilon_{\max}$ has also an inevitable impact. It has been shown that the performance of the proposed method improves as $\varepsilon_{\max}$ increases. This once again underscores the importance of using the proposed method. Utilizing very accurate sensors can improve the accuracy of OLS estimations, but the proposed method can achieve the same or even higher degree of accuracy using low-cost and accessible sensors.
- The distribution type has the most significant impact. For example, the maximum valued R is 13.54 for the normal distribution whereas it is 22.51 of the same N and d values for the uniform distribution. The difference becomes even larger for the Weibull distribution. This is expected as the distribution is skewed and biased. Therefore, it is expected that the performance difference gets larger when the distribution slides away from the normal distribution. This is a unique feature in the cases the true distribution cannot be foreseen.
- In addition to all, it is observed in some simulations the estimates of the OLS falls outside the solution region where it is impossible for the estimate to be the true solution. Therefore, one may consider to use the proposed method as a benchmark solution for the OLS. Further, the rate of estimates that falls outside the solution region increases if lower sensor accuracy is used, e.g., $\%12$ of the estimates were observed to be outside for $\varepsilon_{\max} = 0.01$, whereas this value can reach approximately up to $\%18$ for $\varepsilon_{\max} = 0.03$.

6.1.4. Real Experiments

This part consists of two separate experiments. In both, the same experimental setup is used with minor modifications. In the first experiment, the position of a body that translates on a straight path, say s, is recorded by an low-cost ultrasonic distance sensor with respect to time t (see Figure 6.6). The true position of the target is controlled by a very precise mechanism the details of which will be presented later. By providing constant velocity, the t versus s curve is a line that corresponds to the simple linear case in which the sensor data are the noisy observations and true model parameters are controlled by the experimental setup.

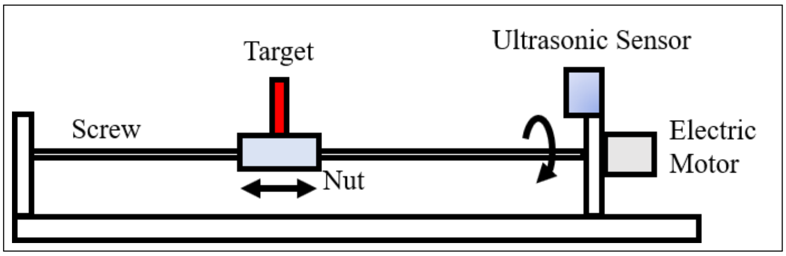

Figure 6.6. An illustration of the first experimental setup

In the second experiment, the vertical position of a free falling body is measured by the same low-cost ultrasonic distance sensor where the experimental setup is used in vertical position (see Figure 6.7) and separated from the components that controls the motion. The true model is a well-known quadratic equation when air resistance and the frictions on the setup are ignored. This model corresponds to a second degree polynomial regression model that is included by the multiple linear model as demonstrated before.

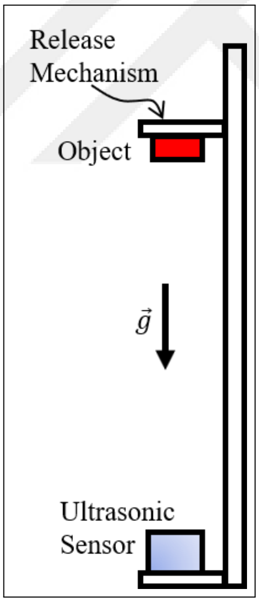

Figure 6.7. An illustration of the second experimental setup

6.1.4.1. The Simple Linear Case

A precision motion control and measurement system has been designed to provide one degree of freedom to a target body (see Figure 6.8). The system consists of a carriage that moves along two metal rods using linear bearings, and a stepper motor that controls the rotation of a screw which drives the motion of the carriage through a M4 nut. The position of the target body, mounted on the carriage, can be related to the angular position of the motor with sufficient precision. An ultrasonic distance sensor is fixed to the system to measure the distance of the target body.

The Arduino UNO is a microcontroller board that features the ATmega328P microcontroller and has 14 digital input/output pins and 6 analog inputs. In the described setup, the Arduino UNO is utilized as both a data acquisition tool and a motor driver simultaneously. Three of the digital input/output pins are reserved for sensor use, while four of them are used to drive the motor. The power for both the Arduino UNO and the motor is supplied through the $+5V$ and ground (GND) pins on the board.

The HC-SR04 ultrasonic digital sensor is used in the experiment as a distance sensor. This sensor is Arduino compatible [46] and is chosen for its low cost and low accuracy in order to demonstrate the performance of different regression methods with low accuracy measurements. It is expected that the proposed method will be able to provide accurate model parameter estimates using these low accuracy measurements, and that it will perform even better with high accuracy measurements. Additionally, it is expected that the proposed method will show significant improvement compared to OLS when used with low accuracy measurements. By demonstrating the effectiveness of the proposed method with low accuracy measurements, it is expected that the method will be able to provide even better results with higher accuracy measurements.

The ultrasonic digital sensor used in the experiment is specified by the manufacturer to have an accuracy of ±3 mm. However, through testing and calibration procedures, the actual accuracy of the sensor was found to be ± 3.8 mm. In order to evaluate the performance of the proposed method, the manufacturer-specified accuracy of ±3 mm was used, rather than incorporating any additional information obtained through calibration processes. This decision was made to demonstrate the effectiveness of the proposed method using the information provided, which we have considered from the beginning of this study, rather than relying on additional information obtained through calibration.

The stepper motor used in the experiment is the 28BYJ-48 [47] model, which is driven by an Arduino-compatible Uln2003 driver. This motor has a high resolution of 512 steps per revolution, which allows for precise linear displacement when coupled with the M4 screw lead of 0.7 mm.

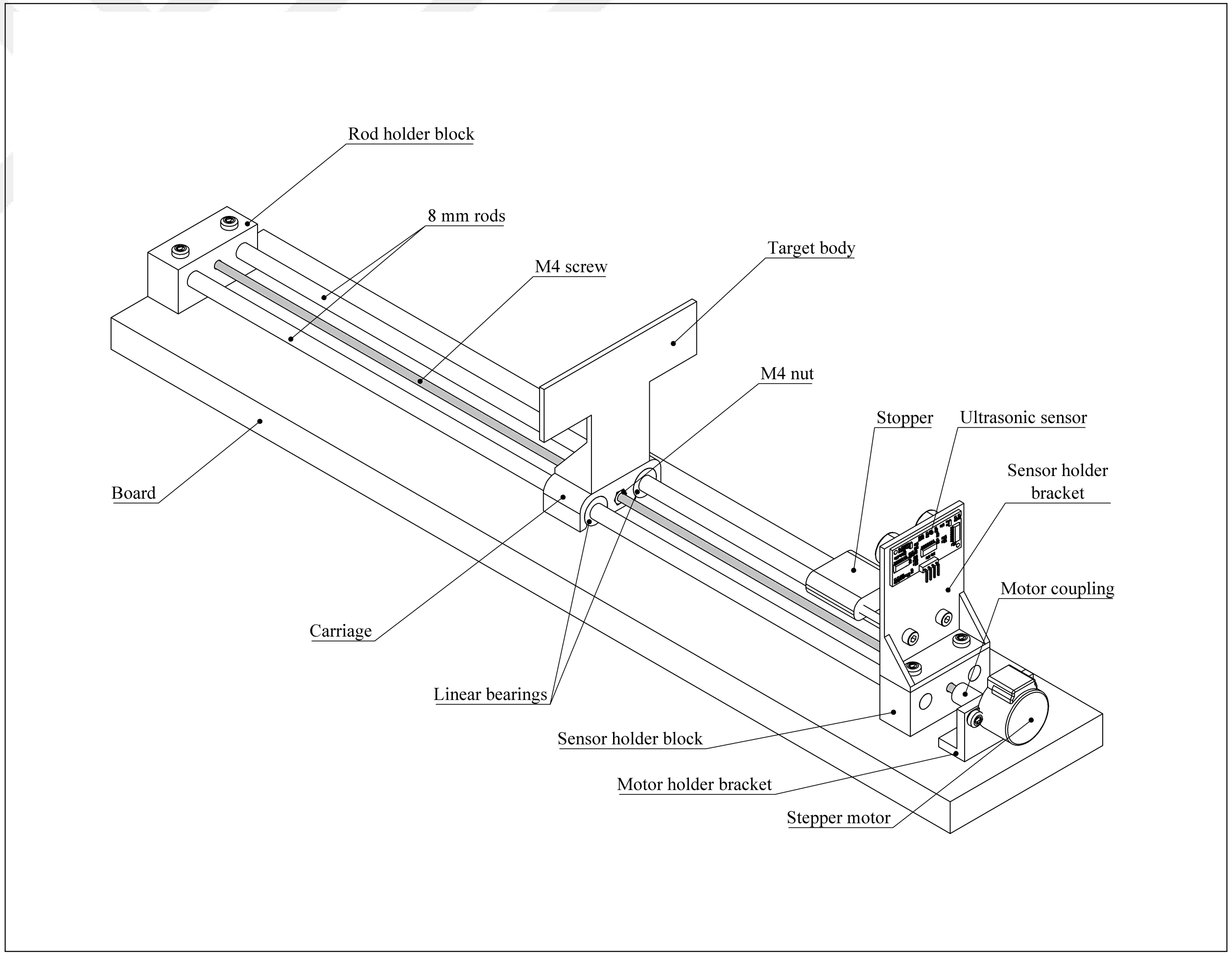

Figure 6.8. Detailed view of the experimental setup and its components

The linear displacement of the body for each step can be calculated based on the lead of the screw and the number of steps. This high resolution and precise linear displacement make the stepper motor a suitable choice for applications that require accurate control of motion. The linear displacement of the body for each step can be calculated as follows.

$$\Delta s = \frac{1rev}{512steps}\frac{0.7mm}{1rev} = 0.001367mm \cong 1\mu m/step \tag{6.10}$$

The stepper motor used in the experiment is capable of reaching a maximum speed of (1rev)/(3.62sec). The speed of the motor can be adjusted by changing the time delay between steps in the Arduino code, allowing for a range of speeds below the maximum value. In the described setup, the speed of the motor is set to (1rev)/(4sec)=$\pi/2$ (rad/sec), which results in a linear displacement of 100 mm over a period of 571 seconds (9.52 minutes).

However, taking a measurement immediately after the motor rotates one step would require a large number of measurements (73142 steps in this case), which is impractical. Therefore, a data elimination procedure is applied in which most of the data is discarded. As a result, measurements are taken approximately every 1000 steps, resulting in approximately 73 measurements in total for a 100 mm long path. This approach allows for a reasonable number of measurements to be taken without overwhelming the system with unnecessary data.

When the motor is driven at a constant speed of 1 rev/4 sec, the linear displacement of the body is v_c =(0.7mm)/(1rev)(1rev)/(4sec)=0.175 mm/s . Using this constant velocity value, the displacement of the body over time can be described as $s(t) = v_c t = 0.175t$, where s and t denote the displacement in mm and time in seconds, respectively. The initial conditions for this equation are assumed to be $s(0) = 0$ and $t_0 = 0$. This equation describes the linear displacement of the body as a function of time, with the body starting at a displacement of zero at time $t = 0$.

The equation $s(t) = 0.175t$ describes a simple linear relationship between s and time t, with a slope of 0.175 and an intercept of zero. However, the operating range of the ultrasonic distance sensor used in the experiment is limited to distances between 2 cm and 4 m. As a result, the minimum measurable distance must be updated to 2 cm. In order to ensure that the displacement of the body is within the range of the sensor, the initial displacement $s(0)$ is set to 50 mm. This results in the updated displacement function as

$$s(t) = 0.175t + 50 \tag{6.11}$$

where the true model parameters are 50 for the intercept and $0.175$for the slope, see Figure 6.9.

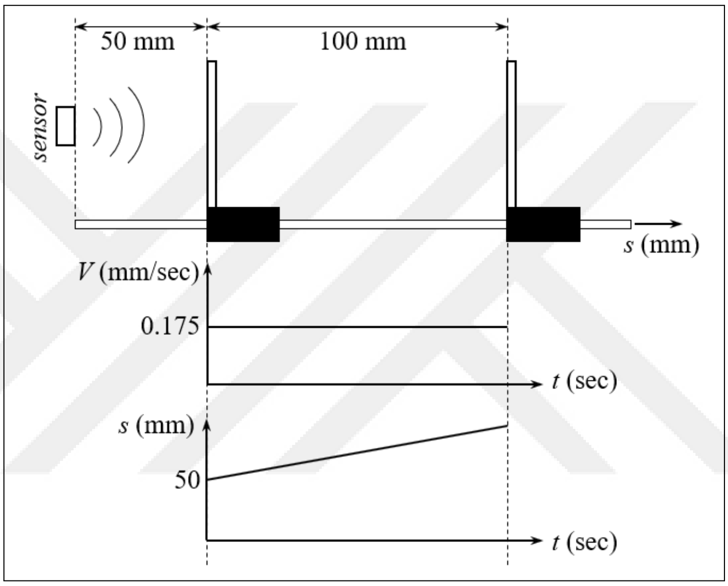

Figure 6.9. Ideal velocity and displacement profiles of the first experiment

The collected distance and time measurements for the experimental setup are plotted in Figure 6.10, along with the true model that describes the relationship between these variables. This plot allows for visual comparison between the measured data and the predicted values based on the true model.

After performing the necessary calculations, the solution polytope for the proposed method is obtained. The centroid of this polygon is used as the best estimate of the model parameters. In addition, OLS is applied to the dataset and the resulting estimates are presented in Table 6.2, with the estimates obtained using the proposed method.

MRSS values can be used to assess the accuracy of the estimates. The proposed method demonstrates its superiority when the MRSS values are compared. The superiority of the proposed method is also evident when the solution regions for each method are examined.

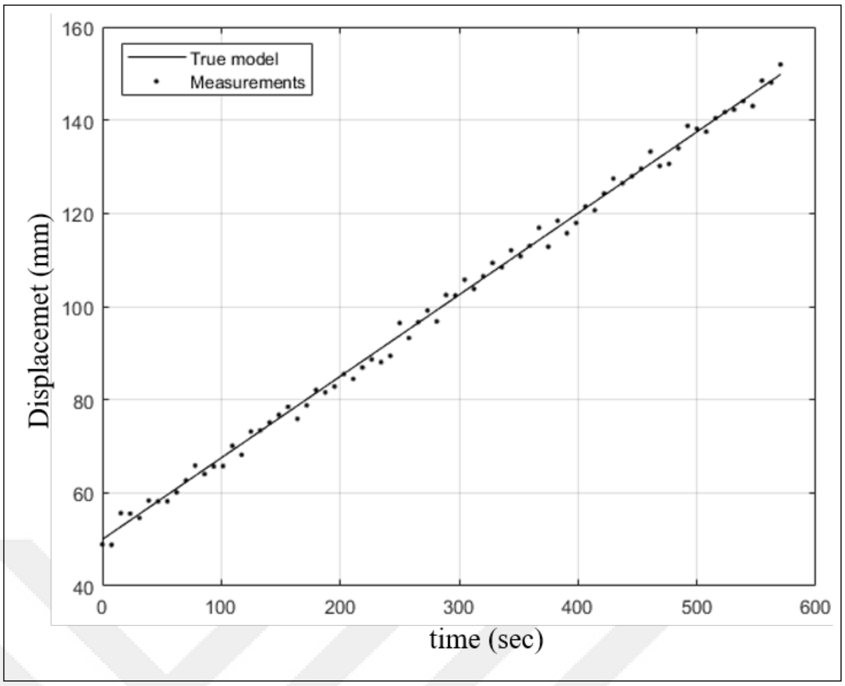

Figure 6.10. Plot of the displacement versus time for the first experiment

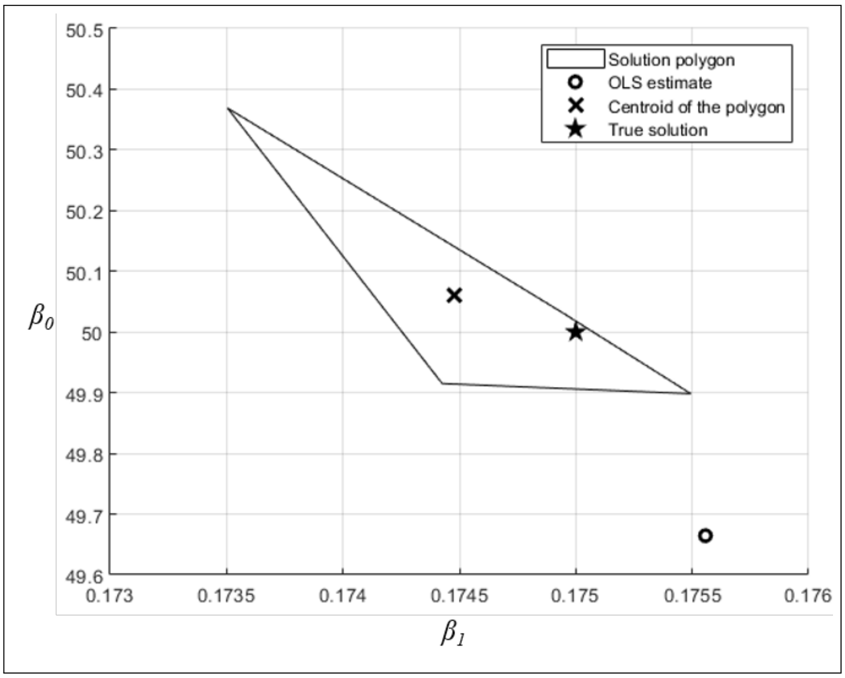

Figure 6.11. Solution region of the first experiment data

Table 6.2. Comparison of the MRSS values for the first experiment

	Intercept	**Slope**	**MRSS**
True solution	50	0.175	0
OLS	49.6645	0.1756	2.9489
Proposed method	50.0614	0.1745	1.1347

The solution regions and estimates for both methods are illustrated in Figure 6.11. The solution polygon for the proposed method has three vertices that define the boundaries of the solution region. It is important to note that the proposed method guarantees that the true solution lies within these boundaries, as demonstrated in Figure 6.11, whereas, the OLS estimate is clearly outside of the polygon.

In cases where the OLS estimate falls outside of the solution region defined by the proposed method, it can be concluded with significant confidence that the OLS estimate is unlikely to be the true solution. This is a distinctive feature of the proposed method, as it allows for the elimination of estimates that fall outside of the solution region. While estimates within the solution region have an equal probability of being the true solution, estimates that fall outside of the region can be rejected with confidence.

6.1.4.2. The Multiple Linear Case

In the second experiment, the multiple linear case is studied using a free fall experiment as a model system. The free fall experiment is a well-known example in which the true model parameters are known theoretically, see Figure 6.12. This allows for a direct comparison between the estimated model parameters and the true values.

$$h(t) = h_{\max} - \frac{1}{2}gt^2 \tag{6.12}$$

where $h_{\max}$ is the initial elevation of the object, g is the gravitational acceleration. This equation represents a second-degree polynomial, which can be studied by multiple linear regression model.

For this experiment, $h_{\min}$, $h_{\max}$ are set to 50 mm and 400 mm, respectively. This results in

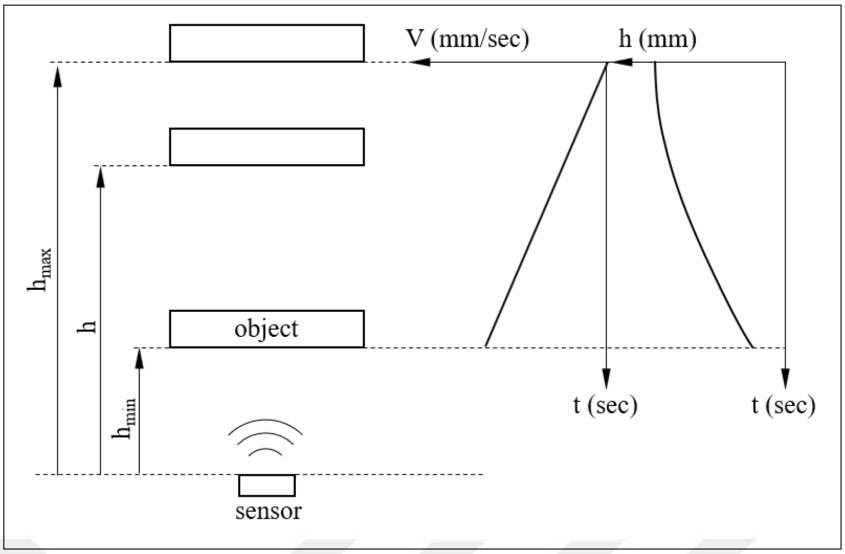

Figure 6.12. Ideal velocity and displacement profiles of the second experiment

$$h(t) = 400 - \frac{1}{2}9806.65t^2 \tag{6.13}$$

The gravitational acceleration g varies slightly depending on the altitude and latitude of the location where the experiment is performed. However, for most engineering calculations it is assumed to be 9806.65 mm/sec^2 at a standard location of sea level and a latitude of 45. Therefore, the true model parameters for this experiment are $\beta = [400, 0, -4903.325]$.

In order to conduct the free fall experiment, the setup from the first experiment can be modified to accommodate the vertical orientation and the falling mechanism. This involves adding components to hold the setup in a vertical position and designing a mechanism that triggers the fall of the object. The mechanism used in this experiment consists of a slider activated by a miniature servo motor. The M4 screw and nut mechanism that converts the rotation of the stepper motor into linear motion, along with the stepper motor itself, are removed from the experimental setup. This modification is necessary to accommodate the vertical orientation and falling mechanism of the free fall experiment. The experimental setup after the required modifications is demonstrated in Figure 6.13.

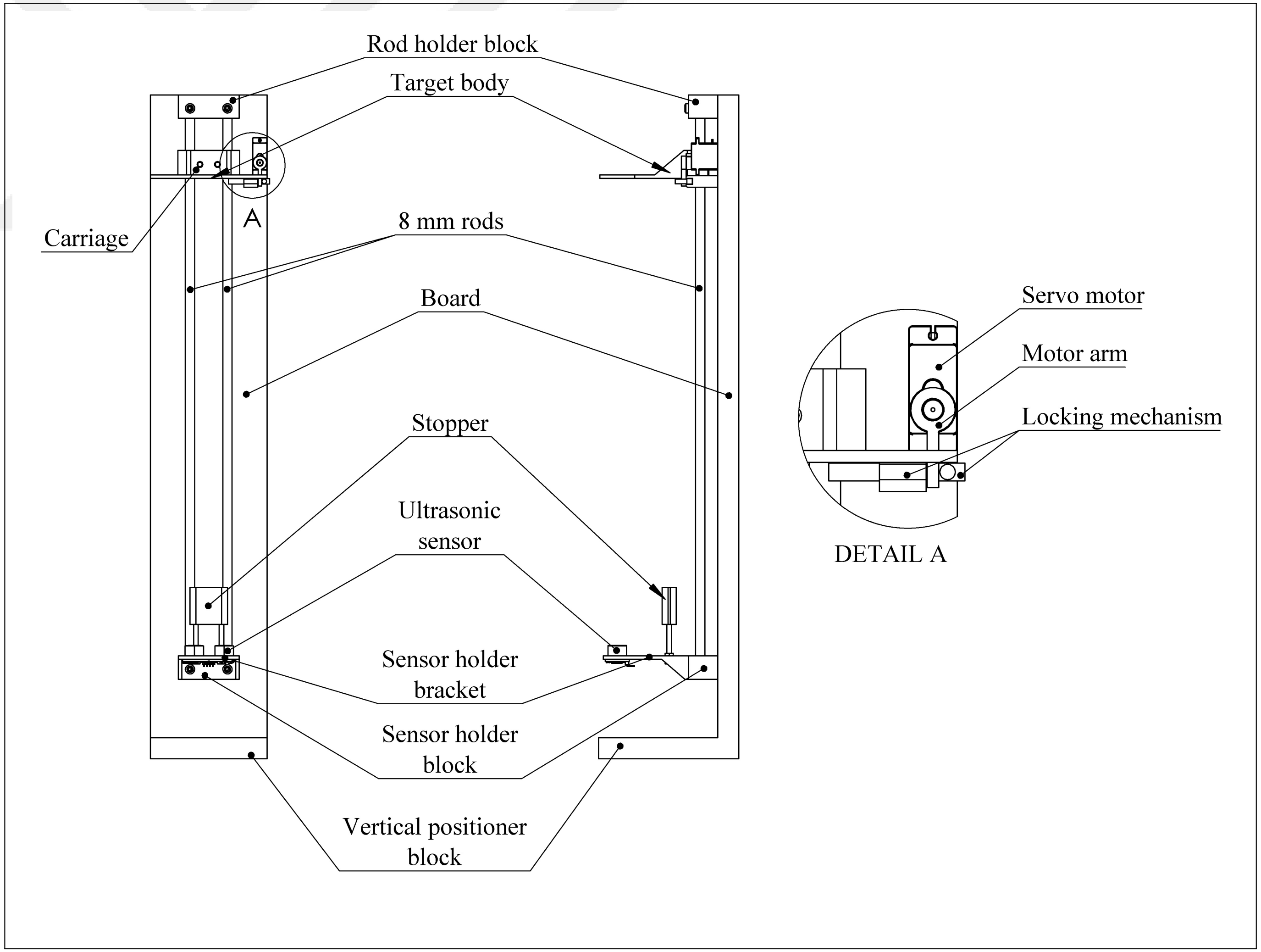

Figure 6.13. Detailed view of the second experimental setup and its components

Figure 6.14 shows the experimental data for the free fall experiment, along with the true model that describes the relationship between the variables. This plot allows for visual comparison between the measured data and the predicted values based on the true model.

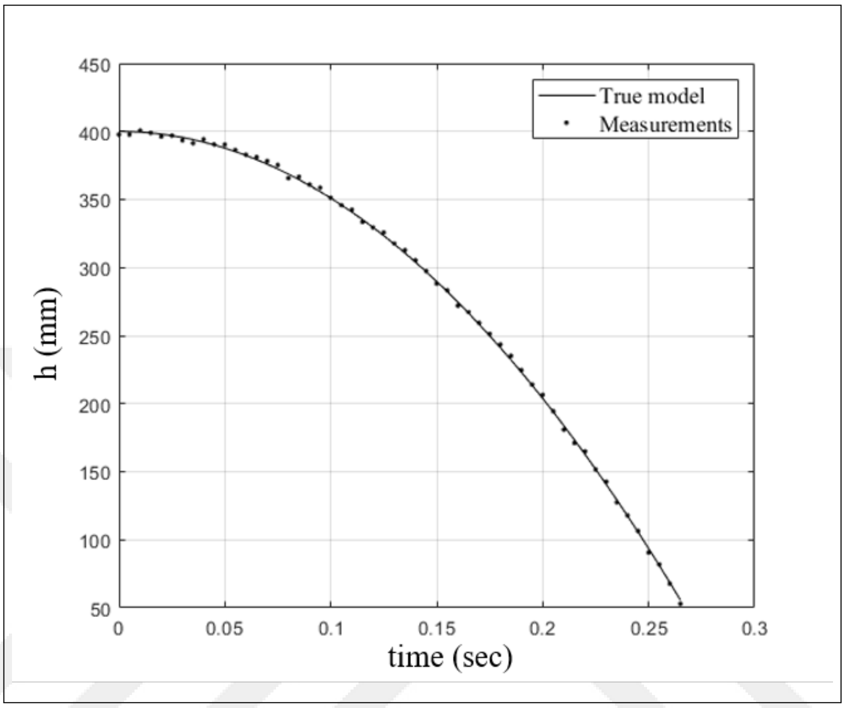

Figure 6.14. Plot of displacement versus time in the second experiment

After performing the necessary calculations, the solution polytope for the proposed method is obtained. The mean of the vertices of this polygon is used as the best estimate for the model parameters. The estimates obtained using OLS are also presented in Table 6.3, with the estimates obtained using the proposed method. The table also includes MRSS values for each method.

The differences in MRSS values between the proposed method and OLS are even greater in this experiment compared to the first experiment. The proposed method consistently yields

Table 6.3. Regression estimates of both methods for the second experiment

	β_0	β_1	β_2	**MRSS**
True solution	400	0	-4903.325	0
OLS	398.8403	34.5735	-5035.0174	32.8128
Proposed method	399.6699	5.5294	-4923.5074	0.9572

more accurate results. In addition, the estimates of the gravitational acceleration, g, obtained using OLS and the proposed method can be compared using the absolute percentage error (APE) as follows

$$APE = \frac{|g - 2\beta_2|}{g} \times 100\% \tag{6.14}$$

The APE values for the OLS and the proposed method estimates are 2.68% and 0.41%, respectively. These values demonstrate that the proposed method provides a significantly more accurate estimate of g compared to OLS.

The MRSS values and APE values show that the proposed method provides superior results compared to OLS. This difference is further highlighted when the solution regions for each method are examined. The solution region, estimates for OLS and the proposed method, and the true solution are illustrated in Figure 6.15. Note that the estimates obtained using OLS is once again outside of the solution region that can be fully described by the proposed method.

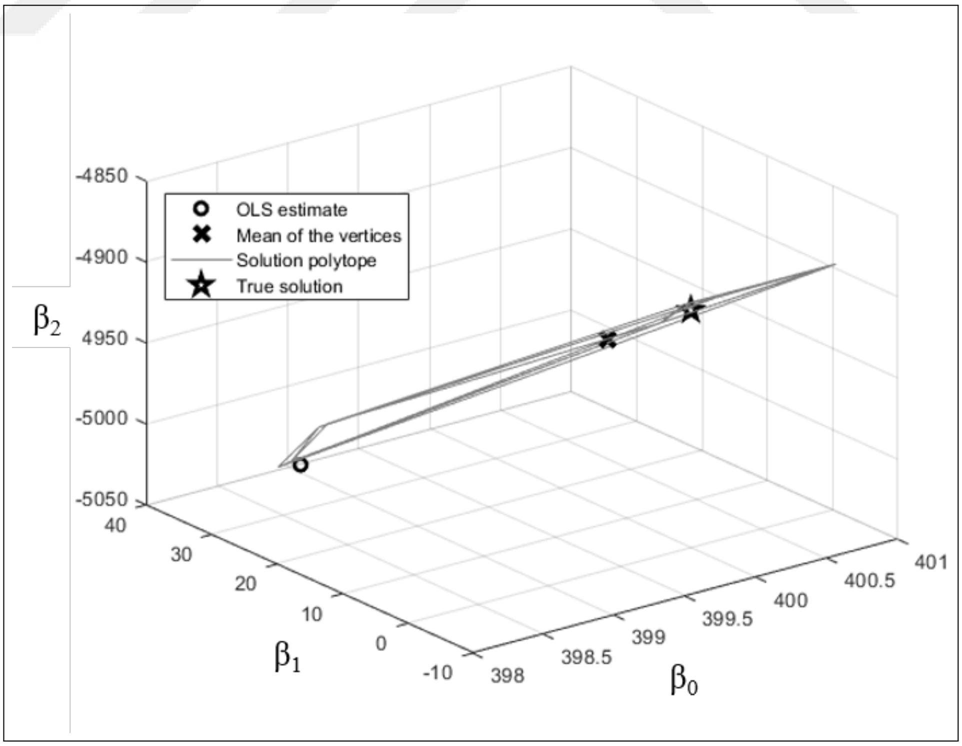

Figure 6.15. Solution region of the second experiment

6.2. TOLERANCE ANALYSIS BY POLYTOPES

Geometric product specification is a standardized method of communication that allows designers, manufacturers, and metrologists to exchange information about a product's assembly, functionality, and aesthetics in the face of geometric uncertainties. This communication is formalized through several standards, including ISO-1101 (2012) and ASME-Y14.5 (2009). These standards provide a way to define acceptable manufacturing deviations based on design requirements, and they form the basis for Geometric Dimensioning and Tolerancing (GD&T), a system used in defining engineering tolerances and relationships.

The selection of a tolerancing scheme is a significant decision that affects all stages of a product's life cycle and therefore has garnered significant attention in the field of tolerance management research. A central unresolved issue in this area is how to decompose the tolerances on individual parts in the assembly design process, given the constraints imposed by manufacturing capabilities.

The problem of decomposing assembly design requirements into tolerances on individual parts is typically approached in three steps. The first step involves modeling geometric deviations. The second step involves mathematically describing the design requirements. Finally, the mathematical constraints of the assembly are translated into constraints on the deviations of the individual parts.

Ideally, the transfer of assembly constraints to tolerances on individual parts should be carried out using a top-down approach, also known as tolerance synthesis. However, due to the intricate nature of this problem, it is often resolved using a bottom-up method. This involves identifying the parts within the tolerance chain that have the most impact, selecting tolerances for these parts, and then confirming that the requirements are met. The polytopes can be used to carry out this confirmation.

In this section, we study the use of polytopes in the tolerance analysis. We will use an example flange assembly including two parts; part H that has the holes and part P that has pins. The section views of both parts are demonstrated in Figure 6.16. Also, an exploded view of the assembly CAD model is shown in Figure 6.17.

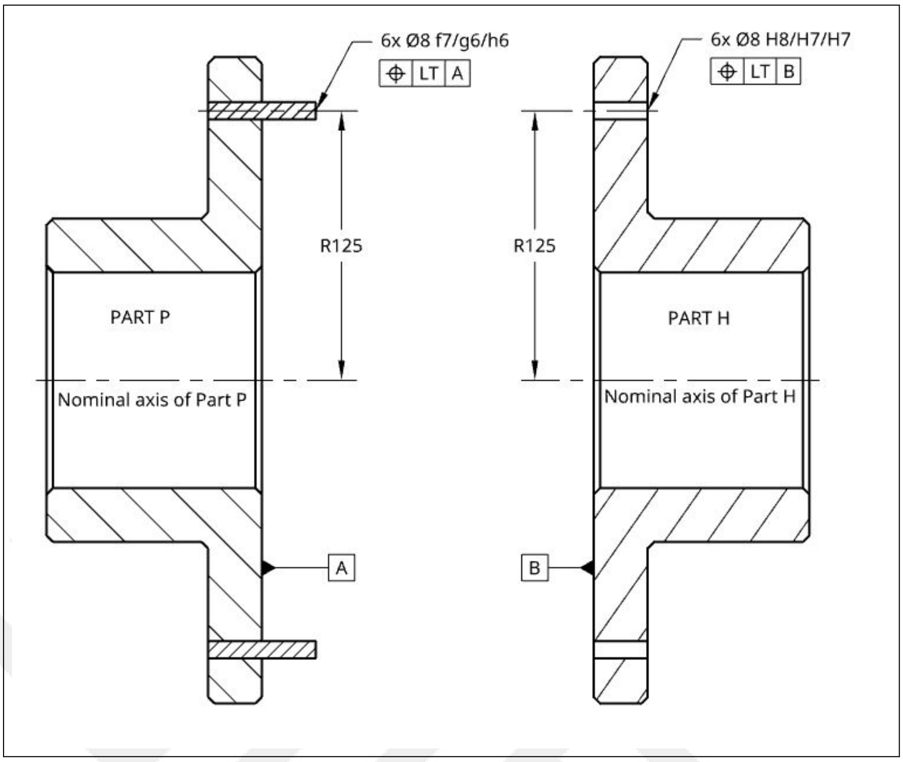

Figure 6.16. Section view of the assembly parts

The joint between the parts is comprised of a planar pair joint combined with N_h pin-hole pair joints secured by pins on part P and corresponding holes on part H arranged in a circular pattern, the radius of which is $\rho = 125$ mm. This design represents a highly overconstrained mechanism. Typically, the parts are fastened by bolts, but these components are not included in this study. The determination of geometric specifications can be carried out through various methods, depending on the desired level of compliance in the assembly. If maximum compliance in all assemblies is desired, it is recommended to perform calculations based on worst case assumptions.

The proposed work is based on the following assumptions:

• Form defects are not taken into account.

• Parts are treated as rigid bodies.

• Small rotations are treated as linear.

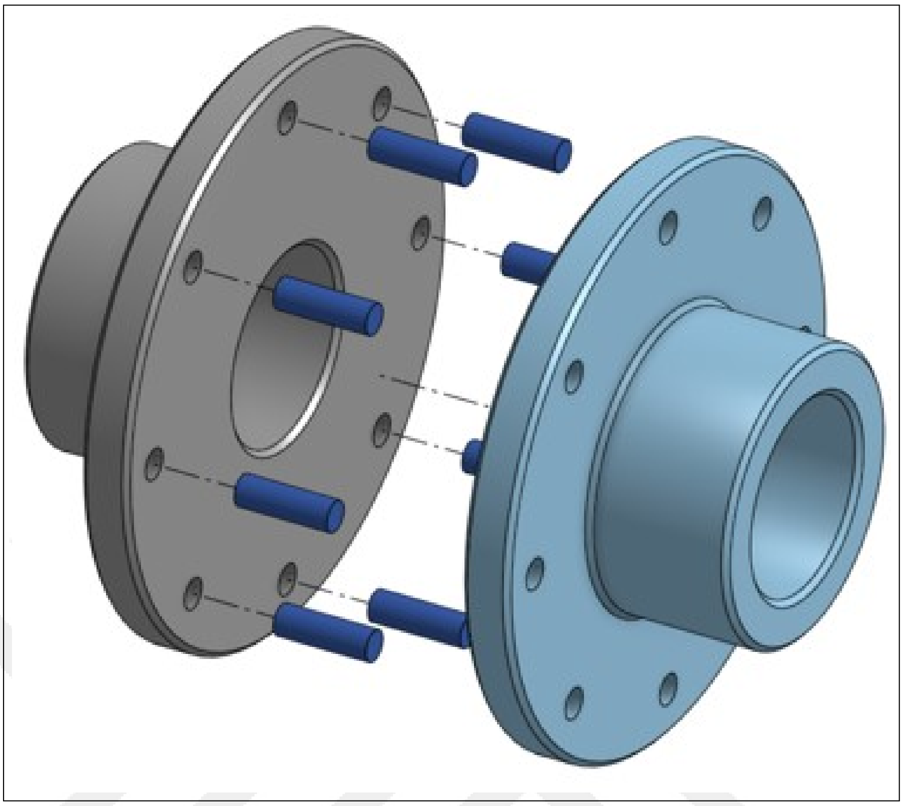

Figure 6.17. Exploded view of the assembly CAD model

6.2.1. The Geometric Model

The cylindrical surfaces of the holes and the pins are spread uniformly at equal angles and their axes are parallel to the nominal axis of the flange in the notional configuration specified in CAD model. Let the pin-hole pairs be indexed by i and θ_i be the angle of the nominal pin-hole position vector, as measured from x-axis in CCW direction, for $i = 1, \cdots, N_h$. Therefore, $\theta_i = (i-1)\frac{2\pi}{N_h}$, see Figure 6.18.

The nominal position vector of the i^{th} pin-hole pair is

$$\overrightarrow{r_{N,i}} = \rho \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \\ 0 \end{bmatrix} \tag{6.15}$$

$r_{N,i}$ represents the ideal (nominal) center positions of the i^{th} pin and hole pair for both parts. The likelihood of pin and hole pairs being produced in perfect positions is low in reality.

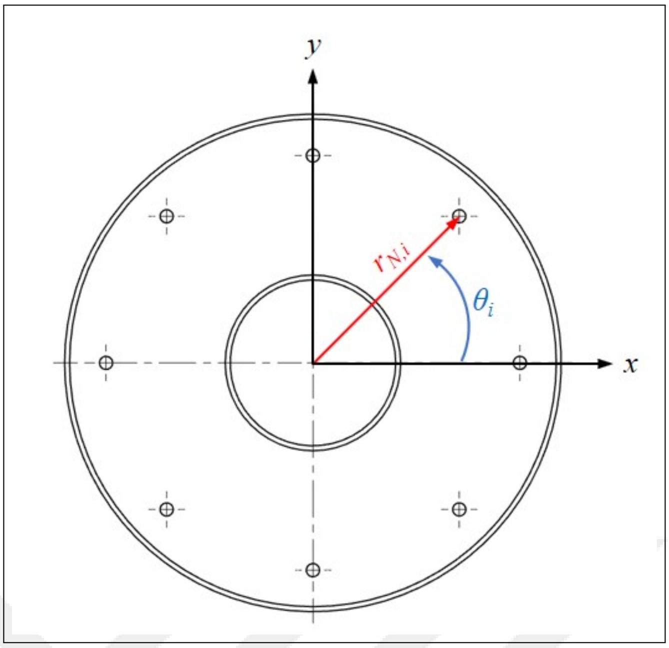

Figure 6.18. The pin-hole position vector

Deviation vectors, represented as $\overrightarrow{e_{p,i}}$ and $\overrightarrow{e_{h,i}}$, depict the deviation with respect to the nominal axes for the pin and hole respectively. The locational tolerance of the components, as depicted in Figure 6.16, limits the deviation to a maximum value LT. Therefore, the pin and hole center positions lie within a circle of radius LT and center $\overrightarrow{r_{N,i}}$, as shown in Figure 6.19. This forces $\left|\overrightarrow{e_{p,i}}\right|, \left|\overrightarrow{e_{h,i}}\right| \leq LT$. The center position vectors of the i^{th} pin and hole respectively are

$$\overrightarrow{r_{p,i}} = \overrightarrow{r_{N,i}} + \overrightarrow{e_{p,i}} \tag{6.16}$$

$$\overrightarrow{r_{h,i}} = \overrightarrow{r_{N,i}} + \overrightarrow{e_{h,i}} \tag{6.17}$$

The pin-hole compliance must consider both location deviations and size deviations (diameter deviations) in its geometric model. Denote the diameter of the i^{th} pin as $D_{p,i}$ and that of the hole as $D_{h,i}$. The size tolerances of the pin and hole are defined by fit tolerances, e.g., Ø8 H8/f7. In the given example, the nominal diameter is 8 mm, and the hole tolerance is H8, with the diameter limits of $D_{h,i} \in [8.000, 8.022]$. The pin tolerance is f7, resulting in a diameter interval of $D_{p,i} \in [7.972, 7.987]$. It is important to note that this study does not consider interference fits, hence, care must be taken in determination of the size tolerance

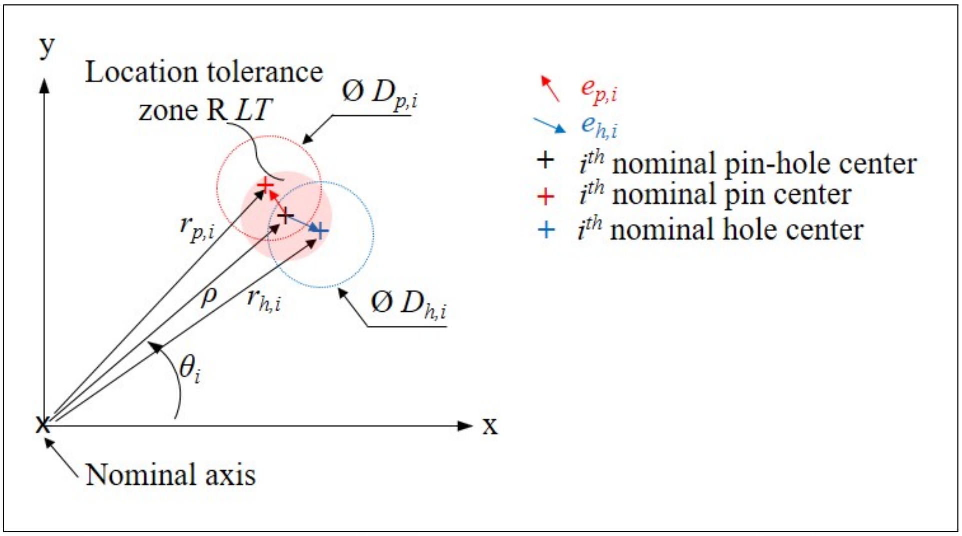

Figure 6.19. Illustration of the locational tolerance zone for a single pin-hole pair

since the lower limit of $D_{h,i}$ must always be equal to or greater than the upper limit of $D_{p,i}$.

Let $\overrightarrow{e_i}$ be the difference vector defined between the centers

$$\overrightarrow{e_i} = \overrightarrow{r_{h,i}} - \overrightarrow{r_{p,i}} = \overrightarrow{e_{h,i}} - \overrightarrow{e_{p,i}} \tag{6.18}$$

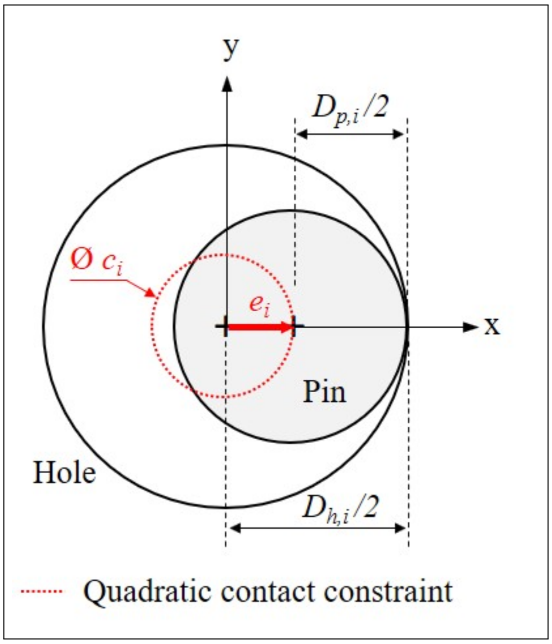

Figure 6.20. Demonstration of the quadratic constraint

To insert the pin into the hole, the magnitude of the radial clearance, $|\overrightarrow{e_i}|$, must be less than or equal to half of the diametrical clearance, $c_i/2$, where c_i is the difference between the diameter of the hole, $D_{h,i}$, and the diameter of the pin, $D_{p,i}$. The equation for this relationship is $\left|\overrightarrow{e_i}\right| \leq D_{h,i}/2 - D_{p,i}/2 = c_i/2$. This quadratic contact constraint ensures that the pin and the hole do not interfere with each other during their interaction. It requires the center of the pin to be within a specific circular region, as illustrated in Figure 6.20, with a diameter determined by c_i.

The nonlinear contact constraint between the pin and the hole is beyond the scope of this thesis. Therefore, the constraint is linearized through the discretization of the pin-hole pair. The number of discretization points is denoted by N_d. The unit vector from the nominal center $\overrightarrow{r_{N,i}}$ to the j^{th} discretization point is denoted by $\overrightarrow{n}_j$, where j ranges from 1 to N_d, see Figure 6.21.

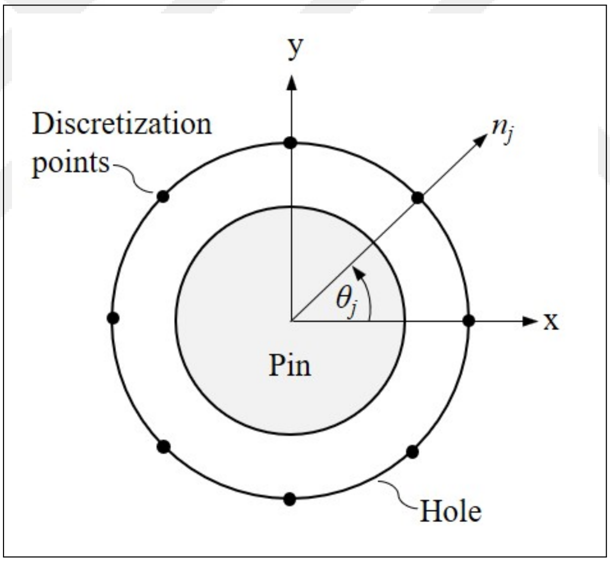

Figure 6.21. Discretization of the pin-hole pair

The vector can be expressed as

$$\overrightarrow{n}_j = \begin{bmatrix} \cos\theta_j \\ \sin\theta_j \\ 0 \end{bmatrix} \tag{6.19}$$

where $\theta_j = (j-1)\frac{2\pi}{N_d}$. The projection length of the radial clearance, $\overrightarrow{e_i}$, onto each unit vector, $\overrightarrow{n}_j$, must be less than or equal to half of the diametrical clearance, $c_i/2$. The linearized contact constraint is given by

$$-c_i/2 \leq \overrightarrow{e_i} \cdot \overrightarrow{n_j} \leq c_i/2 \tag{6.20}$$

This linear constraint provides $2N_d$ linear inequalities for a single pin-hole pair. The precision of the solution is proportional to the number of discretization points used. The circular region defined by the quadratic contact constraint is approximated by a polygonal region inscribed within the circular region. The precision of the approximation is represented by the distance p between the polygonal and circular regions, as illustrated in Figure 6.22.

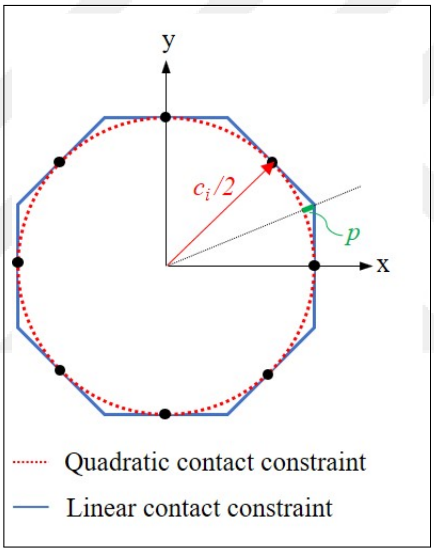

Figure 6.22. The quadratic and the linearized contact constraints

The quantity p is defined as the maximum distance between the boundary of the quadratic contact constraint and the boundary of the linear contact constraint. It can be calculated using the equation

$$\frac{c_i}{2} = \left(p + \frac{c_i}{2}\right) \begin{bmatrix} \cos\frac{\pi}{N_d} \\ \sin\frac{\pi}{N_d} \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \cos 0 \\ \sin 0 \\ 0 \end{bmatrix} \tag{6.21}$$

$$p = \frac{c_i}{2} \frac{1 - \cos\frac{\pi}{N_d}}{\cos\frac{\pi}{N_d}} \tag{6.22}$$

In order to ensure a safe assembly, the value of p is integrated into the linear contact constraint as

$$-c_i/2 - p \le \overrightarrow{e_i} \cdot \overrightarrow{n_j} \le c_i/2 + p \tag{6.23}$$

However, it is usually ignored as it is small and negligible.

If $2N_d$ linear inequalities are satisfied, it is certain that the pin is in the hole. The assembly is designed with the nominal axes of both parts coincident, following the ideal CAD model. However, to account for manufacturing variability, a relative freedom between the parts is introduced through tolerances. The part H is fixed while the part P is positioned with translational freedom on the $x - y$ plane and rotational freedom about its nominal axis, represented by $\overrightarrow{t}$ and R respectively.

$$\overrightarrow{t} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \quad R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.24}$$

where θ represents the degree of rotation about the nominal axis of part P. Now, $\overrightarrow{e_i}$ can be expressed as

$$\overrightarrow{e_i} = \overrightarrow{r_{N,i}} + \overrightarrow{e_{h,i}} - (R(\overrightarrow{r_{N,i}} + \overrightarrow{e_{p,i}}) + \overrightarrow{t}) \tag{6.25}$$

Here, x, y, and θ are free variables and the aim is to determine the range of these variables for which the assembly is feasible, considering a single pin-hole pair. Let the space of these variables be $\mathbb{D} \subset \mathbb{R}^3$, the assembly deviation space. Thus, the problem transforms into a membership problem, where the feasible $\begin{bmatrix} x & y & \theta \end{bmatrix}$ vectors define a solution region S_i in $\mathbb{D}$ for the i^{th} pair.

In order to have a set of linear inequalities the rotation matrix R should be linearized. Since the rotation will be very small, usual $sin(\theta)$ and $cos(\theta)$ approximations can be used.

$$R = \begin{bmatrix} 1 & \theta & 0 \\ -\theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.26}$$

In this way, the difference vector becomes

$$\overrightarrow{e_i} = \overrightarrow{e_{h,i}} - \overrightarrow{e_{p,i}} - \theta(\overrightarrow{r_{N,i}} + \overrightarrow{e_{p,i}}) \times \overrightarrow{k} - \overrightarrow{t} \tag{6.27}$$

where $\overrightarrow{k}$ denotes the unit vector along the z-axis. Integrating the above result into the linearized contact constraint yields

$$-c_i/2 \leq (\overrightarrow{e_{h,i}} - \overrightarrow{e_{p,i}} - \theta(\overrightarrow{r_{N,i}} + \overrightarrow{e_{p,i}}) \times \overrightarrow{k} - \overrightarrow{t}) \cdot \overrightarrow{n_j} \leq c_i/2 \tag{6.28}$$

Let

$$\overrightarrow{e_{h,i}} = \begin{bmatrix} x_{h,i} \\ y_{h,i} \\ 0 \end{bmatrix} \quad \overrightarrow{e_{p,i}} = \begin{bmatrix} x_{p,i} \\ y_{p,i} \\ 0 \end{bmatrix} \tag{6.29}$$

Substituting into the contact constraint gives

$$-\frac{c_i}{2} \leq \left(\begin{bmatrix} x_{h,i} \\ y_{h,i} \\ 0 \end{bmatrix} - \begin{bmatrix} x_{p,i} \\ y_{p,i} \\ 0 \end{bmatrix} - \theta \begin{bmatrix} \rho\cos\theta_i + x_{p,i} \\ \rho\sin\theta_i + y_{p,i} \\ 0 \end{bmatrix} \times \overrightarrow{k} - \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \right) \cdot \begin{bmatrix} \cos\theta_j \\ \sin\theta_j \\ 0 \end{bmatrix} \leq \frac{c_i}{2} \tag{6.30}$$

which can be expressed in matrix notation with respect to the independent variables x, y, and θ as follows.

$$\begin{bmatrix} -\cos\gamma_j & -\sin\gamma_j & -((\rho\sin\theta_i + y_{p,i})\cos\theta_j + (\rho\cos\theta_i + x_{p,i})\sin\theta_j) \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{6.31}$$

$$\leq \frac{c_i}{2} - ((x_h - x_p)\cos\theta_j + (y_h - y_p)\sin\theta_j)$$

and

$$\begin{bmatrix}\cos\theta_j & \sin\theta_j & -(x_p\sin\theta_j - y_p\cos\theta_j)\end{bmatrix}\begin{bmatrix}x\\ y\\ \theta\end{bmatrix} \tag{6.32}$$

$$\leq \frac{c_i}{2} + ((x_h - x_p)\cos\theta_j + (y_h - y_p)\sin\theta_j)$$

With the definition of new linear constraints, the compliance of a single pair is checked by $2N_d$ linear inequalities. ETA can be employed to find the solution region S_i, an unbounded polytope with N_d facets corresponding to the hyperplanes of the inequalities. Note that the remaining N_d inequalities of the system are redundant since the compliance is checked symmetrical for each discretization point.

Now, the assembly status is to be evaluated by considering all pin-hole pairs. The solution regions of each pair are intersected to determine the final polytope $S_f \subset \mathbb{D}$, see Figure 6.23.

$$S_f = \bigcap_{i=1}^{N_h} S_i \tag{6.33}$$

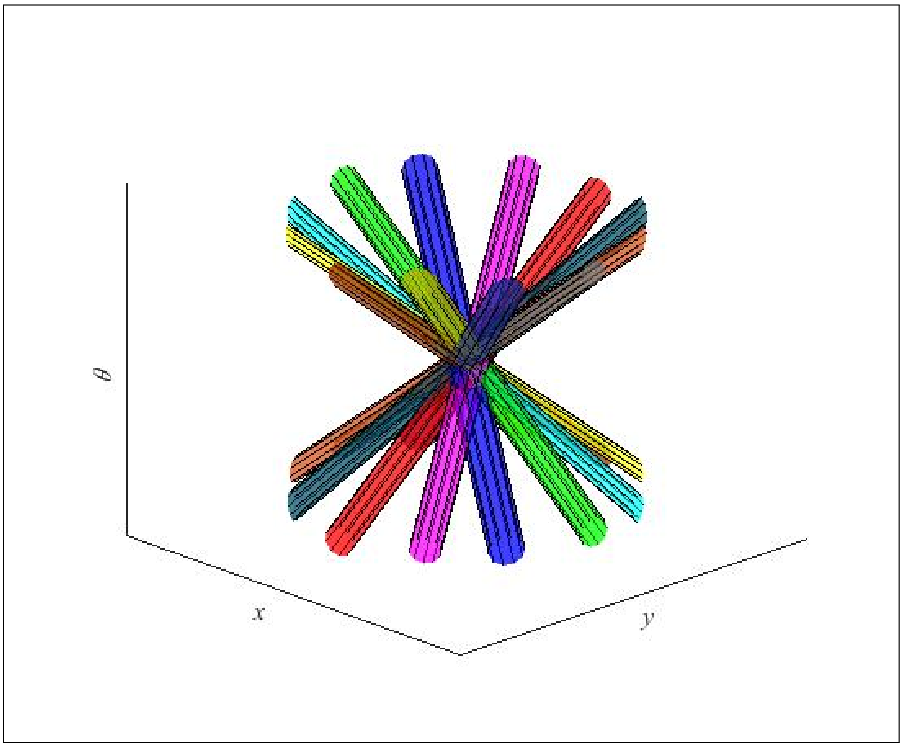

Figure 6.23. The polytopes of all pin-hole pairs

The resulting polytope of the intersection is demonstrated in Figure 6.24.

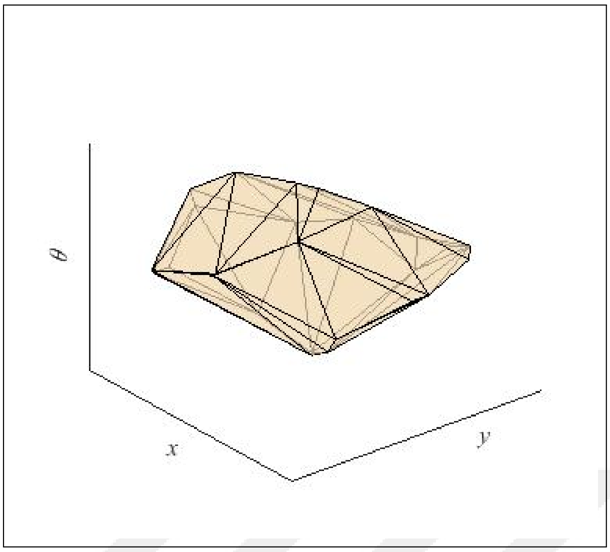

Figure 6.24. The resultant polytope of the intersection

6.2.2. Functional Condition

If $S_f \neq \emptyset$, then the assembly is possible. However, the feasibility of the assembly does not necessarily mean that it meets the functional condition (FC) of the assembly. For our model, FC can be defined as the distance between the nominal axes of both parts. The greater the distance, the higher the misalignment. Therefore, FC sets the limit for the maximum permissible misalignment.

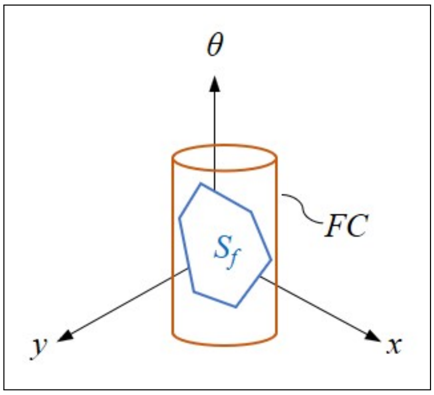

Figure 6.25. Illustration of the FC in the deviation space

Consider the image of FC in $\mathbb{D}$. It is a hollow cylinder where S_f has to be contained within, as shown in Figure 6.25. The radius of the cylinder defines the maximum possible distance between the nominal axes.

To check the inclusion, the analysis can be performed on the $x-y$ plane by considering a circle instead of the cylinder, since rotation does not affect the distance between the nominal axes. If the projected image of S_f on the $x-y$ plane remains inside the FC circle, then the FC is satisfied. To this end, the extreme points (vertices) of S_f are used as they represent the boundary of S_f. Their projections onto the $x-y$ plane are calculated by zeroing their θ coordinates.

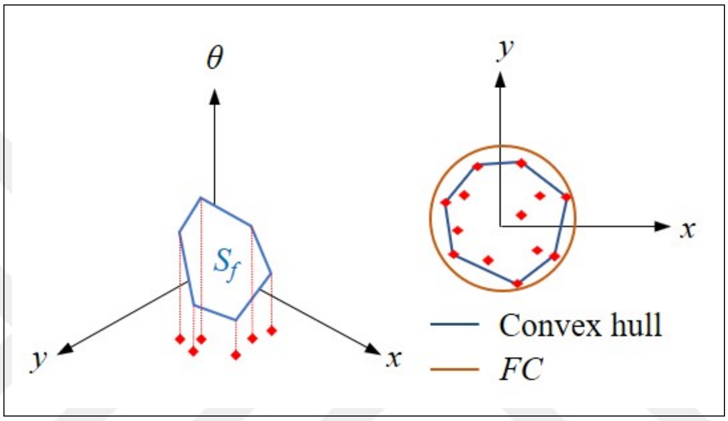

Figure 6.26. Projection of S_f onto $x-y$ plane

Any convex hull algorithm can be employed to eliminate the interior points, see Figure 6.26. The resulting points will represent the boundary of the S_f on the $x-y$ plane. In this way, S_f becomes a polygon.

Let the set $S_f^* \subset \mathbb{R}^2$ be the polygonal area. If the element (clearly a vertex) of S_f^* that has the greatest distance to the origin, say $e_{\max}$, is located in or on the FC circle, then the FC is satisfied. This vertex represents the worst case scenario during the assembly. Therefore, one can determine the maximum possible assembly misalignment by $e_{\max}$.

On the other hand, S_f^* also informs about the expected misalignment, say e_{ave}. Since the execution of the assembly is possible with any point(element) of S_f^*, we consider that all points have the same probability. Therefore, with equal chances, the expected point (misalignment) is always the *centroid* of the S_f^*. Thus, e_{ave} is the distance from the origin to the centroid. Furthermore, the element of S_f^* that has the minimum distance to the origin defines the minimum possible misalignment $e_{\min}$. Note that if the origin is an element of S_f^*, then $e_{\min} = 0$.

In this section, we demonstrated a method for determining the feasibility of an assembly and estimating the potential misalignment based on the measured dimensions of the manufactured parts. For given part dimensions, one can determine the following quantities:

- Feasibility of the assembly: S_f.
- The maximum possible misalignment: $e_{\max}$.
- The expected misalignment: e_{ave}.
- The minimum possible misalignment: $e_{\min}$.

Based on these calculations, a decision can be made about the suitability of the assembly with respect to the given functional requirement. The following scenarios are possible:

1. If $e_{\max} < FC$, then the assembly can be performed with confidence, as the maximum misalignment will always be less than the FC value.
2. If $e_{\min} \leq FC \leq e_{\max}$, then a decision is to be made, since the FC may be violated. Here, e_{ave} as the mean, a distribution can be modelled based on S_f^*, in the interval $[e_{\min}, e_{\max}]$. This distribution can give an idea about the probability of violating FC. An example is demonstrated in Figure 6.27.
3. In the remaining case, $FC \leq e_{\min}$, the assembly should not be performed as it will result in a violation of the FC for all scenarios.

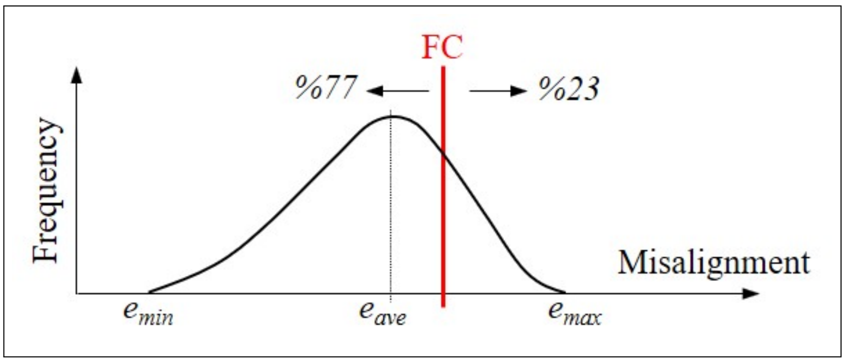

Figure 6.27. An example distribution of the misalignment

6.2.3. Monte Carlo Simulations to Determine the Tolerance Effects

In what follows, we will analyze the effect of the tolerances and the the number of pin-hole pairs N_h on the assembly, in terms of feasibility of the assembly and $e_{\max}$. In the simulations, virtual parts are created using MATLAB with respect to the nominal geometry of the parts and predetermined set of tolerances. In the nominal geometry, the dimensions are fixed and are given as follows for both parts. The theoretical radius of pin and hole centers ρ is assigned to be 125 mm. The nominal diameter of pins and holes is fixed to 8 mm.

In order to simulate a real-life manufacturing scenario, one considers random variables that will provide the necessary defects (deviations) on the parts. The first deviation is related to the position of the pin and hole centers and originates from $\overrightarrow{e_{p,i}}$ and $\overrightarrow{e_{h,i}}$. For this reason, these vectors are random vectors that are constrained by the locational tolerance such that $\left|\overrightarrow{e_{p,i}}\right| \leq LT$ and $\left|\overrightarrow{e_{h,i}}\right| \leq LT$.

In order to generate any of $\overrightarrow{e_{p,i}}$ and $\overrightarrow{e_{h,i}}$, first, the length of the vector is randomly generated using a truncated normal distribution $N(0, LT/3)$. The estimate of the standard deviation of the distribution is determined as $\sigma = LT/3$. In this way, almost all ($\%99.7$) of the generated vectors will be shorter than $3\sigma = LT$. The rest are eliminated by truncation operation. Then, the orientation of the vector is randomly determined using a uniform distribution in the interval of $[0, 2\pi]$.

Considering that the pin insertion holes will be CNC milled for both parts, the precision of the milling machine has a large share in simulating this tolerance. A general accuracy specification of these machines is 0.005 mm. Along with this information, LT is assigned to have 10 different values ranging from 0.001 mm to 0.010 mm with an increment of 0.001 mm in the simulations.

The second random parameter is the size tolerance of the pin-hole pairs. Here, we consider a set of three different hole based fitting tolerances with the nominal diameter 8 mm. These are close running H8/f7, sliding H7/g6, locational clearance H7/h6. The limits of these tolerances are presented in Table 6.4.

Note that all types are deliberately selected to be clearance fits with the minimum clearance

Table 6.4. The size tolerances used in the simulations

	H8/f7	**H7/g6**	**H7/h6**
Hole lower limit	8.000	8.000	8.000
Hole upper limit	8.022	8.015	8.015
Pin lower limit	7.972	7.986	7.991
Pin upper limit	7.987	7.995	8.000

0 for H7/h6, since the pin diameter has to be less than or equal to the hole diameter. The diameters are randomly generated with respect to the limits of the assigned tolerance type. For example, if H8/f7 is selected, then $D_{h,i}$ is randomly generated in the interval $[8.000, 8.022]$ with a normal truncated distribution $N(8.011, 0.00367)$. Here, the standard deviation term is determined by touching the interval limits at a distance of 3 standard deviations from the mean again. Similarly, $D_{p,i}$ is randomly generated in the interval $[7.972, 7.987]$ with a normal distribution $N(7.9795, 0.0025)$.

The last random variable in the model geometry is the number of pin-hole pairs N_h. The minimum required N_h is two to fix part P to the part H. Also, since symmetrical designs are appreciated in mechanical designs, N_h can be considered an even number. In this context, N_h is assigned to be $2, 4, 6, 8, 10, 12, 14, 16$ in the simulations. Therefore, with ten different values of the locational tolerance, three different size tolerances and eight different $N_h$values, the simulation includes $10 \times 3 \times 8 = 240$ different models. Note that each model is virtually manufactured 100 times to see overall distribution of the assembly control parameters.

The control parameters for each model are: PA and $e_{\max}$, where PA represents the percentage of possible assemblies in 100 trials. In a trial, if S_f exists, then the assembly is considered as possible. Further, $e_{\max}$ is the maximum of the $e_{\max}$ values in 100 trials.

The effect of the number of pin-hole pairs N_h on the control parameters (PA and $e_{\max}$) is shown in Figure 6.28. The data displayed represents a portion of a simulation with a fixed LT value of 7μm and a diametrical size tolerance of H7/h6 to isolate the effect of N_h. As N_h increases, PA decreases since each added pair adds additional constraints to the system, reducing the feasibility of assembly. At $N_h = 2$, almost all assemblies are possible, however, as N_h reaches its maximum value of 16, the probability of assembly drops to approximately 10% with a linear downward trend.

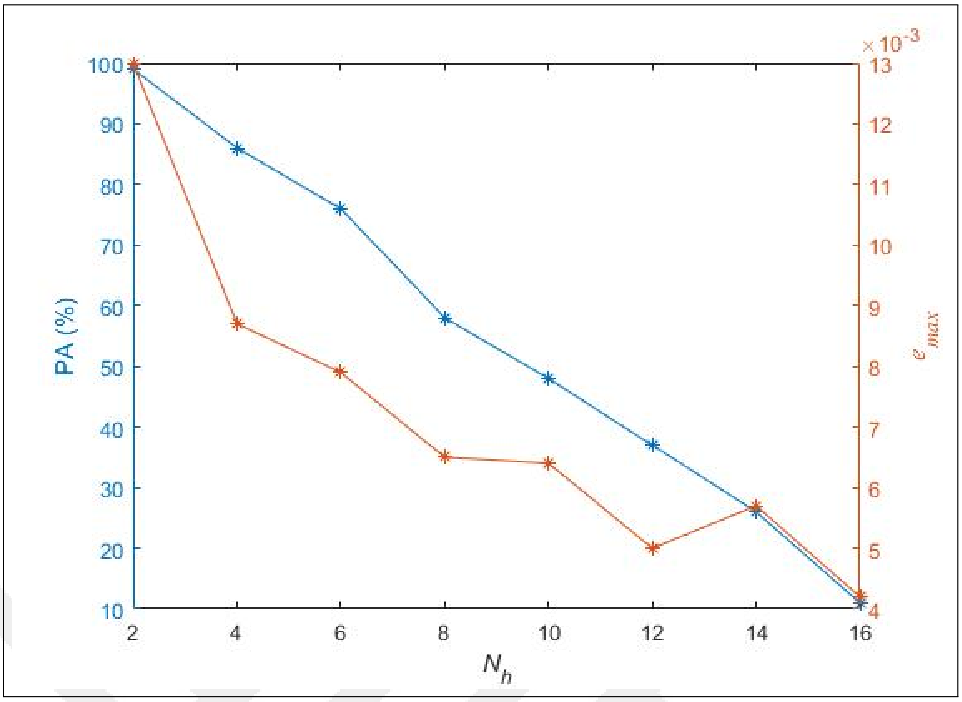

Figure 6.28. The effect of N_h on PA and e_{max}

The additional constraints result in improved positioning accuracy, as evidenced by the decrease in $e_{\max}$. The value of $e_{\max}$ decreases from 13μm at $N_h = 2$ to 4μm at maximum N_h, following a quadratic downward trend. This suggests that while adding pin-hole pairs enhances positioning accuracy and improves satisfaction of functional constraints, assembly becomes more challenging.

Figure 6.29 illustrates the impact of the LT on the control parameters PA and $e_{\max}$. The displayed data represents the models with a N_h value of 8 and a diametrical size tolerance of H7/h6. Up to a value of LT (4μm), PA remains constant at approximately 100%, and then decreases approximately linearly to 10% when the LT reaches 10μm. This decrease is expected as smaller tolerance zones increase the difficulty of achieving pin-hole compliance for all pairs simultaneously. Conversely, $e_{\max}$ increases in the same LT range with a quadratic upward trend, ranging from 0.5μm to 3.35μm. This increase occurs because small LTs bring the parts closer to the ideal model, reducing the probability of assembly, but ensuring minimum misalignment if assembly is successful.

The impact of the final parameter, diameter size tolerance, on the system performance is documented in Table 6.5. The data presented is limited to models with 8 pin-hole pairs,

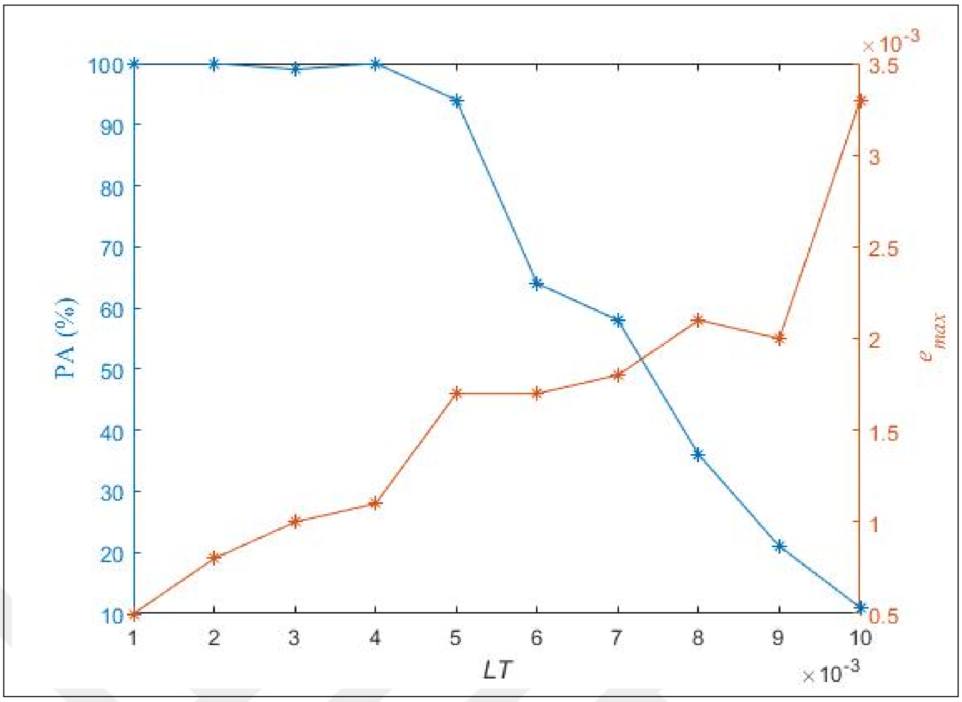

Figure 6.29. The effect of LT on PA and e_{max}

where LT is held constant at 7μm, to isolate the effect of size tolerance. The results indicate that as the tolerance becomes tighter, the positioning accuracy improves, as evidenced by the decrease in $e_{\max}$. However, it should be noted that a trade-off exists, since the feasibility of assemblies decreases as the clearance becomes smaller.

Table 6.5. The effect of the size tolerance on PA and e_{max}

N_h	LT	**Type**	PA	e_{max}
8	0.007	H8/f7	100	0.0173
8	0.007	H7/g6	99	0.0096
8	0.007	H7/h6	58	0.0065

The simulation results also showed that when assembly is not feasible, in most cases, $N_h - 1$ pin-hole pairs can be successfully assembled, with only one pair remaining incompatible due to interference. A potential mitigation strategy exists in this scenario. This option involves omitting the use of that pair, as the misalignment will increase, as indicated by Figure 6.28, with N_h decreasing. If the resulting value of $e_{\max}$ with $N_h - 1$ pairs remains compliant with the FC, this alternative may be pursued.

6.2.4. Compliant Pair Detection in a Batch

In batch production, if the assemblies are performed randomly, analyzed PA values provide a rough estimate of the number of successful assemblies. The remaining parts are considered scrap, increasing the overall manufacturing cost. To improve the PA value, compliant pairs can be identified within the batch.

To achieve this, each part of one type must be evaluated against all parts of the other type, resulting in $Z \times Z \times N_h$ polytope finding operations, where Z is the number of pairs in the batch. While this method may be computationally intensive, its use can result in significant cost savings in the manufacturing process.

In order to estimate the potential saving, a second simulation was conducted using tolerance values $N_h = 8$, $LT = 7\mu$m and a diametrical size tolerance of H7/h6. In this simulation, 100 Part H and Part P were virtually produced. If the parts were randomly assembled, then the resulting PA would be 58%, see Table 6.5.

Finally, an optimization process was implemented to maximize the number of compliant pairs. After the parts were re-matched, the PA value increased to 97%, reducing the number of scrap parts from 42% to 3%.

7. CONCLUSIONS

In Chapter 3, a simple, fast, and novel algorithm (Corridor Method, CM) is developed for enumeration of all vertices of the feasible region of parameters of a simple linear regression model with bounded error. CM is based on bijective maps of lines in the measurement space to the points in the parameter space. The feasible region is fully described by a list of ordered vertices free of duplicates. The worst-case time complexity of CM is determined as $O(N^2)$. Nevertheless, median-case complexity of CM seems to be $O(N)$.

In numerical tests, CM seemed to outperform two other well-known algorithms (DDM and lrs) by large margins. For low number of observations ($N \leq 100$) lrs was superior to DDM, whereas CM was 40 times faster than lrs. For $N > 100$, lrs became inordinately slow, whereas CM was about 200 times faster than DDM.

The CM method was specifically designed for the case of UBBE in linear programming problems. However, in many linear programming problems, constraints are defined independently of each other, rendering the special case of UBBE irrelevant. In Chapter 4, MCM was developed to address these types of problems. By dividing the corridor into two parts, the R_L and R_U solution sets can be found using a method based on CM. The final solution set is obtained by intersecting these two solution sets.

MCM efficiently finds the intersection result by analyzing the geometries of the images of R_L and R_U in the parameter space. The MCM method was applied to general two-parameter linear problems and demonstrated to have broad applicability across two-dimensional linear problems. MCM has a complexity comparable to CM and provides better time complexity compared to lrs and DDM.

In Chapter 5, a solution was sought for linear programming problems with a parameter space dimension greater than 2. A new method, the edge tracking algorithm (ETA), was developed and differs from the CM and MCM. This method detects the vertices of the solution polytope using its geometric elements and starts by identifying a vertex. The polytope is then traversed along its edges, revealing neighboring vertices.

ETA was compared to lrs and DDM, and performance tests showed that it has advantages for some problems, although it does not consistently outperform the other two methods.

The comparison between the performance of ETA and DDM showed that ETA outperforms DDM for small values of N, such as when $N = 10$ and $d = 6$. However, as N increases to approximately 200, the performance ratio between the two algorithms becomes equalized, with DDM outperforming ETA for values of N greater than 200. The influence of d on performance is significant, with increasing d generally leading to an improvement in the performance ratio for the majority of N values.

The comparison between the performance of lrs and ETA showed an inverse relationship, with lrs outperforming ETA for N values up to approximately 200. As N increases, ETA becomes the better performer, with a ratio of 2 observed for the largest N value considered. The effect of d on the performance ratio was variable, with a decrease observed for d values of $2, 5$, and 6. The impact of d on the performance ratio may not be straightforward to interpret, with a larger ratio observed for other values of d. Despite this, it can be concluded that ETA outperforms lrs for high N values.

ETA is proven to be a contender and an efficient tool that can be employed for solving all linear problem types. In order to demonstrate the practical applications of ETA two distinct applications are presented, both in engineering context.

The first application aims to employ the solution set as a regression estimator. To this end, synthetic experiments were performed to identify the most optimal regression parameters within the solution set. The results showed that the real model parameters are more likely to be situated near the center of the solution set rather than at its boundaries. The best estimate among the candidates was found to be the centroid of the solution set. It is noteworthy that the mean of the vertices may also be considered as an alternative, due to its computational simplicity.

After determining the best estimate, the performance of the proposed regression method was evaluated by comparing it to other regression techniques. A comparison was made with ordinary least squares (OLS), which was selected as the main competitor due to previously stated reasons.

In the synthetic experiments, the comparison was conducted across various scenarios, where the number of observations (N), the dimensionality of the parameter space (d), and the error characteristics were considered as experimental design criteria.

Subsequently, the performance was also assessed using two real-world experiments. The first experiment involved tracking the position of a target object using a low-accuracy ultrasonic distance sensor, while the true position was determined by a high-precision motor. The second experiment involved recording the position of a freely falling object using the same ultrasonic distance sensor, where the true model parameters were known theoretically.

In all experiments made with both synthetic and real data, the proposed regression technique showed a superior performance compared to OLS. In addition, in many experiments, the OLS gave results outside the solution region, which cannot be the true result.

In the second application, the feasibility of a flange assembly was studied depending on the production tolerances. This assembly model includes two different parts with pins and holes.

The geometry of the flange assembly was analyzed and the constraints necessary for successful assembly were modeled mathematically over a single pin-hole pair. Since the model was nonlinear, it was linearized to be compatible with the methods developed in this thesis. This resulted in a set of linear constraints which were used to evaluate the feasibility of the assembly based on a single pin-hole pair.

When this set constraints are given, ETA is able to check the pin-hole compliance and therefore, the feasibility of the assembly according to the state of the solution set. If the solution set is not empty, it can be said that the pair is compliant and the assembly is possible.

In the next step, the mathematical model was extended to include multiple pin-hole pairs simultaneously, with each pair adding a set of the same number of constraints to the system. The resulting, larger set of constraints determines the feasibility of the assembly. ETA can again be employed to obtain the solution region (polytope) of the full set of the constraints, which represents the intersection of the solution regions of each individual pair.

Consequently, a Monte Carlo simulation was conducted to investigate the impact of production tolerances on the assembly functionality. The simulations were based on virtual parts produced with different tolerance groups. In addition to the probability of successful assemblies, represented by the variable PA, the potential maximum assembly misalignment ε_{max} was calculated in the simulation. The results of the simulations showed that:

- An increase in the number of pin-hole pairs leads to a decrease in the assembly probability and an increase in the assembly sensitivity.
- An increase in the locational tolerance leads to an increase in the assembly probability, but a decrease in the assembly precision.
- Tighter pin-hole diameter tolerances result in reduced assembly probability and improved assembly accuracy.

REFERENCES

1. Definition of polyhedron from Wikipedia, the free encyclopedia [cited 2023 15 February]. Available from: https://en.wikipedia.org/wiki/Polyhedron.

2. Milanese M, Norton J, Piet-Lahanier H, Walter É. *Bounding approaches to system identification*. Springer Science & Business Media; 1996. Available from: https://doi.org/10.1007/978-1-4757-9545-5.

3. Dantzig GB, Orden A, Wolfe P, et al. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*. 1955;5(2):183-95.

4. Greenberg H. An algorithm for determining redundant inequalities and all solutions to convex polyhedra. *Numerische Mathematik*. 1975 Feb;24(1):19-26. Available from: https://doi.org/10.1007/BF01437214.

5. Kuznetsov VG. Algorithms for finding the general solution of a system of linear inequalities. *USSR Computational Mathematics and Mathematical Physics*. 1966;6(2):197-205. Available from: https://www.sciencedirect.com/science/article/pii/004155536690067X.

6. Chernikova NV. Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities. *USSR Computational Mathematics and Mathematical Physics*. 1965;5(2):228-33. Available from: https://www.sciencedirect.com/science/article/pii/0041555365900455.

7. Dyer ME, Proll LG. An improved vertex enumeration algorithm. *European Journal of Operational Research*. 1982;9(4):359-68. Available from: https://www.sciencedirect.com/science/article/pii/0377221782901801.

8. Fukuda K, Prodon A. Double description method revisited. *Combinatorics and Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg. 1996:91-111.

9. Avis D. Computational experience with the reverse search vertex enumeration algorithm. *Optimization Methods and Software*. 1998;10(2):107-24. Available from: https://doi.org/

10.1080/10556789808805706.

10. Avis D, Jordan C. mplrs: A scalable parallel vertex/facet enumeration code. *Mathematical Programming Computation*. 2018 Jun;10(2):267-302. Available from: https://doi.org/10.1007/s12532-017-0129-y.

11. Raffaele A. From optimization to listing: theoretical advances in some enumeration problems. Ph.D. Thesis, Universities of Verona & Trento, 2022.

12. Matheiss TH, Rubin DS. A Survey and Comparison of Methods for Finding All Vertices of Convex Polyhedral Sets. *Mathematics of Operations Research*. 1980;5(2):167-85. Available from: http://www.jstor.org/stable/3689148.

13. Dyer ME. The Complexity of Vertex Enumeration Methods. *Mathematics of Operations Research*. 1983;8(3):381-402. Available from: http://www.jstor.org/stable/3689308.

14. Avis D, Bremner D, Seidel R. How good are convex hull algorithms? *Computational Geometry*. 1997;7(5):265-301. Available from: https://www.sciencedirect.com/science/article/pii/S0925772196000235.

15. Maňas M, Nedoma J. Finding all vertices of a convex polyhedron. *Numerische Mathematik*. 1968 Oct;12(3):226-9. Available from: https://doi.org/10.1007/BF02162916.

16. Mattheiss TH. An Algorithm for Determining Irrelevant Constraints and all Vertices in Systems of Linear Inequalities. *Operations Research*. 1973;21(1):247-60. Available from: http://www.jstor.org/stable/169104.

17. Klee V. Polytope pairs and their relationship to linear programming. *Acta Mathematica*. 1974 Dec;133(1):1-25. Available from: https://doi.org/10.1007/BF02392139.

18. Dyer ME, Proll LG. An algorithm for determining all extreme points of a convex polytope. *Mathematical Programming*. 1977 Dec;12(1):81-96. Available from: https://doi.org/10.1007/BF01593771.

19. Avis D, Fukuda K. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Computational Geometry*. 1992

Sep;8(3):295-313. Available from: https://doi.org/10.1007/BF02293050.

20. Assad CL, Morales G, Arica J. VERTEX ENUMERATION OF POLYHEDRA. *Pesquisa Operacional*. 2022;42.

21. Chen PC, Hansen P, Jaumard B. On-line and off-line vertex enumeration by adjacency lists. *Operations Research Letters*. 1991;10(7):403-9. Available from: https://www.sciencedirect.com/science/article/pii/016763779190042N.

22. Le Verge H. A Note on Chernikova's algorithm. INRIA; 1992. RR-1662. Available from: https://hal.inria.fr/inria-00074895.

23. Genov B. The convex hull problem in practice: improving the running time of the double description method. Ph.D. Thesis, Universit‰ t Bremen, 2015.

24. Kaya IC, Ulus F. An iterative vertex enumeration method for objective space based vector optimization algorithms. *RAIRO Oper Res*. 2020;55:S2471-85.

25. Fogel E, Huang YF. On the value of information in system identification—Bounded noise case. *Automatica*. 1982;18(2):229-38. Available from: https://www.sciencedirect.com/science/article/pii/0005109882901108.

26. Belforte G, Bona B, Cerone V. Parameter estimation algorithms for a set-membership description of uncertainty. *Automatica*. 1990;26(5):887-98. Available from: https://www.sciencedirect.com/science/article/pii/0005109890900053.

27. Belforte G, Tay TT. Two new estimation algorithms for linear models with unknown but bounded measurement noise. *IEEE Transactions on Automatic Control*. 1993;38(8):1273-9.

28. Deller Jr JR, Nayeri M, Liu MS. Unifying the landmark developments in optimal bounding ellipsoid identification. *International Journal of Adaptive Control and Signal Processing*. 1994;8(1):43-60. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.4480080105.

29. Vicino A, Zappa G. Sequential approximation of feasible parameter sets for identification with set membership uncertainty. *IEEE Transactions on Automatic Control*.

1996;41(6):774-85.

30. Chisci L, Garulli A, Vicino A, Zappa G. Block recursive parallelotopic bounding in set membership identification. *Automatica*. 1998;34(1):15-22. Available from: https://www.sciencedirect.com/science/article/pii/S000510989700160X.

31. Kostousova EK. State estimation for dynamic systems via parallelotopes optimization and parallel computations. *Optimization Methods and Software*. 1998;9(4):269-306. Available from: https://doi.org/10.1080/10556789808805696.

32. Bravo JM, Alamo T, Camacho EF. Bounded error identification of systems with time-varying parameters. *IEEE Transactions on Automatic Control*. 2006;51(7):1144-50.

33. Cerone V, Piga D, Regruto D. Improved parameter bounds for set-membership EIV problems. *International Journal of Adaptive Control and Signal Processing*. 2011;25(3):208-27. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.1218.

34. Blesa J, Puig V, Saludes J. Identification for passive robust fault detection using zonotope-based set-membership approaches. *International Journal of Adaptive Control and Signal Processing*. 2011;25(9):788-812. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.1242.

35. Casini M, Garulli A, Vicino A. A linear programming approach to online set membership parameter estimation for linear regression models. *International Journal of Adaptive Control and Signal Processing*. 2017;31(3):360-78. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.2701.

36. Wittich F, Kroll A. Approximation of the Feasible Parameter Set in Bounded-Error Parameter Estimation of Takagi-Sugeno Fuzzy Models for Large Problems by Using a Ray Shooting Method. *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2022:1-7.

37. Feng X, Corliss GF, Kelnhofer RW. A new interval bounding algorithm for parameter estimation from bounded-error data. *IFAC Proceedings Volumes*. 1999;32(2):5053-8. 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July. Available from: https://www.

sciencedirect.com/science/article/pii/S1474667017568603.

38. Avis D. *A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm*. Basel: Birkhäuser Basel. 2000:177-98. Available from: https://doi.org/10.1007/978-3-0348-8438-9_9.

39. Avis D. lrs home page. 2021. http://cgm.cs.mcgill.ca/~avis/C/lrs.html.

40. Løhne A, Weißing B. Bensolve, version 2.1. 0. A Free Vector Linear Program Solver. 2017.

41. Löhne A, Weißing B. The vector linear program solver Bensolve – notes on theoretical background. *European Journal of Operational Research*. 2017;260(3):807-13. Available from: https://www.sciencedirect.com/science/article/pii/S0377221716300765.

42. Billard L, Diday E. Regression Analysis for Interval-Valued Data. *Data Analysis, Classification, and Related Methods*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2000:369-74.

43. Billard L, Diday E. Symbolic Regression Analysis. *Classification, Clustering, and Data Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2002:281-8.

44. Lima Neto A, de Carvalho AT. Centre and Range method for fitting a linear regression model to symbolic interval data. *Computational Statistics Data Analysis*. 2008;52(3):1500-15.

45. Shigley JE, Mischke CR. *Mechanical Engineering Design*. New York, McGraw-Hill. 1989:135-48.

46. Adafruit electronic components/ HC-SR04 ultrasonic sensor specification page [cited 2023 02 February]. Available from: http://www.adafruit.com/product/4007.

47. 28BYJ-48 – 5V stepper motor specification sheet [cited 2023 02 February]. Available from: https://www.mouser.com/datasheet/2/758/stepd-01-data-sheet-1143075.pdf.

APPENDIX A: ALGORITHMS

Algorithm A.1. V_L determination algorithm

Require: x_p, L_p, U_p (for $p = 1$ to N)
Set a and b as empty
Set V_L as empty
Set pivot index $i = 1$
while $i < N$ **do**
    for $j = i + 1$ to N **do**
        Calculate $m_{ij} = \frac{(L_j - L_i)}{(x_j - x_i)}$ of line through (x_i, L_i) and (x_j, L_j)
    end for
    Find the index of maximum slope $J = \arg\max_j(m_{ij})$.
    In case of multiple maxima, use the greatest J.
    Calculate intercepts $n_i = L_i - m_{iJ}x_i$
    // Check the corridor condition
    if $m_{iJ}x_k + n_i < U_k$ for $k = 1, \cdots, N$ **then** Append n_i, m_{iJ} to V_L
        if a is empty **then**
            $a = i$
        end if
    end if
    Set pivot index $i = J$
end while
Output: Output V_L, a, b

Algorithm A.2. $l_{\min}$ and $l_{\max}$ determination algorithm

Require: x_q, L_q, U_q (for $q = 1$ to N), a, b, c, d
if $V_L \neq \emptyset, V_U \neq \emptyset$ **then**
    Calculate $m_{\min}$ and $n_{\min}$ of $l_{\min}$ through (x_b, L_b) and (x_c, U_c)
    Calculate $m_{\max}$ and $n_{\max}$ of $l_{\max}$ through (x_a, L_a) and (x_d, U_d)
    Go to Exit
else if $V_L \neq \emptyset, V_U = \emptyset$ **then**
    for $i = a + 1$ to $b - 1$ **do**
        Calculate m_i of the line through (x_a, L_a) and (x_i, U_i)
    end for
    Find the index of the minimum slope $I = \arg\min_i(m_i)$
    Set upper bound point index $w = I$
    Calculate $m_{\min}$ and $n_{\min}$ of $l_{\min}$ through (x_b, L_b) and (x_w, U_w)
    Calculate $m_{\max}$ and $n_{\max}$ of $l_{\max}$ through (x_a, L_a) and (x_w, U_w)
    Go to Exit
else if $V_L = \emptyset, V_U \neq \emptyset$ **then**
    for $i = c + 1$ to $d - 1$ **do**
        Calculate m_i of the line through (x_i, L_i) and (x_c, U_c)
    end for
    Find the index of the maximum slope $I = \arg\max_i(m_i)$
    Set lower bound point index $z = I$
    Calculate $m_{\min}$ and $n_{\min}$ of $l_{\min}$ through (x_z, L_z) and (x_c, U_c)
    Calculate $m_{\max}$ and $n_{\max}$ of $l_{\max}$ through (x_z, L_z) and (x_d, U_d)
    Go to Exit
end if
Output: $m_{\min}, n_{\min}, m_{\max}, n_{\max}$

Algorithm A.3. Determination of V_L and r_L

Require: $x_i^L,\ L_i,\ i = 1, \ldots, N_L$

Set $V_L, h_L, \overrightarrow{d_1}, \overrightarrow{d_2}$ as empty
if $|L| = 1$ **then**
  Save $\overrightarrow{d_1} = [-x_1^L, 1]^T \quad \overrightarrow{d_2} = [x_1^L, -1]^T$
  Calculate the position of $l_0 = sign(\overrightarrow{d_1}) \times Inf$ and save in V_L
  Calculate the position of $l_p = sign(\overrightarrow{d_2}) \times Inf$ and save in V_L
else if $|L| > 1$ **then**
  Save $\overrightarrow{d_1} = [-x_1^L, 1]^T \quad \overrightarrow{d_2} = [x_1^L, -1]^T$
  Calculate the position of $l_0 = sign(\overrightarrow{d_1}) \times Inf$ and save in V_L
  Set pivot index $P = 1$
  while $P < N_L$ **do**
    for $j = P + 1$ to N_L **do**
      Calculate slope $m_{Pj} = \frac{(L_j - L_P)}{(x_j^L - x_P^L)}$
    end for
    Find the maximum slope $m_{PJ} = \arg\max_j(m_{Pj})$
    Calculate intercept $n_P = L_P - m_{PJ}x_P^L$
    Save n_P, m_{PJ} in V_L
    Save (x_P^L, L_P) in h_L
    if there are multiple maximums in m_{PJ} **then**
      Set J point index as the rightmost contact point
    end if
    $P = J$
  end while
  Save (x_P^L, L_P) in h_L
  Calculate the position of $l_p = sign(\overrightarrow{d_2}) \times Inf$ and save in V_L
end if
Output $V_L, h_L, \overrightarrow{d_1}, \overrightarrow{d_2}$

Algorithm A.4. Determination of V_U and r_U

Require: $x_i^U,\ U_i,\ i = 1, \ldots, N_U$

Set $V_U, h_U, \overrightarrow{e_1}, \overrightarrow{e_2}$ as empty
if $|U| = 1$ **then**
    Save $\overrightarrow{e_1} = [-x_{N_U}^U, 1]^T \quad \overrightarrow{e_2} = [x_1^U, -1]^T$
    Calculate the position of $u_0 = sign(\overrightarrow{e_1}) \times Inf$ and save in V_U
    Calculate the position of $u_q = sign(\overrightarrow{e_2}) \times Inf$ and save in V_U
else if $|U| > 1$ **then**
    Save $\overrightarrow{e_1} = [-x_{N_U}^U, 1]^T \quad \overrightarrow{e_2} = [x_1^U, -1]^T$
    Calculate the position of $u_0 = sign(\overrightarrow{e_1}) \times Inf$ and save in V_U
    Set pivot index $P = N_U$
    while $P > 1$ **do**
        for $j = P - 1$ to 1 **do**
            Calculate slope $m_{Pj} = \frac{(U_j - U_P)}{(x_j^U - x_P^U)}$
        end for
        Find the maximum slope $m_{PJ} = \arg\max_j(m_{Pj})$
        Calculate intercept $n_P = U_P - m_{PJ} x_P^U$
        Save n_P, m_{PJ} in V_U
        Save (x_P^U, U_P) in h_U
        if there are multiple maximums in m_{PJ} **then**
            Set J as the leftmost contact point index
        end if
        $P = J$
    end while
    Save (x_P^U, U_P) in h_U
    Calculate the position of $u_q = Unit(\overrightarrow{e_2}) \times Inf$ and save in V_U
end if
Output $V_U, h_U, \overrightarrow{e_1}, \overrightarrow{e_2}$

Algorithm A.5. End status algorithm

Require: $x_1^L, x_{N_L}^L, x_1^U, x_{N_U}^U$

if $x_1^L \geq x_{N_U}^U$ **then**
    Set $Ends = [0, 1]$ *(open top, closed bottom)*
else
    if If $x_{N_L}^L \leq x_1^U$ **then**
        Set $Ends = [1, 0]$ *(closed top, open bottom)*
    else
        Set $Ends = [1, 1]$ *(closed top, closed bottom)*
    end if
end if
Output: $Ends$

Algorithm A.6. Top intersecting algorithm

```
Require: V_L, V_U, h_L, h_U
Set lower bound vertex index i to 1
Set upper bound vertex index j to 1
Set lower bound segment index I_T to 1
Set upper bound segment index J_T as empty
Set typ to 1 (typ = 1 test of V_U elements, typ = 2 test of V_L elements)
while true do
    if typ = 1 then
        for j = j to q do
            if u_j is feasible against h_{I_T} then
                Break For Loop
            end if
        end for
        if j = J_T then
            Go To Exit
        else
            J_T = j and typ = 2
        end if
    else if typ = 2 then
        for i = i to p do
            if l_i is feasible against k_{J_T} then
                Break For Loop
            end if
        end for
        if i = I_T then
            Go To Exit
        else
            I_T = i and typ = 1
        end if
    end if
end while
Exit Here: Output I_T and J_T
```

Algorithm A.7. Bottom intersecting algorithm

Require: $V_L,\ V_U,\ h_L,\ h_U$
Set lower bound vertex index i to $p-1$
Set upper bound vertex index j to $q-1$
Set lower bound segment index I_B to p
Set upper bound segment index J_B as empty
Set typ to 1 ($typ = 1$ test of V_U elements, $typ = 2$ test of V_L elements)
while true **do**
    if $typ = 1$ **then**
        for $j = j$ to 0 **do**
            if u_j is feasible against h_{I_B} **then**
                Break For Loop
            end if
        end for
        if $j+1 = J_B$ **then**
            Go To Exit
        else
            $J_B = j+1$ and $typ = 2$
        end if
    else if $typ = 2$ **then**
        for $i = i$ to 0 **do**
            if l_i is feasible against k_{J_B} **then**
                Break For Loop
            end if
        end for
        if $I_B + 1 = I$ **then**
            Go To Exit
        else
            $I_B = i+1$ and $typ = 1$
        end if
    end if
end while
Exit Here: Output I_B and J_B

Algorithm A.8. MCM algorithm

Require: $V_L, V_U, h_L, h_U, \overrightarrow{d_1}, \overrightarrow{d_2}, \overrightarrow{e_1}, \overrightarrow{e_2}$

Set $V, d, V_{\min}, V_{\max}$ as empty sets
if $|h_L| = 0$ and $|h_U| = 0$ **then**
Set $V = \{(0,0)\}$ $D = \{[1,0]^T, [0,1]^T\}$ and Go To Exit
else if $|h_L| = 0$ **then**
$V = V_U$ $D = \{\overrightarrow{e_1}, \overrightarrow{e_2}\}$ and Go To Exit
else if $|h_U| = 0$ **then**
$V = V_L$ $D = \{\overrightarrow{d_1}, \overrightarrow{d_2}\}$ and Go To Exit
else
Find $Ends$
if $Ends = [1,0]$ (close top-open bottom) **then**
Set $D = \{\overrightarrow{d_2}, \overrightarrow{e_2}\}$
Find the indices of the intersecting segments of the top I_T, J_T
Intersect $h^L_{I_T}$ and $h^U_{J_T}$ to determine $l_{\max}$ and set $V_{\max} = \{l_{\max}\}$
Delete vertices in V_L that are above $h^L_{I_T}$
Delete vertices in V_U that are above $h^U_{J_T}$
Reorder V_U elements in reverse
else if $Ends = [0,1]$ (open top-close bottom) **then**
Set $D = \{\overrightarrow{d_1}, \overrightarrow{e_1}\}$
Find the indices of the intersecting segments of the bottom I_B, J_B
Intersect $h^L_{I_B}$ and $h^U_{J_B}$ to determine $l_{\min}$ and set $V_{\min} = \{l_{\min}\}$
Delete vertices in V_L that are below $h^L_{I_B}$
Delete vertices in V_U that are below $h^U_{J_B}$
Reorder V_U elements in reverse
else if $Ends = [1,1]$ (close top-close bottom) **then**
Find the indices of the intersecting segments of the top I_T, J_T
Find the indices of the intersecting segments of the bottom I_B, J_B
if $I_T \geq I_B$ and $J_T \geq J_B$ **then**
Intersect $h^L_{I_T}$ and $h^U_{J_T}$ to determine $l_{\max}$ and set $V_{\max} = \{l_{\max}\}$
Intersect $h^L_{I_B}$ and $h^U_{J_B}$ to determine $l_{\min}$ and set $V_{\min} = \{l_{\min}\}$
Delete vertices in V_L that are above $h^L_{I_T}$ and below $h^L_{I_B}$
Delete vertices in V_U that are above $h^U_{J_B}$ and below $h^U_{J_B}$
Reorder V_U elements in reverse
else
Go To Exit
end if
end if
Set $V = \{V_{\min}, V_U, V_{\max}, V_L\}$
Eliminate the duplicate vertices in V and Go To Exit
end if
Exit Here: Output: $V, D, V_L, V_U, V_{\min}, V_{\max}$

Algorithm A.9. End point finder algorithm

Require: A, b, v, E
Create an empty vertex list V^*
Create an empty extreme ray list r^*
Set the list of edges to be tracked E^* as the edges connected to v
Compare E^* and E and remove the same edges from E^*
for $i = 1$ to Number of edges in E^* **do**
    Set $flag$ to 0
    for $j = 1$ to N : excluding the constraint indices of v **do**
        Combine the constraints in E_i^* and $\{A_j, b_j\}$
        Find the vertex position $\overline{x}$
        if vertex pass the test $0 \leq b - A\overline{x}$ **then**
            Append $\{E_i^*, \{A_j, b_j\}\}$ to V^*
            Set $flag$ to 1
            Break the j Loop
        end if
    end for
    if $flag = 0$ **then**
        Append E_i^* to r^*
    end if
end for
Append E^* to E
Output: V^*, r^*, E

Algorithm A.10. Edge tracking algorithm

Require: A, b, v_i
Create an empty vertex list V
Append v_i to V
Create an empty extreme ray list r
Create an empty tracked edges list E
Set $i = 0$
Set Q = Number of vertices in V
while $i < Q$ **do**
    $i = i + 1$ A(l)gorithm A.9 with A, b, V_i, E and get V^*, r^*, E
    Append V^* to V
    Append d^* to d
    Update Q
end while
Output: V and r