Applications of Convex Optimization

An internship report submitted

in partial fulfillment for the award of the degree of

Bachelor of Technology

in

Electronics & Communication Engineering(Avionics)

by

Debajyoti Chakrabarti

Department of Avionics

Indian Institute of Space Science and Technology

Thiruvananthapuram, India

June-July,2018

Certificate

This is to certify that the internship report titled ***Applications of Convex Optimization*** submitted by **Debajyoti Chakrabarti**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering(Avionics)**, is a bonafide record of the internship work carried out by him/her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

B.V. Ramkiran
Scientist/Engineer 'SD'

Chinna Ponnu V
Division Head,CDAD-3
Control Design & Simulation Group
(CDSG)
ISRO Satellite Centre
Bengalore-560017

Place: Thiruvananthapuram
Date: September 18, 2018

Declaration

I declare that this internship report titled ***Applications of Convex Optimization*** submitted in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering(Avionics)** is a record of original work carried out by me under the supervision of **B.V. Ramkiran**,**Scientist/Engineer'SD',CDSG group, ISAC,Bengalore**, and has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
Date: September 18, 2018

Debajyoti Chakrabarti
(SC15B084)

Acknowledgements

I would like to express my special thanks of gratitude to my guide **Mr. B.V. Ramkiran,Scientist/Engineer'SD',CDSG group, ISAC,Bengalore** who gave me the golden opportunity to do this wonderful project on the topic **'Applications of Convex Optimization**.Without his guidance and continuous support, this project would have never been completed successfully within the limited time frame.

I convey my special thanks to **Mr. Suraj Kumar,Scientist/Engineer 'SC',CDSG group, ISAC,Bengalore** for helping me at various stages of the project.

I am thankful to **Mrs. Chinna Ponnu V,Division Head,CDAD-3,CDSG group, ISAC,Bengalore** for giving her valuable suggestions during project review.

I would like to thank my parents,who have been continuous source of inspiration throughout the project work. I also take this opportunity to thank **Department of Avionics,IIST** and all my friends.

Debajyoti Chakrabarti

Abstract

Convex Optimization has applications in various engineering problems, where some kind of distinction is to be made between two sets of data. In this project, two design problems are considered, where Convex Optimization is applied for design of certain coefficients, considering available constraints in each of the problem .
First problem deals with design of convex boundary for *'Chandrayan-2 Lander'* boundary data set.As per as the mission requirement, *Chandrayan mission-2'*, there will be a lander which will land on surface of moon on a pre-determined location from the orbiter rotating on a fixed orbit. Now for a given mass, altitude and down-range for the lander, it will not be always possible to land exactly at the precise landing site, considering the real-world scenario. The set of data-points(set of coordinates(downrange,altitude,mass)) , for which landing can be achieved constitutes the *feasible region*.Remaining all valid data-points for which landing is not possible, forms the *infeasible region*.Boundary points between these two regions, is important because during time of mission, it will be possible for the *on-board computer*, to take decision for landing, having the current values of downrange, altitude, mass. Boundary is expressed in terms of function because it is easy to store the coefficients of function instead of storing all the boundary data points, and it will ensure usage of less memory; which is very crucial for space application.A convex function is used to implement that boundary on the given boundary data set using the Optimization techniques, considering some necessary constraints on that function.

Satellite's attitude control is one of the most important tasks in spacecraft design.Satellite should be properly oriented for various purposes like: solar panel operation, or pointing antenna towards Earth etc. Satellite's desired attitude is represented with respect to some frame of reference, and controller is used to achieve and maintain that attitude throughout its life cycle.There are many ways to control the attitude. In the second problem, PD controller is designed to control the attitude of the satellite along three axes.Convex Optimization is used to design the controller considering practical constraints on performance of satellite under closed loop operation. Finally the simulation results are compared against desired response.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

LP	Linear Programming
DCM	Direction Cosine Matrix
ACS	Attitude Control System
CMG	Control Moment Gyros
PD	Proportional-Derivative
RK	Runge-Kutta Method

Nomenclature

H	Half space
Θ	Convex set
h	Angular momentum
J	Moment of inertia
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw Angle
q	Quaternion
ω	Angular velocity
ζ	Damping ratio
Ts	Settling time
ω_n	Natural frequency of a system
T_c	Control torque
θ_{Sat}	Saturation angle

Chapter 1

Introduction

Convex Optimization is the field of Optimization where convex functions are minimized over convex sets. Convexity makes it easier for optimization because local minimum is a global minimum here, hence first order conditions are sufficient to ensure optimality. In the scope of this project Convex Optimization is used to address two problems.
First problem is related to 'Chandrayan-2' mission. Here a set of data given, for which a boundary is to be formulated, which will mathematically describe those data points. Each data point has three features, namely: downrange, altitude, mass. The constraint considered in the problem is that, the curve described the convex function must lie inside the convex region described by the boundary datapoints.
The next problem is to control the attitude of a satellite. Control of orientation of a satellite is the attitude control problem, and it is necessary to maintain the attitude at every point in the orbit for various purposes. The designed controller is supposed to control the attitude of satellite along the three axis namely: Roll, Pitch, Yaw. This type of control is called as: *'three axis control'*.PD controller is used for controlling attitude of the satellite. The reason behind choosing PD controller is that PD controller requires future values of the control variable which is available in case of satellite attitude control.While designing the controller, practical constraints on controlling the attitude of satellite is to be considered, which calls for optimization techniques.

1.1 Objectives of the Project

- Study and understand linear programming methods.
- Implement the following linear programming algorithms:
  (i) Simplex Algorithm (ii) Karmarkar's Algorithm

- Test the implemented algorithms for sample problems on boundary design and modify the algorithm

- Apply the modified algorithm for **'Chandrayan-2 Lander'** boundary data set and design convex boundary for the given data set.
- Study and understand various attitude representation techniques, compare and choose one of them for attitude representation of satellite.
- Study and implement spacecraft dynamics.
- Design PD controller(three axis attitude controller) on the basis of available constraints on satellite's attitude control using Convex Optimization techniques developed.
- Test the controller on the satellite's dynamics under close loop condition and compare the response with the desired response from satellite.

1.2 Related Literature

The main reference for the first problem of the project is [1].This book serves as foundation for Mathematical concepts and linear Programming methods discussed on chapter 2 and 3. This article[2] gives an better insight of karmarkar's Algorithm with detailed proof of some properties. The main references for second problem are:[3], [4].Basic concepts of Satellite dynamics(governing equation,attitude representation), concept on different methods for attitude control are taken from these books [3] [4] .Different methods for Attitude representation is discussed thoroughly in [5]. This paper is used for concepts of DCM, Euler angles, Quaternions and their inter conversions.

1.3 Outline of the Project

- **Chapter 1** contains scope, objectives of the project.It also contains brief description of content of each chapter.
- **Chapter 2** contains basic mathematical concepts needed for the project and also brief introduction to unconstrained optimization techniques.

- In **Chapter 3** analysis is done on Linear Programming methods (constrained optimization) with description of Simplex and Karmarkar's Algorithm.
- In **Chapter 4**, first problem of the project is formulated, which is *Convex Boundary Design for 'Chandrayan-2 Lander' dataset*.
- In **Chapter 5**, optimization techniques developed previously are tested for sample problems on boundary design and modifications are made. Finally modified optimization method is applied on 'Chandrayan-2 Lander' data set, followed by analysis of generation of simulation results and detailed analysis in made on them.
- **Chapter 6** provides brief introduction on the second problem, *'gain Design of a Controller'*. This chapter also contains comparative study on various satellite's attitude techniques, formulation of spacecraft model for which controller is to be designed.
- In **chapter 7**, gain design problem is formulated and gain values for PD controller is designed using optimization techniques. The gain values are validated for single axis attitude control. Finally the controller is used for *three axis attitude control* of the spacecraft, and the performance is compared to desired response.
- **Chapter 8** contains Conclusions & Recommendations, which includes inferences based on the results of the two problems together with suggestions for future work.

Chapter 2

Basic Mathematical Concepts

2.1 Introduction

Mathematical concepts and Linear Programming Techniques which have been the background of the project, are discussed briefly in next few chapters. Here in this chapter the basic Mathematical concepts are discussed accordingly.

2.2 Linear Transformations

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called a *Linear Transformation* if
1.$f(ax) = af(x)$ for every $x \in \mathbb{R}^n$ and $a \in \mathbb{R}$
2. $f(x + y) = f(x) + f(y)$ for every $x, y \in \mathbb{R}^n$

Specifically a *Linear Transformation* can be represented by matrix $A \in \mathbb{R}^{m\times n}$. Consider $x \in \mathbb{R}^n$ a given vector and it is represented by $x^{'}$ w.r.t the basis and $y^{'}$ be representation of $y \in \mathbb{R}^m$ then,
$y^{'} = Ax^{'}$
In case of natural bases used we can write the transformation as
$f(x) = A(x)$
Now consider $(e_1, e_2, ..., e_n)$ and $(e_1^{'}, e_2^{'}, ..., e_n^{'})$ be two bases in $\mathbb{R}^n$, then the matrix
$T = (e_1^{'}, e_2^{'}, ..., e_n^{'})^{-1}(e_1, e_2, ..., e_n)$ is termed as *transformation matrix* from $(e_1, e_2, ..., e_n)$ to $(e_1^{'}, e_2^{'}, ..., e_n^{'})$ and we can write
$(e_1, e_2, ..., e_n) = (e_1^{'}, e_2^{'}, ..., e_n^{'})T$
i.e. i th column of T is the coordinates of e_i w.r.t to the given basis $(e_1^{'}, e_2^{'}, ..., e_n^{'})$.

2.3 Orthogonal Projections

Consider V is a subspace in $\mathbb{R}^n$, then *orthogonal complement* of V , denoted by $V^\perp$, consists of vectors which are perpendicular to vectors in V i.e.
$V^\perp = \{x : v^T x = 0 \text{ for all } v \in V\}$
Consider $x \in \mathbb{R}^n$, then x can be uniquely represented as
$x = x_1 + x_2$, where $x_1 \in V$ and $x_2 \in V^\perp$
This is called *orthogonal decomposition* of x w.r.t V and x_1 , x_2 are *orthogonal projections* of x on the subspaces V and $V^\perp$.
Consider linear transformation P is an *orthogonal projector* onto V if for all $x \in \mathbb{R}^n$, we have $Px \in V$ and $x - Px \in V^\perp$. Consider a matrix $A \in \mathbb{R}^{m \times n}$, *range* of A is
$R(A) = \{Ax : x \in \mathbb{R}^n\}$,
and *nullspace* or *kernel* of A is
$N(A) = \{x \in \mathbb{R}^n : Ax = 0\}$

Theorem 2.1. *let A be a matrix. Then $R(A)^\perp = N(A^T)$ and $N(A)^\perp = R(A^T)$.*

2.4 Geometrical Concepts

Some geometrical concepts used in the project, are described in this scope.

2.4.1 Hyperplane

consider $u_1, u_2, ..., u_n, v \in \mathbb{R}$, where atleast one u_i is non zero.Now consider the equation,

$$u_1 x_1 + u_2 x_2 + ...u_n x_n = v \tag{2.1}$$

set of all points $x = [x_1, x_2, ..., x_n]^T$ satisfying the above equation is called a *hyperplane* of $\mathbb{R}^n$. It is described by $\{x \in \mathbb{R}^n : u^T x = v\}$, where $u = [u_1, u_2, ..., u_n]^T$. The hyperplane

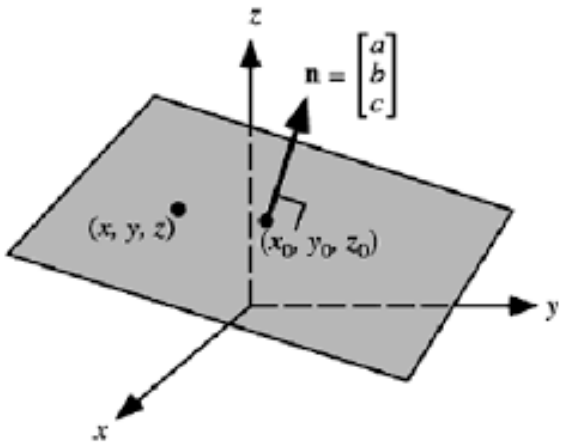

Figure 2.1: The hyperplane $H = \{x \in \mathbb{R}^n : u^T(x-a) = 0\}$
[6]

$H = \{x : u_1x_1 + ...u_nx_n = v\}$ divides $\mathbb{R}^n$ into two *half spaces*. One half consists of points satisfying the inequality $u_1x_1 + u_2x_2 + ... + u_nx_n \geq v$,
$H_+ = \{x \in \mathbb{R}^n : u^Tx \geq v\}$
Other half-space consists of points inequality $H_- = \{x \in \mathbb{R}^n : u^Tx \leq v\}$
The half-space H_+ is called the *positive half-space*, and the half-space H_- is called the *negative half-space*.
Consider $a = [a_1, a_2....a_n]^T$ be any point lying on hyperplane H.Then we have,
$u^Tx - v = u^Tx - v - (u^Ta - v)$
$= u^T(x-a)$
$= u_1(x_1 - a_1) + u_2(x_2 - a_2) + ... + u_n(x_n - a_n) = 0$
$(x_i - a_i), i = 1(.)n$ are components of vector $x - a$. So it can be concluded hyperplane consists of points x for which $\langle u, x-a\rangle = 0$ i.e. hyperplane consists of points x for which vector u (in the figure u is $n = [a, b, c]^T$ and a is $[x_0, y_0, z_0]^T$) and vector $x - a$ are *orthogonal*. So u is *normal* to the hyperplane H . It can be then said that H_+ consists of points for which $\langle u, x-a\rangle \geq 0$ and similarly points on H_- satisfies $\langle u, x-a\rangle \leq 0$.

2.4.2 Linear Variety

A *Linear Variety* set is of the form $\{x \in \mathbb{R}^n : Ax = b\}$ where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^n$
If dim $N(A) = r$, *linear variety* is said to have dimension r. If $b = 0$ *linear variety* can be taken as *Subspace*.If $r < n$, then this set is intersection of finite number of hyperplanes.

2.4.3 Convex Set

Consider two points u and $v \in \mathbb{R}^n$, then the line passing through this two points is given by the set,
$\{w \in \mathbb{R}^n : w = \alpha u + (1-\alpha)v, \alpha \in [0,1]\}$ Point $w = \alpha u + (1-\alpha)v, \alpha \in [0,1]$ is called *convex combination* of points u and v.

Definition 2.1. *A set $\Theta \subset \mathbb{R}^n$ is convex if for all $u, v \in \Theta$, line joining u and v lies in Θ.*

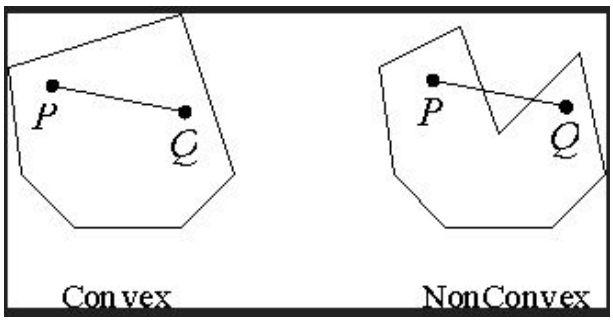

Figure 2.2: Convex and non Convex set
[1]

Here two statements related to convex sets are notable, these theorems state the properties of *convex set*.

Theorem 2.2. *Followings are the properties of Convex sets:*
a. *If Θ is a convex set and α is a real number, then set*
$\alpha\Theta = \{x : x = \alpha v, v \in \Theta\}$ *is also convex.*
b. *If Θ_1 and Θ_2 are convex sets, then set*
$\Theta_1 + \Theta_2 = \{x : x = v_1 + v_2, v_1 \in \Theta_1, v_2 \in \Theta_2\}$ *is also convex.*
c. *Intersection of any number of convex sets is also convex set.*

Now consider a point x in a convex set Θ ,

Definition 2.2. *x is said to be extreme point of Θ, if there are no other distinct points $x_1, x_2 \in \Theta$ such that $x = \alpha x_1 + (1-\alpha)x_2$ for some $\alpha \in (0,1)$.*

So extreme point does not lie strictly within line segment connecting two other points of the set.

2.4.4 Polytopes and Polyhedra

Consider Θ be a *convex set* and y be boundary point of Θ, a hyperplane passing through y is called *Supporting hyperplane* of Θ, if the set Θ lies completely on of the two half spaces generated by the hyperplane in $\mathbb{R}^n$.
From theorem 2.2 it is known that intersection of *convex sets* is also convex. Now *half-spaces* H_+ and H_- are *convex sets* in $\mathbb{R}^n$, so it can be concluded that intersection of finite number of *half-spaces* is a *convex set*.

Definition 2.3. *A set that can be represented as intersection of finite number of half-spaces is called a Convex Polytope and a non-empty, bounded polytope is called a Polyhedron.*

Let Θ be a *convex polyhedron*, then $\exists\ k \leq n$, such that a *linear variety* of dimension k contains Θ , but it is not contained in any $(k-1)$ dimensional *linear variety* of $\mathbb{R}^n$, also $\exists$ only one k dimensional linear variety containing Θ, which is the carrier of that *polyhedron*. $(k-1)$ dimensional polyhedra which forms boundary of k dimensional polyhedron are called *faces* of the polyhedron.A zero dimensional face is the *vertex* of polyhedron, one dimensional face is the *edge* of the polyhedron.

2.5 Concepts of Unconstrained Optimization

Consider function $f : \mathbb{R}^n \to \mathbb{R}$ be a real valued function and it is to be minimized.This function is called the *Objective or Cost function*.So the problem can be put as

$$\begin{aligned} \textbf{minimize} \quad & f(x) \\ \textbf{subject to} \quad & x \in \Omega, \end{aligned}$$

where Ω is the *feasible set*.
So optimization problem lies in finding "best" vector x over Ω and result is that f has *minimum* value at that point of the feasible set.
Depending on Ω , if $\Omega = \mathbb{R}^n$, the problem is *unconstrained* optimization problem and when $\Omega \subset \mathbb{R}^n$, the problem is *constrained* optimization problem.
Constraint $x \in \Omega$ is called *set constraint*. General form of set Ω is,
$\Omega = \{x : \alpha(x) = 0, \beta(x) < 0\}$, where α and β corresponds to *functional constraints*. For any function f two kind of minimizer can be there namely, *Local Minimizer* and *Global*

Minimizer.A point x to be a *Local minimizer* of f, it should satisfy *first and second order Necessary conditions*, if x also satisfies *second order Sufficient condition*, then x can be called as *strict local minimizer*.
In case of unconstrained optimization, there are different search methods for finding the minimum of function.Line search methods plays important role for optimization problems.For one dimensional case few algorithm can be noted,

- Golden Section Search
- Fibonacci Search
- Newton's Method
- Secant Method

In case function on $\mathbb{R}^n$ also, there are several algorithms to find optimum value of function.

- Gradient Methods (Method of Steepest Descent)
- Newton-Raphson Methods(with levenberg-marquradt modification)
- Conjugate Direction Methods

Among these algorithms Newton-Raphson method is used in *Karmarkar's Algorithm*(optimization algorithm used to address our problem later), and here Newton-Raphson method is discussed briefly for the sake of the problem.

2.5.1 Newton-Raphson Method

Basic idea behind Newton-Raphson Method is that, given a starting point, a *quadratic approximation* of the function is made whose first and second derivatives match with that of the objective function at that point. Then this approximation function is minimized. This minimizer is used as starting point in the next step and the iteration continues.

Consider $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable, objective function. *Taylor series* approx of it about current point $x^{(k)}$ is,

$$f(x) \approx f(x^{(k)}) + (x - x^{(k)})^T g^{(k)} + \frac{1}{2}(x - x^{(k)})^T F(x^{(k)})(x - x^{(k)}) \doteq q(x) \qquad (2.2)$$

where $g^{(k)} = \nabla f(x^{(k)})$. From first order necessary condition,

$$0 = \nabla q(x) = g^{(k)} + F(x^{(k)})(x - x^{(k)}) \tag{2.3}$$

if $F(x^{(k)}) > 0$ then q achieves a minimum at

$$x^{(k+1)} = x^{(k)} - F(x^{(k)})^{-1} g^{(k)} \tag{2.4}$$

This is the recursive formula for Newton's method.Here $F(x^{(k)})$ denotes the *Hessian Matrix*.
Basically kth iteration can be written as,
1. Solve $F(x^{(k)})d^{(k)} = -g^{(k)}$ for $d^{(k)}$
2. Set $x^{(k+1)} = x^{(k)} + d^{(k)}$. If we neglect the problem of solving $n \times n$ linear system in step 1, other major **Problems** associated with Newton-Raphson method are:

- *Hessian* matrix $F(x^{(k)})$ should always be *positive definite*,i.e. $F(x^{(k)}) > 0$.
- Starting point x^0 should be sufficiently close to the minimizer value x^*.

Although $F(x^{(k)})$ is not *positive definite*, search direction $d^{(k)} = -F(x^{(k)})^{-1} g^{(k)}$ may not be descent direction, in that case modification to Newton-Raphson method is made using *Levenberg-Marquardt* technique(Discussion of it is beyond the scope of this analysis).

- Newton-Raphson method is the underlying method for *interior point methods*, discussed in the next chapter.

Constrained optimization with linear objective function and linear constraints (which, in other words, called as *linear programming*) are discussed in details in the next chapter.

Chapter 3

Concepts of Linear programming

In this chapter basic concepts and algorithms of *linear programming* are discussed. This concepts along with the algorithm help to build the fundamentals to solve the problem addressed in the project.

3.1 Introduction

Linear Programming is used to maximize or minimize a linear objective function, where decision variables are subject to linear constraints. It is a special kind of general constrained optimization problem.A *standard form* of linear program is of the form,

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$.
Set of points satisfying the constraints can be represented as intersection of finite number of closed half spaces. So, the constraints forms a *Convex Polytope*.If this polytope is assumed to be non empty and bounded, then these constraints define a *Polyhedron* M in $\mathbb{R}^n$.
Say H be a *hyper plane of support* of M . If dimension of M is less than n, then set of points common to M and hyperplane H, coincides with M. If dimension of M is equal to n, then set of points common to H and M forms a face of the polyhedron. If this face is $(n-1)$ dimensional, then this *hyper plane of support* is only one, called the *carrier* of the face. If the dimension of the face is less than $(n-1)$, then there exists infinite number of hyperplane of support that whose intersection with M generates this face.
In linear programming, we need to find extremal of the linear objective function $f(x) = c^T x = c_1 x_1 + c_2 x_2 + ... + c_n x_n$ on the convex polyhedron M.

In case of problems where constraints are not in standard form, it is converted to standard form, by adding *slack* or *surplus* variables. In case of constraints like $Ax \geq b$, it is converted to standard form by subtracting *slack* variables, say $y_1, y_2....y_m$ from the m constraint equations and the problem is reformulated as,

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax - I_m y = b \\ & x \geq 0, y \geq 0 \end{aligned}$$

where I_m is an identity matrix
Similarly less than constraint can be converted to equality constraint by adding *surplus* variables.

3.1.1 Basic Solutions

Consider the system $Ax = b$, where $rank(A) = m$,
let, B be a square matrix whose columns are the m linearly independent columns of A. So A can be written as $A = [B, D]$, where D is a $m \times (n - m)$ matrix containing remaining columns of A. Solving the equation ,

$$Bx_B = b \tag{3.1}$$

vector x is constructed where $x = [x_B^T, 0^T]^T$, and it is in fact a solution of $Ax = b$.

Definition 3.1. *The solution $[x_B^T, 0^T]^T$ is called a basic solution to equation $Ax = B$ w.r.t. the basis B, and components of vector x_B are basic variables.*

If some of the basic variables of the basic solution are zero, then it is called *degenerate basic solution*.
A vector x satisfying $Ax = b, x \geq 0$, is called *feasible solution*. A feasible solution that is also basic is *basic feasible solution*.

Definition 3.2. *Vector x that yields minimum value of the objective function $c^T x$,over the set of vectors satisfying $Ax = b, x \geq 0$,is called optimal feasible solution.*

3.1.2 Theorems

Here few theorems are stated which are important to find the solution of problems related to linear programming.

Theorem 3.1. ***Fundamental Theorem of LP.****Consider a linear program in standard form.*
1. If there exists a feasible solution, then there exists a basic feasible solution.
2. If there exists an optimal feasible solution, then there exists an optimal basic feasible solution.

.

Theorem 3.2. *Consider Ω be a convex set containing all feasible solutions,i.e. vector x satisfying*

$$Ax = b, x \geq 0 \tag{3.2}$$

where $A \in \mathbb{R}^{m \times n}$,$m < n$. Then x is an extreme point of Ω if and only if x is a basic feasible solution to $Ax = b, x \geq 0$.

From the above two theorems, set containing *extreme points* of constraint set $\Omega = x : Ax = b, x \geq 0$ is equal to the set of basic feasible solutions to $Ax = b, x \geq 0$.
Combining above observation with *fundamental theorem*, it can be said that to find solution of LP problem, only extreme points of the constraint set need to be checked.

3.2 Simplex Method

Simplex Algorithm is one of the basic algorithms available for solving linear programming problems.Before going to the main algorithm, some concepts related to simplex are discussed.

3.2.1 Canonical Augmented Matrix

Consider a linear system $Ax = b$,where $rank(A) = m$. By *elementary row operations*, the equation is converted in the following form,

$$\begin{aligned} x_1 \qquad\qquad & + y_{1m+1}x_{m+1} + ... + y_{1n}x_n = y_{10} \\ x_2 \qquad\qquad & + y_{2m+1}x_{m+1} + ... + y_{2n}x_n = y_{20} \\ & \vdots \\ x_m + y_{mm+1}x_{m+1} + ... + y_{mn}x_n = y_{m0} \end{aligned} \tag{3.3}$$

Which can be represented in matrix notation, $[I_m, Y_{m,n-m}]x = y_0$
This form is termed as *Canonical form* of the system w.r.t. basis $a_1, a_2, ...a_m$.The variables $x_1, ..., x_m$ are *basic variables* and other variables are *non basic variables*.
Corresponding basic solution is,

$$\begin{aligned} x_1 &= y_{10} \\ &\vdots \\ x_m &= y_{m0} \\ x_{m+1} &= 0 \\ &\vdots \\ x_n &= 0 \end{aligned} \tag{3.4}$$

that is, $\mathbf{x} = \begin{bmatrix} \mathbf{y_0} \\ 0 \end{bmatrix}$
Corresponding to the equation $Ax = b$, the *canonical augmented matrix* is,

$$[I_m, Y_{m,n-m}, y_0] = \begin{bmatrix} 1 & 0 & \dots & 0 & y_{1m+1} & \dots & y_{1n} & y_{10} \\ 0 & 1 & \dots & 0 & y_{2m+1} & \dots & y_{2n} & y_{20} \\ \dots & \dots & \dots & \dots & \dots & \ddots & \dots & \dots \\ 0 & 0 & \dots & 1 & y_{mm+1} & \dots & y_{mn} & y_{m0} \end{bmatrix} \tag{3.5}$$

From the arguments above, it can be said that $b = y_{10}a_1 + y_{20}a_2 + ... + y_{m0}a_m$,i.e. last columns of canonical augmented matrix are the coordinates of vector b w.r.t. basis $\{a_1, ..., a_m\}$. For any other column say jth ($j = 1(.)n$), entries of the column are coordinates of a_j w.r.t. basis $\{a_1, ..., a_m\}$.

3.2.1.1 Updating the Augmented Matrix

For a canonical representation of a system $Ax = b$, where first m columns are basic variables, If a basic vector a_p,$1 \leq p \leq m$ is to be replaced by a non basic vector a_q,$m < q \leq n$, following is done.
a_q in terms of old basis,

$$a_q = \sum_{i=1}^{m} y_{iq} a_i = \sum_{i=1, i \neq p}^{m} y_{iq} a_i + y_{pq} a_p \tag{3.6}$$

set of vectors $\{a_1, ..., a_{p-1}, a_q, a_{p+1}, ..., a_m\}$ is linearly independent if and only if $y_{pq} \neq 0$, solving we get

$$a_p = \frac{1}{y_{pq}} a_q - \sum_{i=1, i \neq p}^{m} \frac{y_{iq}}{y_{pq}} a_i \tag{3.7}$$

In terms of old augmented matrix any vector a_j, $m < j \leq n$ can be written as,

$$a_j = y_{1j} a_1 + y_{2j} a_2 + ... + y_{mj} a_m \tag{3.8}$$

combining last two equations we get,

$$a_j = \sum_{i=1, i \neq p}^{m} (y_{ij} - \frac{y_{pj}}{y_{pq}} a_i) + \frac{y_{pj}}{y_{pq}} a_q \tag{3.9}$$

Let the new augmented matrix be $y_{ij}{}'$, it can be written

$$y_{ij}{}' = y_{ij} - \frac{y_{pj}}{y_{pq}} y_{iq}, i \neq p \tag{3.10}$$

$$y_{pj}{}' = \frac{y_{pj}}{y_{pq}} \tag{3.11}$$

In this way new augmented matrix can be obtained from old augmented matrix by sing above formula. the above equations are called *pivot equation* and y_{pq} is called *pivot element*.

3.2.2 Simplex Algorithm

The main concept of simplex is to move from one basic feasible solution to other until a optimal basic feasible solution is obtained.Canonical augmented matrix plays an important

role here.Here one result is used to construct the algorithm, which is

$$c^T x = z_0 + \sum_{i=m+1}^{n} c_i - z_i x_i, \tag{3.12}$$

where $z_i = c_1 y_{1i} + ... + c_m y_{mi}$, where $i = m + 1(.)n$
Let $r_i = 0$ for $i = 1(.)m$ and $r_i = c_i - z_i$ for $i = m + 1(.)n$. r_i is called th *reduced cost coefficient*.

Theorem 3.3. *A basic feasible solution is optimal if and only if corresponding reduced cost coefficients are all non negative.*

Algorithm 3.1: The Simplex Algorithm

1. Form a canonical augmented matrix corresponding to initial basic feasible solution.
2. Calculate reduced cost coefficients corresponding to non basic variables.
3. If $r_j \geq 0$ for all j, then terminate the algorithm–*current basic feasible solution is optimal*.
4. Select q corresponding to most negative r_q.
5. If no $y_{iq} > 0$, stop–*solution is unbounded*; else, find $p = argmin_i\{y_{i0}/y_{iq} : y_{iq} > 0\}$.(If more than one i minimizes y_{i0}/yiq, then p is smallest of those index.)
6. Update canonical augmented matrix by pivoting about (p, q)th element.
7. Go back to step 2.

3.2.2.1 Two phase Simplex Algorithm

In the above algorithm stated, an initial solution is already taken.But if it is not given,there is $\binom{n}{m}$ ways of selecting m basic columns among n columns, if n is sufficiently large; choosing the initial guess is hard job. So a better way to proceed is as following.
Consider a linear program in standard form,

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Consider associated *artificial problem*,

$$\begin{aligned} \text{minimize} \quad & y_1 + y_2 + ... + y_m \\ \text{subject to} \quad & \begin{bmatrix} A, I_m \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = b \\ & \begin{bmatrix} x \\ y \end{bmatrix} \geq 0 \end{aligned}$$

where $y = [y_1, y_2, ..., y_m]^T$,are called *artificial variables*.It has initial basic feasible solution, $\begin{bmatrix} 0 \\ b \end{bmatrix}$
Here a proposition is stated.

Proposition 3.1. *The original LP problem has a basic feasible solution if and only if the associated artificial problem has an optimal feasible solution with objective function value zero.*

So the solution of artificial problem will have all $y_i = 0, i = 1(.)m$.So first n components contains basic variables.We can use this basic feasible solution as starting point of our original problem.
So in general Simplex algorithm will have two phases. In Phase I, artificial variables and artificial objective function are introduced, and initial basic feasible solution is guessed.In phase II,this solution is used to solve the original problem.
This concludes brief discussion on Simplex method.

3.3 Duality

Here a brief introduction to duality is given which will be used in *Karmarkar's Algorithm* in later part of this chapter.
Consider LP of the form,

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

This is called *primal problem*.Corresponding *dual problem* is,

$$\begin{aligned} \text{maximize} \quad & \lambda^T b \\ \text{subject to} \quad & \lambda^T A \leq c^T \\ & \lambda \geq 0 \end{aligned}$$

This is called *symmetric form of duality*.
Again consider a LP problem in standard form,

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Dual problem corresponding to this primal problem is,

$$\begin{aligned} \text{maximize} \quad & \lambda^T b \\ \text{subject to} \quad & \lambda^T A \leq c^T \end{aligned}$$

This is called *asymmetric form of duality*.(Here point to e noted is that $\lambda \geq 0$ condition is released.) This brief section on duality is concluded with the duality theorem.

Theorem 3.4. *If the primal problem(either in symmetric or asymmetric form) has an optimal solution, then so the dual problem has, and the optimal values of their respective objective functions are equal.*

3.4 Non-Simplex Methods

Although simplex method is successful to solve the optimization problems, but the main *drawback* of simplex is that computation time grows rapidly as number of components n of the variable $x \in \mathbb{R}^n$ increases. In worst case, Lp problem using simplex, requires $2^n - 1$ steps to reach the final solution. Clearly this implies time involved in solving the problem, this is termed as *complexity* of the algorithm. Simplex is said to have *exponential complexity* of order $2^n - 1$ and written as $O(2^n - 1)$.
As n increases, it is better to have *polynomial complexity*. So after simplex comes the *interior point method*, which has polynomial complexity. Here three methods can be mentioned.

- Khachiyan's method
- Karmarkar's method
- Affine Scaling method

Among these methods, Khachiyan's algorithm has no such real world application. So here mainly two algorithms are of importance, *Karmarkar's algorithm* and *Affine Scaling* method. Here Karmarkar's method, which is used in our problem of interest(discussed in next chapter), is discussed only.

3.4.1 Karmarkar's Method

Karmarkar's algorithm is an *interior point* method, and hence it differs from Simplex method. It starts from an initial point, and after certain iterations when it appears that value of objective function is less than certain prescribed value, the iteration stop.
Before applying Karmarkar's algorithm to any problem, it should be in certain form called *Karmarkar's canonical form*.

3.4.1.1 Karmarkar's Canonical Form

Karmarkar's Canonical form is given by,

$$\begin{aligned} \text{miniimize} \quad & c^T x \\ \text{subject to} \quad & Ax = 0 \\ & \sum_{i=1}^{n} x_i = 1 \\ & x \geq 0 \end{aligned}$$

where $x = [x_1, ..., x_n]^T$.
Let $e = [1, ..., 1]^T$ be vector in $\mathbb{R}^n$ with each component equal to 1, Also let Ω denote *nullspace* of A, i.e.

$$\Omega = \{x \in \mathbb{R}^n : Ax = 0\} \tag{3.13}$$

Also define *simplex* Δ in $\mathbb{R}^n$ by

$$\Delta = \{x \in \mathbb{R}^n : e^T x = 1, x \geq 0\} \tag{3.14}$$

So the *centre* of simplex Δ is $a_0 = \frac{e}{n} = [\frac{1}{n}, ..., \frac{1}{n}]^T$.
Clearly $a_0 \in \Delta$. Hence Karmarkar's canonical form is given by,

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & x \in \Omega \cap \Delta \end{aligned}$$

where constraint set(or feasible set) $\Omega \cap \Delta$ can be represented as,

$$\begin{aligned} \Omega \cap \Delta &= \{x \in \mathbb{R}^n : Ax = 0, e^T x = 1, x \geq 0\} \\ &= \{x \in \mathbb{R}^n : \begin{bmatrix} A \\ e^T \end{bmatrix} x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, x \geq 0\} \end{aligned} \tag{3.15}$$

Karmarkar's Restricted Problem

While solving LP problem using Karmarkar's method, following assumptions are used.

- Center a_0 of the simplex Δ belongs to feasible set Ω i.e. $a_0 \in \Omega$,
- Minimum value of the objective function over the feasible set is zero,
- The $(m+1) \times n$ matrix $\begin{bmatrix} A \\ e^T \end{bmatrix}$ has rank $m+1$,
- If the feasible point x satisfies $\frac{c^T x}{c^T a_0} \leq 2^{-q}$ (where q is *termination parameter*), we say the LP problem is solved.

Any lP problem in Karmarkar's canonical form and satisfying above 4 assumptions, is called *Karmarkar's Restricted Problem*.

3.4.1.2 General Form to Karmarkar's Canonical Form

Any general problem of LP can be converted to equivalent Karmarkar's Canonical form, so that solution to one can be used to other. For the discussion consider a linear programming in standard form.

$$\begin{aligned} \text{minimize} \quad & c^T x,\ x \in \mathbb{R}^n \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

The above problem is converted to Karmarkar's canonical form in below steps. Consider $z \in \mathbb{R}^{n+1}$ by,
$z = \begin{bmatrix} x \\ 1 \end{bmatrix}$ Also define $c' = [c^T, 0]^T$ and $A' = A, -b]$. Using above notations above LP an be converted new form,

$$\begin{aligned} \text{minimize} \quad & c^T z,\ z \in \mathbb{R}^{n+1} \\ \text{subject to} \quad & A'z = 0 \\ & z \geq 0 \end{aligned}$$

now it can be noted that above form does not ensure that sum of decision variables are 1. So the following step is done. Let $y = [y_1, ..., y_n, y_{n+1}]^T \in \mathbb{R}^{n+1}$.Where

$$
\begin{aligned}
y_i &= \frac{x_i}{x_1 + ... + x_n + 1}, i = 1(.)n \\
y_{n+1} &= \frac{1}{x_1 + ... + x_n + 1}
\end{aligned}
\tag{3.16}
$$

This transform is *projective transformation*.
So the given LP in standard form is converted to Karmarkar's Canonical form:

$$
\begin{aligned}
\text{minimize} \quad & c'^T y,\ y \in \mathbb{R}^{n+1} \\
\text{subject to} \quad & A'y = 0 \\
& e^T y = 1 \\
& y \geq 0
\end{aligned}
$$

To satisfy assumption 1, following treatment is done. Given point $a = [a_1, ..., a_n]$ is assumed to be *strictly interior* feasible point, i.e. $Aa = b, a > 0$. Let P_+ denote *positive orthant* of $\mathbb{R}^n$, given by $P_+ = \{x \in \mathbb{R}^n : x \geq 0\}$. Let $\Delta = \{x \in \mathbb{R}^{n+1} : e^T x = 1, x \geq 0\}$, be the simplex in $\mathbb{R}^{n+1}$. Consider $T : P_+ \mapsto \Delta$ by

$$
T(x) = [T_1(x), ..., T_{n+1}(x)]^T \tag{3.17}
$$

with

$$
\begin{aligned}
T_i(x) &= \frac{x_i/a_i}{x_1/a_1 + ... + x_n/a_n + 1}, \qquad i = 1(.)n \\
T_{n+1}(x) &= \frac{1}{x_1/a_1 + ... + x_n/a_n + 1}
\end{aligned}
\tag{3.18}
$$

Map T is called *projective transformation* of the positive orthant p_+ into the simplex Δ. We can find a vector $c' \in \mathbb{R}^{n+1}$ and a matrix $A' \in \mathbb{R}^{m \times (n+1)}$ such that for each $x \in \mathbb{R}^n$,

$$
\begin{aligned}
c^T x = 0 \iff c'^T T(x) = 0, \\
and \ \ Ax = b \iff A'T(x) = 0,
\end{aligned}
\tag{3.19}
$$

Also for each $x \in \mathbb{R}^n$, we have $e^T T(x) = 1$, i.e. $T(x) \in \Delta$ and for each $x \in \mathbb{R}^n$,

$$
x \geq 0 \iff T(x) \geq 0 \tag{3.20}
$$

Taking this into account, Karmarkar's canonical form is obtained as,

$$\begin{aligned} \text{minimize} \quad & c'^T y \\ \text{subject to} \quad & A'y = 0 \\ & e^T y = 1 \\ & y \geq 0 \end{aligned}$$

where $a_0 = T(a)$ is the center of the simplex Δ. In the next analysis, point a is explicitly available.
Consider a LP problem of the form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \geq 0 \\ & x \geq 0 \end{aligned}$$

and consider the dual problem

$$\begin{aligned} \text{minimize} \quad & \lambda^T b \\ \text{subject to} \quad & \lambda^T A \leq c^T \\ & \lambda \geq 0 \end{aligned}$$

Now combining primal and dual problem,

$$\begin{aligned} & c^T x - b^T \lambda = 0 \\ & Ax \geq b \\ & A^T \lambda \leq c \\ & x \geq 0 \\ & \lambda \geq 0 \end{aligned} \tag{3.21}$$

Adding *surplus* and *slack* variables to these equations we get,

$$\begin{aligned} & c^T x - b^T \lambda = 0 \\ & Ax - v = b \\ & A^T \lambda + u = c \\ & x, \lambda, u, v \geq 0 \end{aligned} \tag{3.22}$$

Now let $x_0 \in \mathbb{R}^n, \lambda_0 \in \mathbb{R}^m, u_0 \in \mathbb{R}^n, v_0 \in \mathbb{R}^m$ be chosen such that all >0(for example $x_0 = [1, ..., 1]^T$).Now consider the following problem,

$$
\begin{aligned}
\text{minimize} \quad & z \\
\text{subject to} \quad & c^T x - b^T \lambda + (-c^T x_0 + b^T \lambda_0) z = 0 \\
& Ax - v + (b - Ax_0 - v_0) z = b \\
& A^T \lambda + u + (c - A^T \lambda_0 - u_0) z = c \\
& x, \lambda, u, v \geq 0
\end{aligned}
$$

This problem is termed as *Karmarkar's Artificial Problem*, which can be represented in matrix form,

$$
\begin{aligned}
\text{minimize} \quad & \tilde{c}^T \tilde{x} \\
\text{subject to} \quad & \tilde{A}\tilde{x} = \tilde{b} \\
& \tilde{x} \geq 0,
\end{aligned}
$$

where

$$
\begin{aligned}
\tilde{x} &= [x^T, \lambda^T, u^T, v^T, z]^T \\
\tilde{c} &= [0^T{}_{2m+2n}, 1]^T \\
\tilde{A} &= \begin{bmatrix} c^T & -b^T & 0_n{}^T & 0_m{}^T & (-c^T x_0 + b^T \lambda_0) \\ A & 0_{m\times m} & 0_{m\times n} & -I_m & (b - Ax_0 + v_0) \\ 0_{n\times n} & A^T & I_n & 0_{n\times m} & (c - A^T \lambda_0 - u_0) \end{bmatrix} \\
\tilde{b} &= [0, b^T, c^T]^T
\end{aligned}
\tag{3.23}
$$

The following point is strictly interior point of above problem:

$$
\begin{bmatrix} x \\ \lambda \\ u \\ v \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ \lambda_0 \\ u_0 \\ v_0 \\ 1 \end{bmatrix} \tag{3.24}
$$

Further the minimum value of the objective function for Karmarkar's artificial problem is zero if and only if previous set of relations has a solution, there exists x, λ, u, v satisfying

$$
\begin{aligned}
& c^T x - b^T \lambda = 0 \\
& Ax - v = b \\
& A^T \lambda + u = c \\
& x, \lambda, u, v \geq 0
\end{aligned}
\tag{3.25}
$$

3.4.1.3 karmarkar's Algorithm

So our LP problem in Karmarkar's Canonical form

$$\begin{array}{ll} \text{minimize} & c^T x, \qquad x \in \mathbb{R}^n \\ \text{subject to} & x \in \Omega \cap \Delta \end{array}$$

where $\Omega = \{x \in \mathbb{R}^n : Ax = 0\}$, and $\Delta = \{x \in \mathbb{R}^n : e^T x = 1, x \geq 0\}$. Karmarkar's algorithm solves the above problem in a iterative way, given an point $x^{(0)}$ and parameter q, the algorithm generates the sequence $x^{(1)}, x(2), ..., x^{(n)}$. Karmarkar's algorithm has following steps:

- **Initialize**: Set $k = 0; x^{(0)} = a_0 = e/n$
- **Update**: Set $x^{(k+1)} = \Psi(x^{(k)})$, where Ψ is the update rule.
- **Check stopping criterion**: If condition $c^T x^{(k)}/c^T x^{(0)} \leq 2^{-q}$ is satisfied, then stop it.
- **Iterate**: Set $k := k + 1$, go to 2.

The update rule Ψ is as follows. In first step of algorithm set $x^{(0)} = a_0$,then update rule used as

$$x^{(1)} = x^{(0)} + \alpha d^{(0)}, \tag{3.26}$$

where α is a step size and $d^{(0)}$ is update direction, $\alpha \in (0, 1)$. The update direction is chosen as follows. The direction of maximum rate of decrease of the objective function $-c$. But chosen direction should be such that $x_{(1)}$ lies in the constraint set,

$$\begin{aligned} \Omega \cap \delta = & \quad \{x \in \mathbb{R}^n : Ax = 0, e^T x = 1, x \geq 0\} \\ = & \quad \{x \in \mathbb{R}^n : \begin{bmatrix} A \\ e^T \end{bmatrix} x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, x \geq 0\} \\ = & \quad \{x \in \mathbb{R}^n : B_0 x = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, x \geq 0\} \end{aligned} \tag{3.27}$$

where $B_0 \in \mathbb{R}^{(m+1)\times n}$ is = $\begin{bmatrix} A \\ e^T \end{bmatrix}$. Since $x_{(0)} \in \Omega \cap \Delta$, the vector $d^{(0)}$ should lie in the *null space* of B_0. For that $d^{(0)}$ is chosen to be *projection of* $-c$ *onto nullspace of* B_0.For

that the projection matrix is chosen to be

$$P_0 = I_n - B_0^T(B_0B_0^T)^{-1}B_0 \tag{3.28}$$

Specifically $d^{(0)}$ is chosen $d^{(0)} = -r\hat{c}^{(0)}$, where

$$\hat{c}^{(0)} = \frac{P_0c}{\|P_0c\|} \tag{3.29}$$

and $r = 1/\sqrt{n(n-1)}$, where r is a scalar added to update vector $d^{(0)}$, which the radius of the largest sphere inscribed in simplex Δ.So, the vector $d^{(0)} = r\hat{c}^{(0)}$ points in the direction of the projection of $\hat{c}^{(0)}$ of c onto the nullspace of B_0 and $x^{(1)}$ lies in constraint set $\Omega \cap \Delta$ and infact $x^{(1)}$ is a *strictly interior point*.

In general $x^{(k+1)} = \Psi(x^{(k)})$ updation will not ensure that $x^{(k+1)}$ will be the center of simplex.Therefore this point needs to be transferred to the center, to maintain assumptions. Consider, the diagonal matrix D_k ,

$$D_k = \begin{bmatrix} {x_1}^k & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {x_n}^k \end{bmatrix} \tag{3.30}$$

$x_{(k)}$ is a strictly interior point of Δ for all k, hence D_k is non-singular.So,

$${D_k}^{-1} = \begin{bmatrix} 1/{x_1}^k & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1/{x_n}^k \end{bmatrix} \tag{3.31}$$

Now we use the mapping, $U_k : \Delta \mapsto \Delta$ given by:

$$U_k(x) = {D_k}^{-1}x/e^T{D_k}^{-1}x \tag{3.32}$$

This U_k is used for transformation

$$\tilde{x} = U_k(x) \tag{3.33}$$

It can be noted that $U_k(x^{(k)}) = e/n = a_0$ i.e. $x^{(k)}$ is mapped to the center of simplex. Now, $\tilde{x}^{(k)}$ is used to get the update $\tilde{x}^{(k+1)}$,

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k)} + \alpha d^{(k)} \tag{3.34}$$

To compute $d^{(k)}$, consider original problem in $\tilde{x}$,

$$\begin{array}{ll} \text{minimize} & c^T D_k \tilde{x} \\ \text{subject to} & AD_k\tilde{x} = 0 \\ & \tilde{x} \in \Delta \end{array}$$

Now $B_k = \begin{bmatrix} AD_k \\ e^T \end{bmatrix}$ and $d^{(k)} = -r\hat{c}^{(k)}$ and $\tilde{c}^{(k)}$ is normalized projection of $-(c^T D_k)^T = -D_k c$ on the nullspace of b_k. Here the projection matrix is,

$$P_k = I_n - B_k^T (B_k B_k^T)^{-1} B_k \tag{3.35}$$

vector $\hat{c}^{(k)}$ is given by,

$$\hat{c}^{(k)} = \frac{P_k D_k c}{\|P_k D_k c\|} \tag{3.36}$$

Therefore $d^{(k)}$ is ,

$$d^{(k)} = -r\tilde{c}^{(k)} = -r\frac{P_k D_k c}{\|P_k D_k c\|} \tag{3.37}$$

Hence $\tilde{x}^{(k+1)} = \tilde{x}^{(k)} + \alpha d^{(k)}$, the update equation is obtained.
Final step is to find $x^{(k+1)}$, which is equal to ${U_k}^{-1}(\tilde{x}^{(k+1)})$.
Now from definition of U_k we get, $x = {U_k}^{-1}(\tilde{x}) = D_k\tilde{x}/e^T D_k\tilde{x}$. Hence,

$$x^{(k+1)} = {U_k}^{-1}(\tilde{x}^{(k+1)}) = \frac{D_k\tilde{x}^{(k+1)}}{e^T D_k\tilde{x}^{(k+1)}} \tag{3.38}$$

So the algorithm is summarized below,

Algorithm 3.2: Karmarkar's Algorithm

1 Compute the matrices: $D_k = \begin{bmatrix} {x_1}^k & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {x_n}^k \end{bmatrix}$ and $B_k = \begin{bmatrix} AD_k \\ e^T \end{bmatrix}$

2 Compute the *projection matrix*: $P_k = I_n - B_k^T(B_k B_k^T)^{-1} B_k$

3 Compute the normalized orthogonal projection of c on nullspace of B_k:
$\hat{c}^{(k)} = \frac{P_k D_k c}{\|P_k D_k c\|}$

4 Compute the direction vector: $d^{(k)} = -r\tilde{c}^{(k)}$, where $r = 1/\sqrt{n(n-1)}$

5 Now compute $\tilde{x}^{(k+1)}$ using: $\tilde{x}^{(k+1)} = \tilde{x}^{(k)} + \alpha d^{(k)}$, where $\alpha \in (0, 1)$ is the step size

6 Finally find $x^{(k+1)}$ using inverse transformation ${U_k}^{-1}$:
$x^{(k+1)} = {U_k}^{-1}(\tilde{x}^{(k+1)}) = \frac{D_k \tilde{x}^{(k+1)}}{e^T D_k \tilde{x}^{(k+1)}}$

This concludes discussion on Karmarkar's algorithm. The concepts of LP,mainly Interior Point Method(Karmarkar's algorithm) is used to solve the problem addressed in the next chapter.

Chapter 4

Convex Boundary Design

4.1 Introduction

In this chapter, the first of problem of the project is addressed. The main aim of this problem is to design convex boundary of a given data set. The method discussed for boundary design will be used finally for **Chandrayan-2 Lander data set** .

4.2 Objective

The main goal is to design a boundary between feasible and infeasible region of **Chandrayan-2 lander data set** *aiming zero tolerance for infeasible region*.

4.3 Convex Function

In this section a small introduction is given to *Convex function* which will be used for formulating later in this chapter.

Definition 4.1. *A function is said to be convex if line segment joining any two points on the graph of the function lies above or on the graph.*

From above definition, a function f is said to be convex, if it satisfies the following condition:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \tag{4.1}$$

where x_1 and x_2 are two points in the domain of f, $\lambda \in [0, 1]$

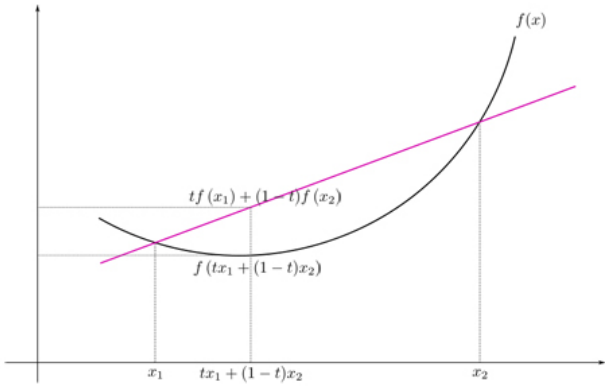

Figure 4.1: Convex Function
[7]

4.3.1 Necessity of Convex Function

4.3.1.1 Function Approach

Our main aim is to find boundary between feasible and infeasible region of Chandrayan-2 lander data set. Now storing all the boundary data points; which are of the form (mass,altitude, downrange); is not efficient method from the point of view of limit in memory size of on-board computer.
So the alternative way is to describe boundary by a function. Here the advantage is we only need to store the coefficients of the terms, which will consume less memory.

4.3.1.2 Choice of Convex function

On-board computer having current mass, altitude, downrange; has to take online decision to land safely, So there must some distinguishing criteria for that purpose. Convex function provides an easy way for that.
From the property of convex function, if a point lies inside the convex region(region bounded by the convex function); and evaluation of the function at that point results in positive quantity, then it can be said that evaluation of the same function for a point outside convex region results in negative quantity and vice-versa. If the point lies on the curve generated by the function, evaluation of the function at that point gives zero.
This property of convex function can be used for our problem. Here if we store coefficients of convex function in on-board computer, then it can easily evaluate the function value using current mass, altitude, downrange values. Now from the sign of the function, we can

easily check whether current point lies inside feasible region or not and consequent decision on landing can be taken.

4.3.2 Problem Approach

The database for the landing problem of **Chandrayan-2** mission,is generated from **Monte Carlo Simulation**. From that data set, feasible boundary points are identified and those points are used for the problem. The approach for our problem is following:

- Aforesaid data contains variation of downrange,altitude,mass.

- Clear distinction(using function) is needed to be made between feasible and infeasible region of the whole data set generated.

- For that purpose a Convex function is used, which can exclude all boundary points at the same time distance of the curve from the boundary data points should be minimized.

Basically we need a curve which will lie above the boundary data points. But there exists many such curves. For example in the following figure, consider that boundary should follow the coordinate axes, we can fit many convex boundaries for that.

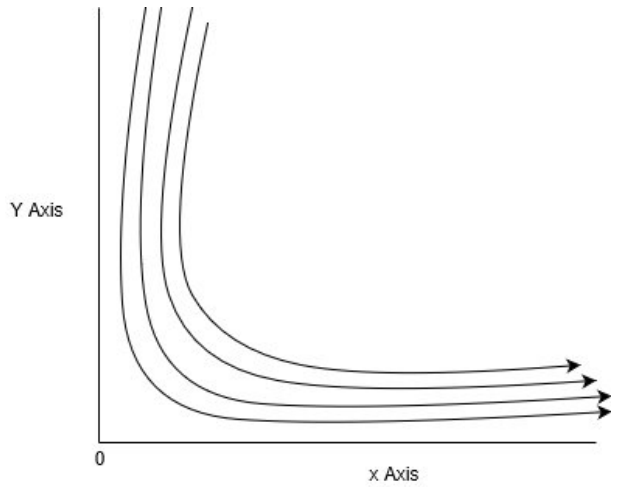

Figure 4.2: Boundary Curves

So we need some condition to choose one of them. And here we consider the curve *having least distance from boundary data points*. The reason behind this is that we can include maximum number of data points of the feasible region, hence avoiding removal of data points from the feasible region. All the above discussion indicates that our boundary design problem boils downs to a constrained optimization problem. We choose convex function of the following form:

$$Z_0 = c_0 + c_1^T X + X^T c_2 X \tag{4.2}$$

where c_0 is scalar, $c_1 \in \mathbb{R}^n$ is a column vector, $c_2 \in \mathbb{R}^{n \times n}$,
n is the size of feature vector X, and c_0, c_1, c_2 are to be found by Optimization.
In our problem feature vector X has dimension 3 (X has three components:mass, altitude and downrange).

4.4 Problem Formulation

The problem can be formulated as follows:
To find a convex function, which has minimum distance from the boundary data points.
In our problem, curve will be designed such that it has minimum distance from boundary data points.
We consider $N \times 1$ dimensional column vector(N is number of data points), whose *each element has magnitude proportional to distance between curve and boundary data points*.
So the problem formulation is:

$$\begin{aligned} &minimize && -\Sigma c_0 + c_1^T X + X^T c_2 X \\ &subject\ to && c_0 + c_1^T X_i + X_i^T c_2 X_i \leq b_i && \text{for all } i = 1(.)N, \text{ each } b_i \geq 0 \end{aligned}$$

where c_0, c_1, c_2 are to be determined through optimization
– This is general formulation of boundary design problem, which can be simplified for our specific problem of *Designing boundary for Chandrayan-2 Lander data set*.
Here our data is 3 dimensional(viz. downrange,altitude, mass). Note our objective function described in the previous section defines conic equation in n dimension space. Same objective function simplifies in 3D space and we can state the problem again in the following form:

$$minimize \qquad -\sum_{x,y,z} a_0 + a_1x + a_2y + a_3z + a_4x^2 + a_5y^2 + a_6z^2 + a_7xy + a_8yz + a_9xz$$
$$subject\ to \qquad AC \leq b \qquad\qquad b \geq 0$$

where $C = [a_0, .., a_9]^T$ coefficients to be determined,

and $A = \begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1^2 & y_1^2 & z_1^2 & x_1y_1 & y_1z_1 & x_1z_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & z_N & x_N^2 & y_N^2 & z_N^2 & x_Ny_N & y_Nz_N & x_Nz_N \end{bmatrix}$,$(x_i, y_i, z_i)$ are data points for all $i = 1(.)N$

–Formulated problem in standard linear programming form and can be solved by standard optimization techniques.

Chapter 5

Simulation Results Analysis

5.1 Introduction

In this chapter, problem of finding boundary curve for **'Chandrayan-2 Lander' data set**, formulated in the previous chapter is addressed. Standard optimization techniques are used to solve the problem. The solutions are then analyzed at the end. Before applying the optimization technique developed, on thee actual data set; we verify it on two sample cases. Finally we apply the technique on 'Chandrayan-2 Lander' dataset.

5.2 Optimization Method

Our formulated problem is in standard linear programming problem format. We can apply linear programming techniques discussed in the previous chapter. As discussed in previous chapters, we can apply either *simplex* or *non-simplex* techniques(also known as *interior point methods*). But interior point method has advantages over simplex, which are:

- Order of Complexity: It has polynomial complexity compared to Simplex with exponential complexity.(complexity explained in 3rd chapter).

- Initial Solution: Simplex requires a starting point to be given. Guessing that initial solution is not possible always.We have to choose a basic, feaible solution as starting point. Now we need to choose m basic column from n columns(where n is number of variables in the constraint equality, after adding surplus or slack variables). The choice can be made $\binom{n}{m}$ ways. If n is sufficiently large, obviously number of available choices will be large, hence choosing initial basic feasible solution is tedious job for simplex. This problem can be avoided if interior point method is used because in-

terior point algorithm starts from an interior point of the feasible region (for example Karmarkar's algorithm has fixed starting point).

Hence interior point method is chosen to solve our optimization problem. We chose **Karmarkar's algorithm** for solving our problem because it has been already established as successful interior point algorithm.

5.3 Performance Evaluation

Karmarkar's algorithm is implemented using MATLAB to solve our optimization problem. Before applying the algorithm to dataset of 'Chandrayan-2 Lander', two sample problems are addressed , where the developed optimization technique is devised and results are analyzed.

5.3.1 Sample Problem 1

Consider a two dimensional problem, **intersection of a straight line and a parabola**, and this curve constitutes sample boundary data set for which curve need to be found . The straight line has equation $y = 30x - 300$, and the parabola has the equation $y = \frac{1}{20}x^2$, where x, y are greater than zero.

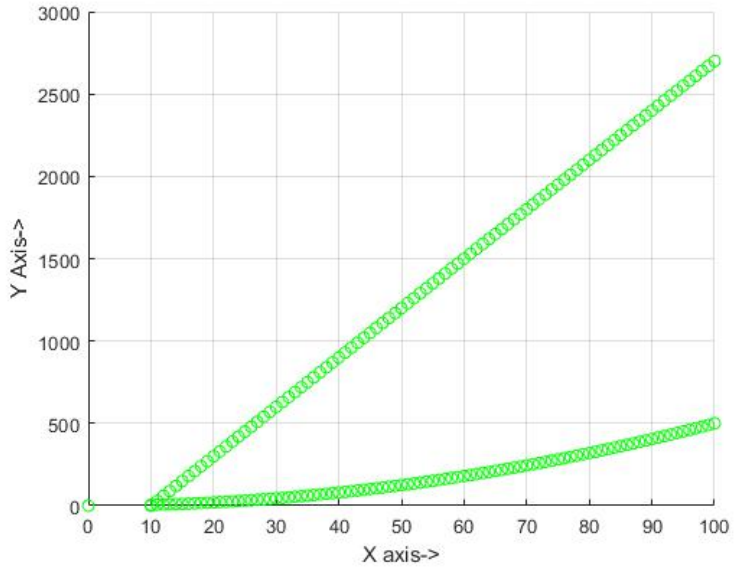

Figure 5.1: 2D Boundary Dataset

5.3.1.1 Result

Result is obtained for the following formulation of optimization problem for above dataset:

$$\begin{aligned} &minimize && \sum_{x,y} a_0 + a_1x + a_2y + a_3x^2 + a_4y^2 + a_5xy \\ &subject\ to && AC \leq 8 \end{aligned}$$

where $C = [a_0, a_1, ..., a_5]^T$ are to be found

and $A = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & y_1^2 & x_1y_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ 1 & x_N & y_N & x_N^2 & y_N^2 & x_Ny_N \end{bmatrix}$

solving above problem in MATLAB using Karmarkar's Algorithm implemented previously, optimized value of the coefficients are obtained:

$C = [-17.33, 1, -0.036, -0.017, -6.729 \times 10^{-5}, 0.0024]^T$,($C$ in normalized form)

And the curve(whose coefficient obtained) plotted against actual dataset:

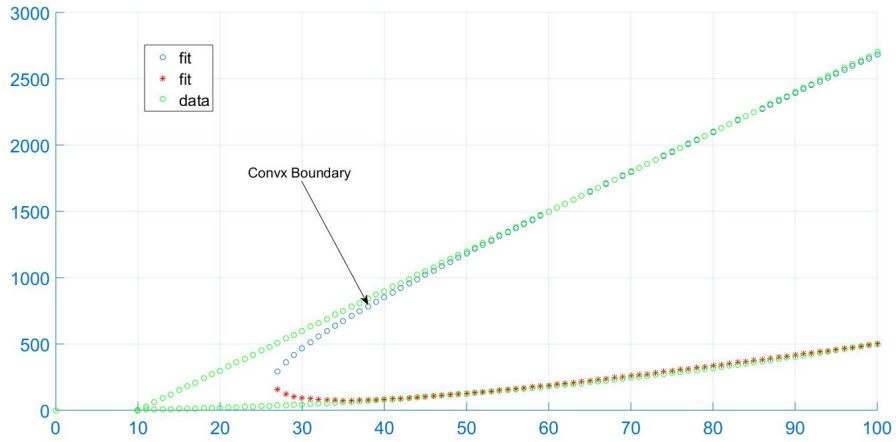

Figure 5.2: 2D Convex Boundary

- It has been considered in the problem that the green points are actual data set and region inside it the *convex zone*.
- The red and blue points denote the convex boundary fitted against the data.

From the figure it is visible that the fitted curve tries to follow the boundary points at the same time lies completely inside the convex zone, hence our objective is achieved.

5.3.2 Sample Problem 2

Here we consider a 3D problem, **intersection of a 3D plane and an elliptic paraboloid**, and this forms the 3D data set for which boundary is to be fit. The equation of 3D plane is:$2x + 2y + 0.08z + 1 = 0$ and the elliptic paraboloid has equation:$z = x^2 + y^2$

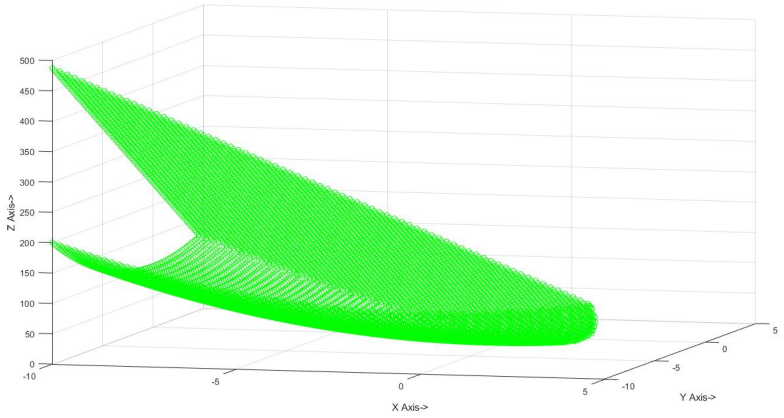

Figure 5.3: 3D Boundary Dataset

5.3.2.1 Result

Here also optimization problem is formulated(with few constants chosen from practical experience) for which results are obtained. The mathematical form of the problem is:

$$minimize \qquad -\sum_{x,y,z} a_0 + a_1x + a_2y + a_3z + a_4x^2 + a_5y^2 + a_6z^2 + a_7xy + a_8yz + a_9xz$$

$$subject\ to \qquad AC \leq 6.02$$

where $C = [a_0, .., a_9]^T$ coefficients to be determined,

and $A = \begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1^2 & y_1^2 & z_1^2 & x_1y_1 & y_1z_1 & x_1z_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & z_N & x_N^2 & y_N^2 & z_N^2 & x_Ny_N & y_Nz_N & x_Nz_N \end{bmatrix}$

While solving this problem using Karmarkar's algorithm developed, it is found that our developed algorithm needs so many add ons to fit on the problem. All such modifications are not possible in the limited time frame for the project. Instead the function **fmincon** is used with interior point method as the underlying algorithm and it is successful to achieve the desired result.

After Solving coefficients obtained are: $C = [-0.606, 1, 0.999, 0.065, 0.297, 0.297, 5.048 \times 10^{-4}, 0.084, 0.021, 0.0206]^T$,

Where C is is normalized form. The curve obtained is plotted against actual data set:

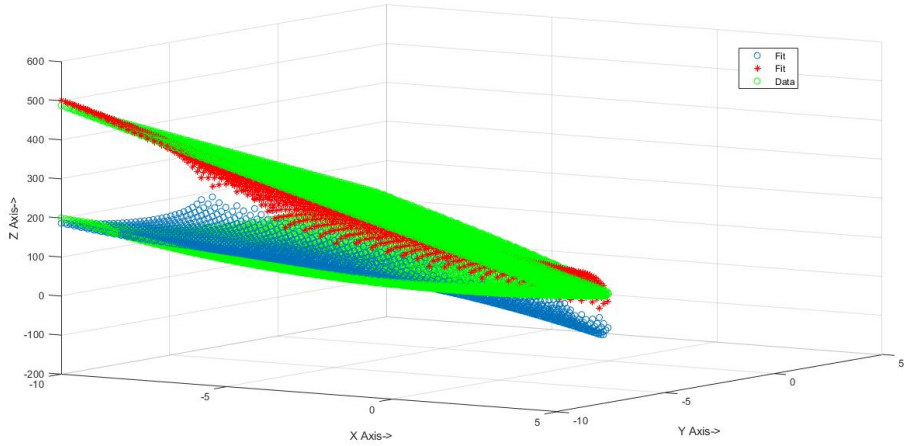

Figure 5.4: 3D Convex Boundary

5.3.3 Convex Boundary Design of Chandrayan-2 Lander

Now we consider, the dataset related to 'Chandrayan-2 Lander'. The given dataset contains only the boundary points between feasible and infeasible region. The goal is to formulate a curve which will follow the boundary datapoints;while lying in convex region and at the same time as close to the boundary datapoints as possible. Above figure shows the boundary points of lander dataset, for which an optimized curve is to be fitted.

5.3.3.1 Result

The mathematical form of this boundary design problem(formulated in previous chapter) is restated here.

$$minimize \qquad -\sum_{x,y,z} a_0 + a_1x + a_2y + a_3z + a_4x^2 + a_5y^2 + a_6z^2 + a_7xy + a_8yz + a_9xz$$

$$suabject\ to \qquad AC \leq 4$$

where $C = [a_0, .., a_9]^T$ coefficients to be determined,

and $A = \begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1^2 & y_1^2 & z_1^2 & x_1y_1 & y_1z_1 & x_1z_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & z_N & x_N^2 & y_N^2 & z_N^2 & x_Ny_N & y_Nz_N & x_Nz_N \end{bmatrix}$,

(x_i, y_i, z_i) are datapoints for all $i = 1(.)N$

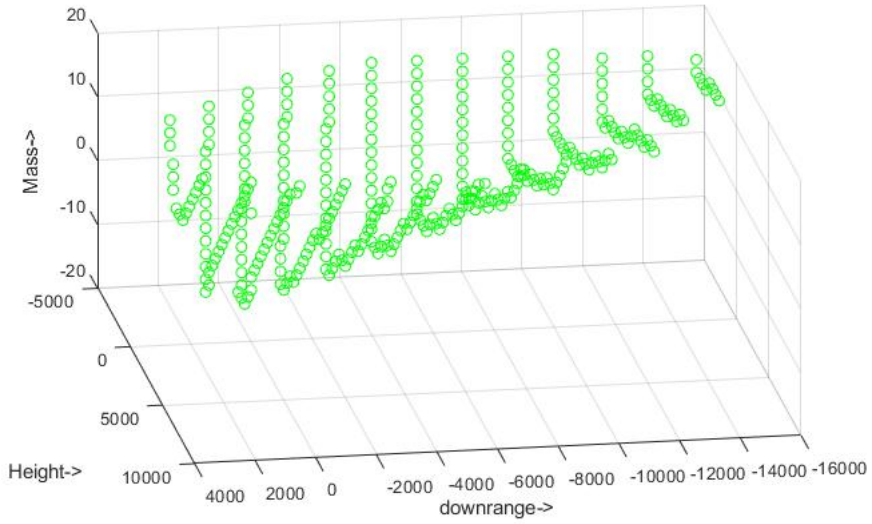

Figure 5.5: 'Chandrayan-2 Lander' Boundary Dataset

Here b vector(distance vector) is chosen to be a column vector of all elements equal to 4, from practical consideration.

While solving the problem formulated above, it is found that Karmarkar's algorithm developed cannot be applied directly, but it needs several modifications to fit on the problem.Instead **fmincon** function is used,which gives us the desired result. After solving the problem, coefficients obtained are given below:

$C = [0.0205, 0.0227, -0.0259, -0.382, 6.30\times10^{-6}, 1.439\times10^{-5}, 1, -9.586\times10^{-6}, -0.006405, 0.0049]$

Apart from solving the problem using optimization, we have also used **'Least Square'** method(unconstrained approach) to find *best fit* curve for given data-set, and both results are compared below:

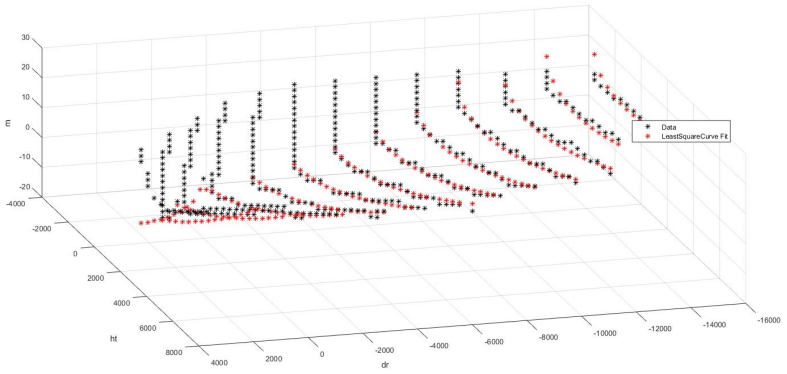

Figure 5.6: 3D Convex Boundary Using Least Square

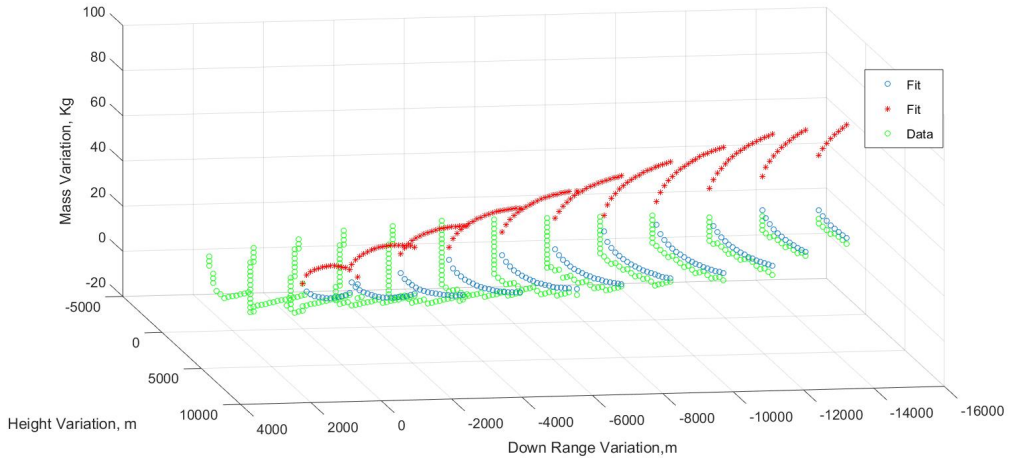

Figure 5.7: 3D Convex Boundary Using Convex Optimization

- From the above plots we can easily observe, that **Least Square fit**, does not exactly in the convex region(feasible region), but it exceeds the convex region(infeasible region).

- Again Convex Boundary designed **interior point** method , fits inside the convex region.Hence , the curve formulated by *constrained optimization* describes the boundary dataset in the best way.
  This gives the advantage of storing only 10 coefficients in *On-board Computer*, and saves the limited memory resource from storing all the boundary data points.

Finally from the mission point of view, we need to first evaluate the designed convex function at downrange,altitude ,mass values,which lies inside convex zone and sign of the result to be stored.Let say the sign is posititve. Now the function is evaluated at current downrange,altitude, mass values and sign is noted. If the sign is positive, then current datapoint lies in feasbile region, hence **landing at the predefined location on moon surface from current configuration(downrange,altitude,mass) is possible**.In the other hand if the sign comes out negative , then **landing at the desired location is not possible from current configuration**.Hence other strategy is required to be followed at that situation.

Chapter 6

Gain Design of Controller

6.1 Introduction

Under this problem, gain of a controller is designed using *optimization techniques*. First we consider a simple spacecraft model, followed by design of controller. Initially, a single axis PD controller is designed, and after finding its gain, the gain values are validated in the closed loop model.
In the next part, designed gain values are used for *'three axis attitude control'* of an spacecraft, and it is verified whether desired specifications are maintained for that model of the spacecraft.

In this chapter, relevant concepts required for the addressed problem are discussed. In the next chapter, main problem is addressed.

6.2 Attitude Representation

By designing the controller for the spacecraft, we want to use it for controlling the attitude of spacecraft. Attitude of spacecraft implies its orientation w.r.t a frame. Given the orientation w.r.t one frame, same orientation can be converted to other frame. There are several ways to it.Here main methods are mentioned.

- Direction Cosine Matrix(DCM)
- Euler Angle
- Quaternion

Vector in one frame can be represented by other frame using *transformation matrix*. Following figure shows a vector ω represented in two different frames.

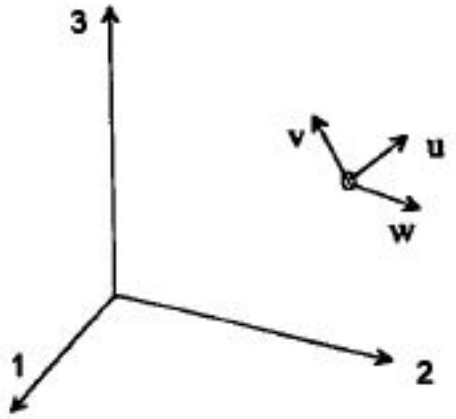

Figure 6.1: Coordinate System of $1, 2, 3$ inertial frame and u, v, w body frame [3]

6.2.1 Direction Cosine Matrix

In the figure, axes $1, 2, 3$ are orthogonal unit vectors defining a right handed triad and the triad is chosen as inertial reference frame.Another orthogonal triad attached to the center of mass is considered namely u, v, w, this is the body frame. Then we can define a matrix A as following,

$$[A] = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \tag{6.1}$$

In the matrix u_1, u_2, u_3 are components of unit vector u along three axes of the reference frame,$u = [u_1, u_2, u_3]^T$. Similarly w, v 's components are there in the matrix.The matrix A is called *direction cosine matrix* or the *attitude matrix*, having the property of mapping vector form one frame to other.
Now consider vector a has components $a = [a_1, a_2, a_3]^T$ in the reference frame. Then it can be shown,

$$[A]a = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} u.a \\ v.a \\ w.a \end{bmatrix} \begin{bmatrix} a_u \\ a_v \\ a_w \end{bmatrix} = a_B \tag{6.2}$$

a_B is the vector a mapped in body frame.

6.2.2 Euler Angle Rotation

Euler angle rotation is defined as successive angular rotation about three orthogonal frame axes.Suppose three orthogonal axes of the body frame be i, j, k, and for the reference frame be I, J, K. There are different combination by which rotation can be performed.Two distinct types of rotations are defined.

- Successive rotation about each of the three axes i, j, k There are six such combinations:1-2-3,1-3-2,2-1-3,2-3-1,3-1-2,3-2-1.(1 means x axis,2 means y axis, 3 means z axis)

- First and third rotation about same axis and second rotation about remaining of the two axes. Again there are six such cases: 1-2-1,1-3-1,2-1-2,2-3-2,3-1-3 and 3-2-3.

Commonly *roll angle*(ϕ) is defined by rotation about body axis i, *pitch angle*(θ) is defined by rotation about body axis j, and *yaw angle*(ψ) is defined about body axis k.
Rotation about each axis can be defined a DCM. For example rotation about z axis is defined by rotation matrix,

$$[A_\psi] = \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

(where c- means cos- and s- means sin-) Now if we consider a particular sequence rotation e.g. 1-2-3, the rotation matrices about all the matrices will be multiplied.

$$[A_{123}] = [A_\psi][A_\theta][A_\phi] = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi + s\psi c\phi & -c\psi s\theta c\phi + s\psi s\phi \\ -s\psi c\theta & -s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi + c\psi s\phi \\ s\theta & -c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{6.4}$$

6.2.3 Unit Quaternions

According to *Euler's Rotation Theorem*, in three-dimensional space, any displacement of a rigid body such that a point on the rigid body remains fixed, is equivalent to a single rotation about some axis that runs through the fixed point.

The corresponding axis is known as *Euler axis* and the angle of rotation is *Euler angle*. This concept is used in formulation of the concept of rotation using quaternion. For rotation using quaternion, principle eigen-vector ($\hat{e}$) is taken as axis of rotation,and quaternion is represented as,

$$q = \begin{bmatrix} e_1 \sin\theta/2 \\ e_2 \sin\theta/2 \\ e_3 \sin\theta/2 \\ \cos\theta/2 \end{bmatrix} \tag{6.5}$$

where $\hat{e} = [e_1, e_2, e_3]^T$ is axis of rotation First three parts constitutes the vector component of quaternion and the last component is the scalar component.
Quarternion is defined in the following way,

$$q = q_4 + iq_1 + jq_2 + kq_3 \tag{6.6}$$

where unit vector i, j, k satisfies following equalities:

$$i^2 = j^2 = k^2 = -1, ij = -ji = k, jk = -kj = i, ki = -ik = j. \tag{6.7}$$

if q is a unit quaternion, then it satisfies the following relation:

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{6.8}$$

6.2.3.1 Few Operations on Quaternion

In our problem few operations are done on quaternion, those operations are briefly mentioned here.

Conjugate of Quaternion If $q = q_4 + iq_1 + jq_2 + kq_3$ is a given quaternion, then its conjugate is denoted by,

$$q^* = q_4 - iq_1 - jq_2 - kq_3 \tag{6.9}$$

Norm of Quaternion Norm of a quaternion is defined by,

$$\|q\| = \sqrt{qq^*} = \sqrt{q^*q} = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} \tag{6.10}$$

Inverse of Quaternion Inverse of a quaternion is defined by,

$$q^{-1} = \frac{q^*}{\|q\|^2} \tag{6.11}$$

Quaternion Multiplication Quaternion multiplication is different from normal multiplication in the sense that quaternion multiplication is not commutative,so if q, p are two quaternions, then $q \otimes p \neq p \otimes q$ definition of quaternion multiplication is,

$$q \otimes p = \begin{bmatrix} q_4 p_{1:3} + p_4 q_{1:3} + q_{1:3} \times p_{1:3} \\ q_4 p_4 - q_{1:3}^T p_{1:3} \end{bmatrix} \tag{6.12}$$

above equation using matrix can be written as,

$$q \otimes p = Q(q)p = \bar{Q}(p)q \tag{6.13}$$

where

$$Q(q) = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \tag{6.14}$$

and $\bar{Q}$ is given by,

$$\bar{Q}(q) = \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \tag{6.15}$$

6.2.3.2 Error quaternions

In a loop, while propagating difference of say reference quaternion(q_r) and measured quaternion(q_m), simple subtraction of two quaternions will not work, instead we have to propagate *error quaternion*(q_e), which is defined by:

$$q_e = q_r \otimes q_m^{-1} \tag{6.16}$$

6.2.4 Quaternion Rate

It is a vector containing time derivative of unit qauternion. Quaternion rate($\dot{q}$) is related to angular velocity(ω_B) (derivation at the appendix):

$$\dot{q}_B^I = \frac{1}{2} q_B^I \otimes \begin{bmatrix} \omega_B^I \\ 0 \end{bmatrix} \tag{6.17}$$

$$\dot{q}_I^B = \frac{1}{2} q_I^B \otimes \begin{bmatrix} \omega_B^B \\ 0 \end{bmatrix} \tag{6.18}$$

,
where B implies body frame and I implies inertial frame of reference.

Among the three ways of attitude representation, DCM has nine components to compute,making it computationally expensive.
Euler angle method has 3 components to compute but it has mainly two flaws:
(1) Expression for attitude determination are trigonometric in nature,hence calculations are not simple.
(2)Using Euler angle may result in *gimbal lock* problem.

This concludes necessary background required for design of gain design of a PD controller, and validation of that gain.

6.2.5 Equations of Motion of Spacecraft

The gain values designed above is used for *three axis attitude control of Spacecraft*.Here the satellite model is discussed briefly. The model has two parts: **kinematics** and **dynamics**.

6.2.5.1 Kinematics

Kinematics of the spacecraft can be explained in terms of quaternions.Satellite's attitude is explained in terms of quaternions(due to the benefits explained in above section). We consider quaternion rate equation to explain the kinematics.

$$\dot{q}_{\omega}(q,\omega) = \frac{1}{2} q \otimes \begin{bmatrix} \omega \\ 0 \end{bmatrix} \tag{6.19}$$

This equation relates quaternion rate with body rate of the satellite. This equation after numerical integration gives the attitude(in terms of quaternion) of the satellite.

6.2.5.2 Dynamics

We describe satellite dynamics by rigid body dynamics. We use Euler-Moment equation to describe satellite dynamics, which states: Net external torque acting on the satellite is equal to rate of change of angular momentum.
Consider, net external torque acting on the satellite be $\tau_{external}$. Then from Euler-Moment equation:

$$\tau_{external} = \dot{h_I} = \dot{h_B} + \omega \times h \tag{6.20}$$

where term with subscript 'I' denotes derivative in inertial frame, and term with subscript 'B' denotes derivative in body frame.(The proof of this equation is given in the appendix)
This equations are non linear and numerical solutions are obtained for these equations.
In our context, external torque is equal to the control torque provided by the controller to the satellite in closed loop dynamics.
Putting $h = I\omega$ in the above equation, where I is *moment of inertia tensor*(I, ω are considered in body frame), we solve numerically for ω, which is used to calculate the quaternion of satellite.
This completes the analysis of dynamics of satellite, which will be considered in next chapter.

6.2.6 Control Design

Satellite's attitude is important for many purposes.For example Solar panels should be facing the sun, for generation of power or antenna should be pointing towards Earth in case of communication satellites.So, proper control is required to achieve exact attitude. Before that attitude should be represented w.r.t. proper reference frame. After that only comes the *Attitude Control System*(ACS).There are many ways to control the attitude of the satellite. Few popular methods are mentioned.

- Gravity Gradient Stabilized Control
- Spin Stabilized Control
- Three Axis Control

Among these we consider three axis control due to the advantage of **fine attitude control**. Here control torque along each axis is achieved by combinations of *momentum wheels(MW), reaction wheels(RW),control moment gyros(CMG),magnetic torquers*.Attitude control system containing reaction wheels/momentum wheels / control moment gyros, is called *momentum storage/exchange systems*.

Chapter 7

Gain Design & Validation

7.1 Introduction

In this chapter gain of PD controller is designed depending on the given constraints using optimization techniques. Validation of the designed gain is done in two phases. In phase I, a double integrator is taken as plant model, and control loop is formed and performance is analyzed. In phase II, plant model is taken with underlying equation as *Euler-moment equation*, and same gain values are used to control that model and corresponding performance is evaluated.

7.2 Nominal Design of Controller

The trivial design of a controller includes *'trial error'* method, this is also called as *'nominal design'* of a controller.
To explain the method, consider a plant model be $\frac{1}{Js^2}$, i.e. a double integrator, and a PD controller is used to control the plant(Consider J to be moment of inertia).Below figure shows the model considered.

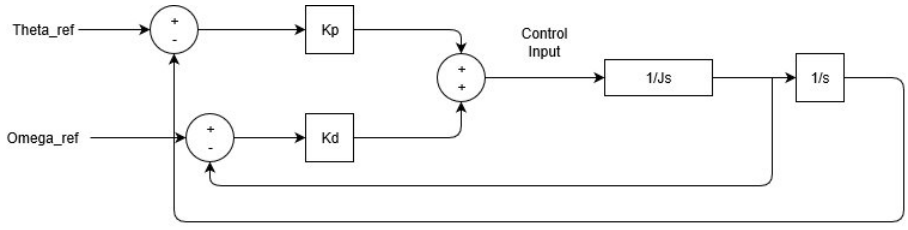

Figure 7.1: Block Diagram of Closed Loop System with PD Controller

So it is clear that the closed loop transfer function of the system is,

$$G(s) = \frac{y(s)}{\theta(s)} = \frac{(k_P + k_D s)}{Js^2 + k_D s + k_P} = \frac{1}{J}\frac{(k_P + k_D s)}{s^2 + \frac{k_D}{J}s + \frac{k_P}{J}} \tag{7.1}$$

Comparing denominator of the last form with standard characteristic equation of a *second order system*, i.e.

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{7.2}$$

(where ζ is damping ratio, ω_n natural frequency of oscillation)
We get,

$$k_p = \omega_n^2 J, k_D = 2\zeta\omega_n J, T_s = \frac{4}{\zeta\omega_n} = \frac{8J}{k_D} \tag{7.3}$$

(T_s is the *settling time*)
Now consider range of ζ, T_s are given as design specifications,
Suppose ζ and T_s are chosen, then from $T_s = \frac{4}{\zeta\omega_n} = \frac{8J}{k_D}$ we get value of k_D and ω_n
Putting value of ω_n in the relation $k_p = \omega_n^2 J$, we get value of k_P.
After finding the gain value the next step is to put them in closed loop model and validate the system's response.
It is often found that system's response is not up to the mark,so in the next step again a choice is made on ζ and T_s, and the above process is repeated. repetition continues until desired response from the system is obtained.
This trial-error method is cumbersome, and sometimes it requires amount of labour to get the desired response. So, we require more efficient and logical way to design the gain values, at the same time satisfy given constraints.
It appears that *Gain design problem*, can be taken as Optimization problem,but for that design specifications should be put properly in the standard form of optimization. In the next section, we formulate our problem in terms of optimization problem.

7.3 Problem Formulation and Analysis

In this problem, Controller is taken as PD controller, whose k_p, k_d values are to be found. Under closed loop operation, plant is supposed to maintain value of certain parameters in given range.

The design specification is given below:

$$\begin{aligned} 0.5 < \zeta < 0.8 \\ 0.1 < \omega_n < 0.5 \end{aligned} \tag{7.4}$$

- Now one thing regarding our problem of 'attitude control of satellite' is that torque applied to the satellite should be minimized, now control torque $T_c = k_P\theta + k_D\omega$ (where θ is single axis attitude error, and ω is angular velocity error). So, to minimize T_c we consider equivalent problem of minimizing the sum, $k_P + k_D$.

Using the relations of (7.3), our design constraints further modifies to,

$$\begin{aligned} 0.5 < \frac{k_D}{2\sqrt{k_P J}} < 0.8 \\ 0.1 < \sqrt{\frac{k_P}{J}} < 0.5 \end{aligned} \tag{7.5}$$

Also further constraints which needs to be considered are given below:

$$\begin{aligned} Coasting\ \ rate < 1.5^0/second \\ T_{cmax} < 0.1\ \ N-m \\ \theta_{Sat} < 5^0 \end{aligned} \tag{7.6}$$

Where T_{cmax} is the maximum control torque that can be applied to the satellite, and θ_{Sat} is to saturate the error angle to be applied to the PD controller.
Basically among the last three constraints, first two constraints are related to the hardware, which determines amount of coasting rate and torque which can be provided practically.
The last constraint is provided so that correction of attitude can be made smoothly. If this saturation is not provided and suppose initial error is attitude is quite large, then controller will try to correct the error within single shot. From practical point of view, such correction will cause high load on actuators and can cause hardware failure in some case.
Finally our problem of Gain design boils down to constrained optimization problem as:

$$
\begin{aligned}
\text{minimize} \quad & k_P + k_D \\
\text{Subject to} \quad & 0.5 < \tfrac{k_D}{2\sqrt{k_P J}} < 0.8 \\
& 0.1 < \sqrt{\tfrac{k_P}{J}} < 0.5 \\
& Coasting\ \ rate < 1.5^0/second \\
& T_{cmax} < 0.1\ \ N-m \\
& \theta_{Sat} < 5^0
\end{aligned}
$$

After formulation of the problem, next step is to find optimum gain. values.

7.3.1 Design of Gain Values

After going through the conditions of the formulated problem, it can be seen that the constraints have non linear nature. We use *fmincon* function to solve such non-linear optimization problem.
After solving the optimization problem, the gain values are found to be:

$$k_p = 9,\, k_D = 30,$$

for moment of inertia $J = 100$ kg-m^2
Now putting k_P, k_D values in eqn. 7.3, we get

$$\zeta = .5, \omega_n = .3\ \ rad/sec$$

Hence Settling time expected from the plant under closed loop operation is:

$$T_s = \frac{4}{\zeta \omega_n} = 26.667 seconds \tag{7.7}$$

(Considering 2 percent tolerance band for response)

7.4 Validation

Before checking the performance of the designed controller for *Three Axis Attitude Control*, the performance of the PD controller designed,is verified for single axis attitude control.

7.4.1 Single Axis Control

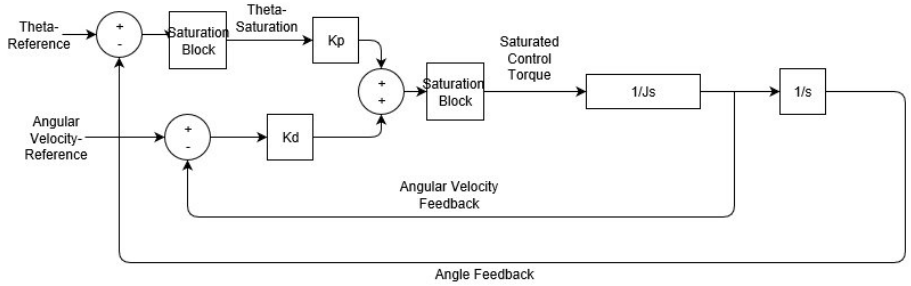

Figure 7.2: Block Diagram of Single Axis Attitude Control

For the above close-loop system, Following design parameters are considered:

Design Parameter	**Value**
Attitude Reference (θ_{Ref})	0^0
Angular Velocity Reference (ω_{Ref})	0 rad/sec
Initial Attitude $(\theta_{initial})$	60^0
Initial Angular Velocity $(\omega_{initial})$	0 rad/s
Saturation Angle (θ_{Sat})	5^0
Maximum Control Torque (T_{cmax})	0.6 N-m
Simulation Time (t_{sim})	150 sec

Table 7.1: Simulation Conditions(single axis control)

7.4.1.1 Simulation Results

The following results are obtained for *Single Axis Attitude Control*.

- First consider the variation of Attitude:

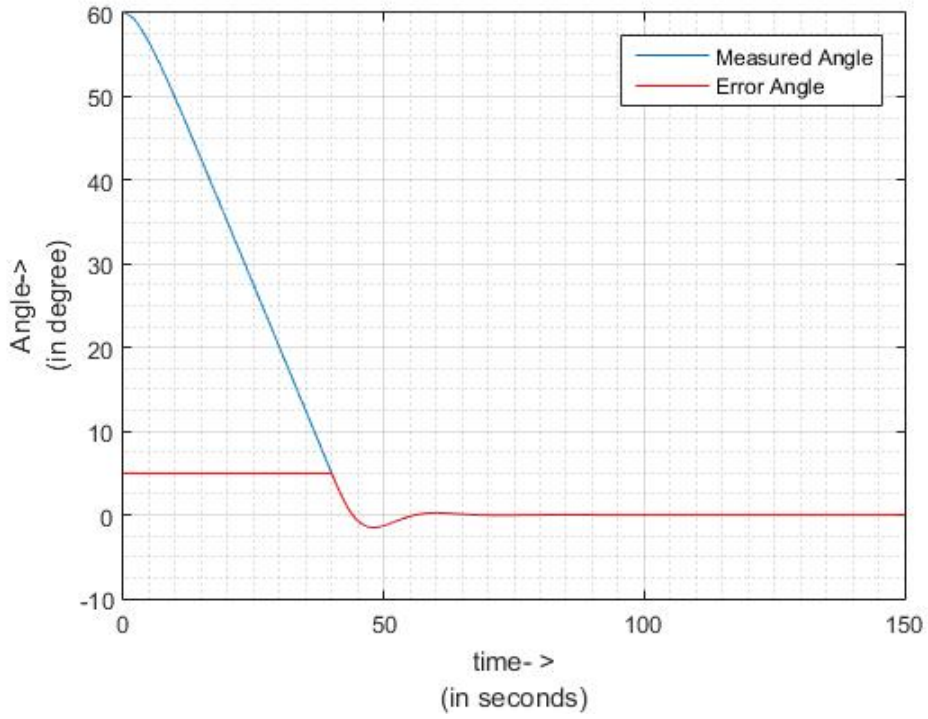

Figure 7.3: Variation of Attitude(along single axis)

□ From the figure it can be seen that, starting from an initial attitude of 60^0, the controller is able to equalize the attitude with reference attitude(0^0). The *settling time* for that is 44.02 sec.

□ Error angle is the difference of Reference angle and Measured angle(at any instant) , which is directly fed to the controller. First, due to application of saturation in attitude, error angle cannot exceed 5^0, and that constant value is applied to the controller. After some instant, when error decreases below 5^0, effect of saturation block is removed, and actual error value is applied to the controller, and finally the error settles down to zero, as expected.

- Next consider the variation of Angular Velocity:

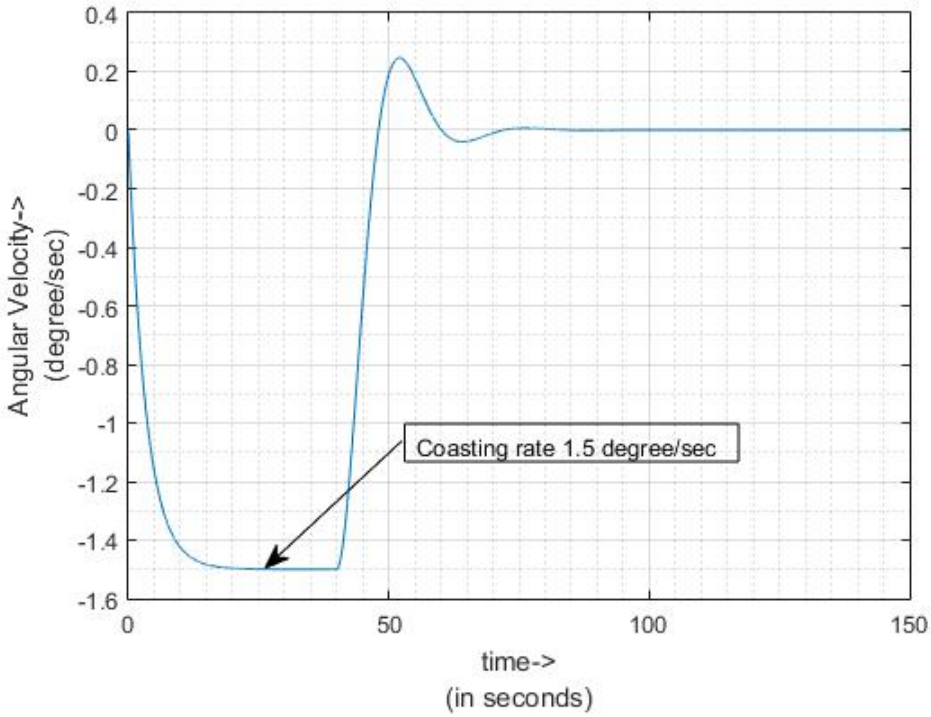

Figure 7.4: Variation of Angular Velocity(along single axis)

□ From the plot it can be seen that initial angular velocity is zero. But due to application of continuous torque(to change the attitude), angular velocity changes continuously. When then satellite enters the *Coasting phase* and applied torque becomes zero, angular velocity becomes constant.

$$\begin{aligned} \tau_{Control} &= k_P\theta + k_D\omega = 0 \\ \implies \omega_{Coasting} &= \omega = -\frac{k_P}{k_D}\theta \end{aligned} \tag{7.8}$$

Putting$\theta = \theta_{Sat}$, maximum achievable Coasting rate is obtained.

$$\begin{aligned} (\omega_{Coasting})_{max} &= -\frac{k_P}{k_D}\theta_{Sat} \\ &= -\frac{9}{30}5^0/sec \\ &= -1.5^0/sec \end{aligned} \tag{7.9}$$

From figure 7.4, it can be seen that angular velocity reaches 1.5^0/sec and remains there for duration of 25.16 second, that is exactly the maximum achievable coasting rate.

- Finally consider the variation of control torque:

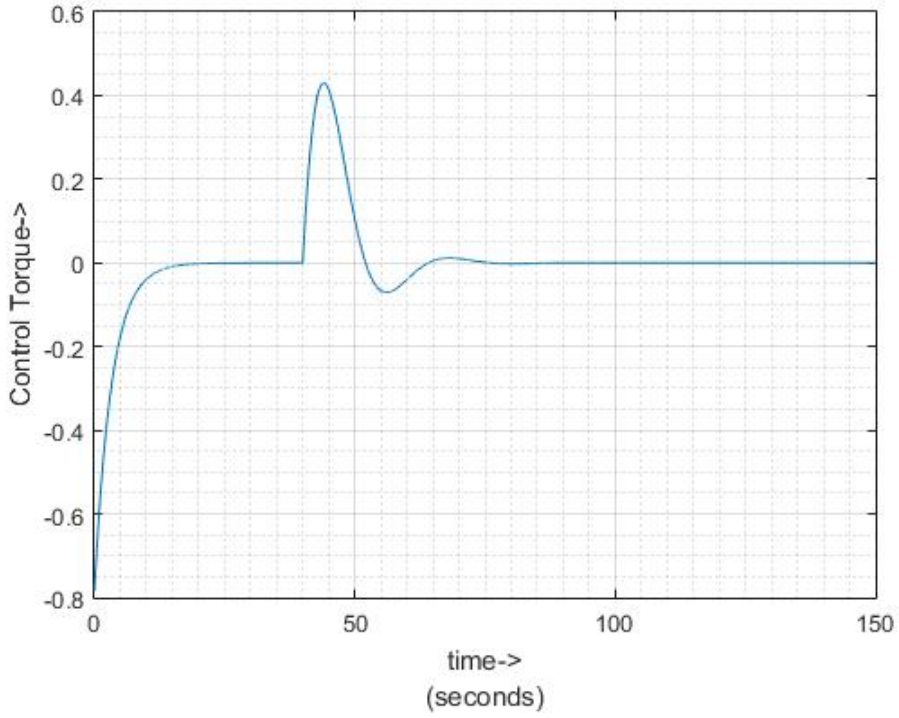

Figure 7.5: Variation of Control Torque(along single axis)

□ From the figure, it can be seen that maximum control torque never exceeds 0.5 N-m,as expected.

□ Also it can be seen that starting from an initial value control torque becomes zero for the duration of 25.16 second(before attaining the positive value), this duration is the Coasting phase, and control torque is zero as expected. Finally, when the error input to the controller becomes zero, the control torque acting on the plant again becomes zero.

7.4.2 Three Axis Control

Now consider the actual model of the spacecraft,defined by the *Euler-moment equation* and *quaternion rate equation*. First consider the block diagram:

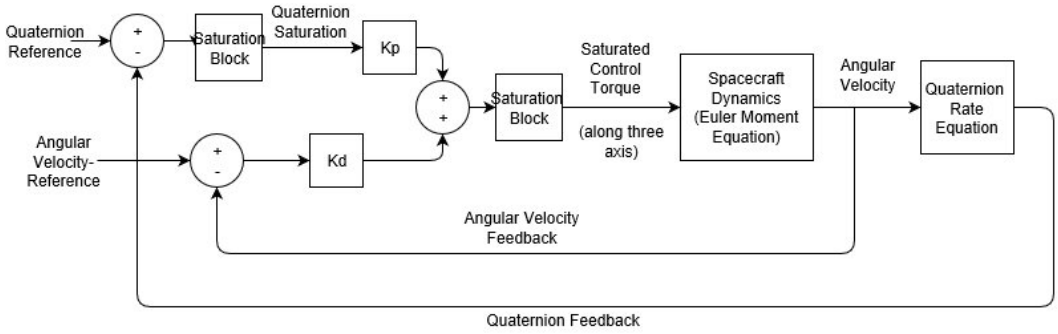

Figure 7.6: Block Diagram of Three Axis Attitude Control

Design Parameter	**Value**
Quaternion Reference $(_{Ref})$	$[1; 0; 0; 0]$
Reference Angular Rate (ω_{Ref})	[0;0;0] rad/sec
Initial Quaternion $(_{initial})$	$[0.7745; 0.1585; 0.5915; 0.1585]$ (corresponding attitude is 60^0 along each axis)
Initial Angular Rate $(\omega_{initial})$	[0.5 ;0.5 ;0.5] deg/sec
Saturation Quaternion $(_{Sat})$	$[0.9972; 0.0416; 0.0454; 0.0416]$ (corresponding to 5^0 along each axis)
Maximum Control Torque (T_{cmax})	0.6 N-m
Moment of Interia(along each axis)	100 kg m^2
Simulation Time (t_{sim})	150 sec

Table 7.2: Simulation Conditions(three axis control)

7.4.2.1 Simulation Results

Finally the gain values designed are used to control attitude of satellite along three axis,following results are obtained. First Consider variation of quaternions:

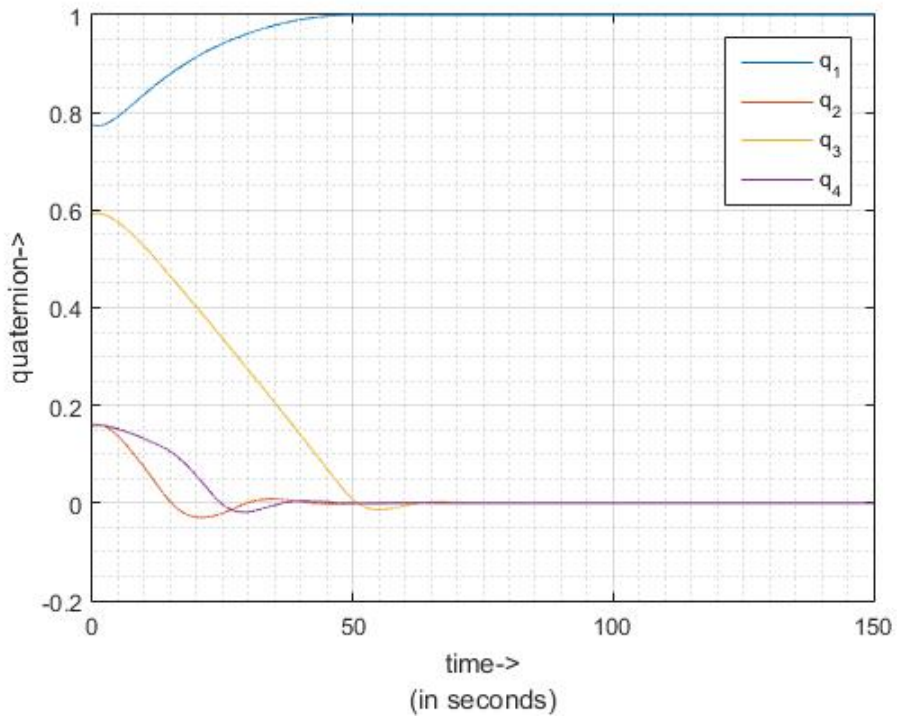

Figure 7.7: Variation of Quaternions

- From the figure it can be seen that starting from initial value of $[0.7745; 0.1585; 0.5915; 0.1585]$, the quaternions settles to $[1; 0; 0; 0]$, which is the reference quaternion vector.

- Overall time for all the quaternions to settle down to 2 percent tolerance band is 50.97 seconds, which is beyond the expected settling time of 26.667 seconds(calculated before).

- Now consider variation of angular rate:

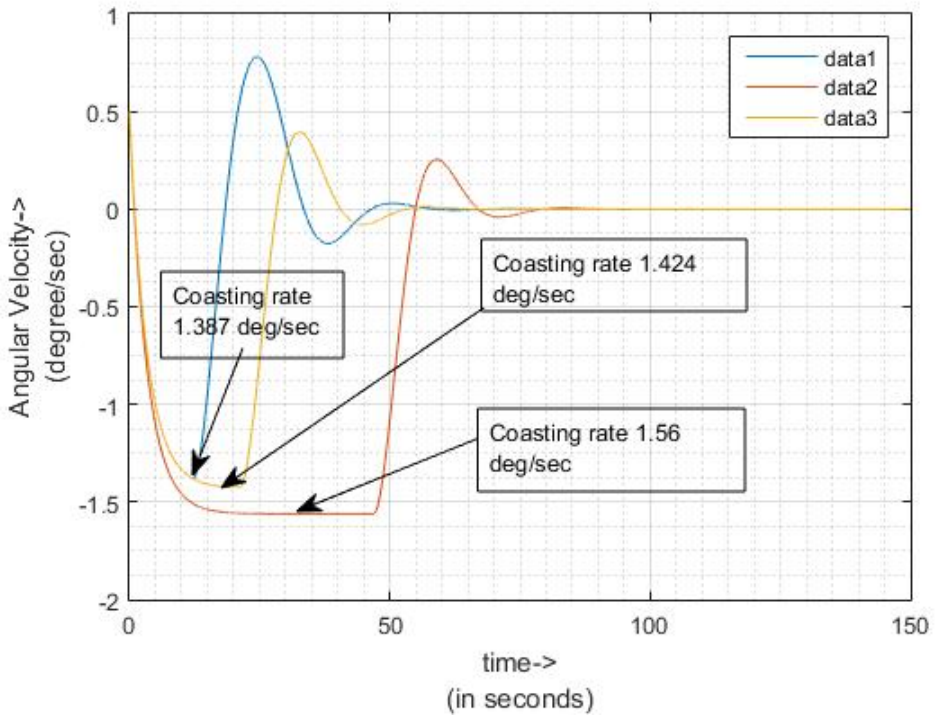

Figure 7.8: Variation of Angular-rates

□ Here we can see that coasting is achieved along three axes at different time with max value of coasting rate along three axes are different, so the settling time.

- Finally consider the variation of control torque:

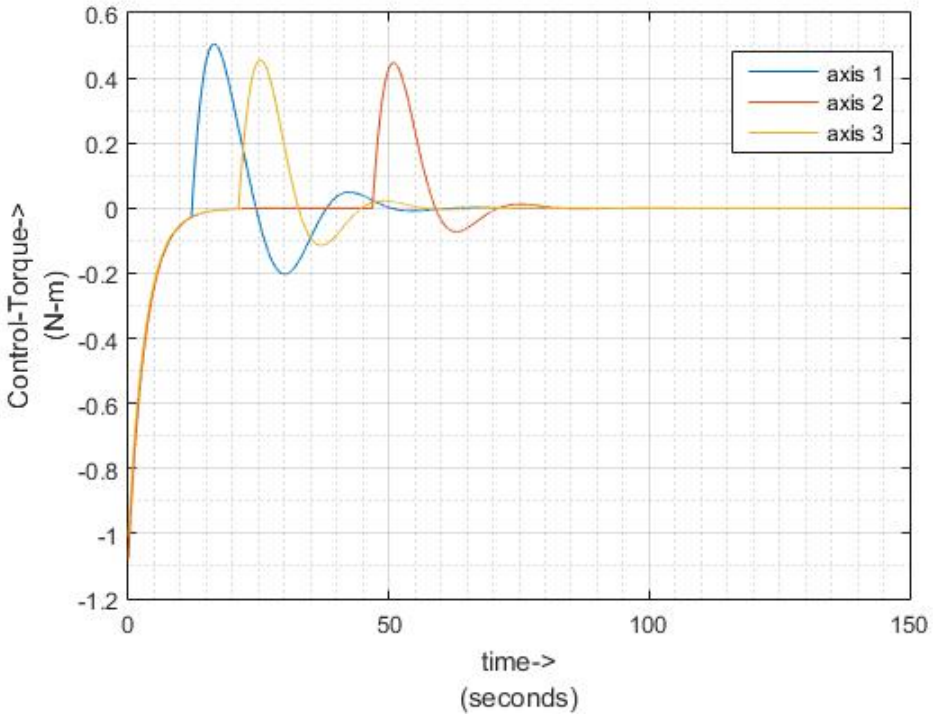

Figure 7.9: Variation of Control-torque(along three axes)

☐ The plot of control torque along three axes again shows that coasting is achieved along three axes as different time.Finally control torque becomes zero as desired attitudes are achieved along three axes.

Chapter 8

Conclusions & Recommendations

In this project two problems are addressed using Convex Optimization techniques. In the first problem, Convex function is designed using constrained optimization techniques. Curve of that closely matches with the boundary data points of **'Chandrayan-2 Lander'**. The optimization method provides an better fit for the boundary points as compare to the *least square method*, which fails to follow the datapoints inside the convex region. One problem can be noticed from the plot of convex boundary. There are few distinct regions in the boundary data set, where no trace of convex function is present to follow the datapoints. In other words there is no real solution for those regions.
So one modification which can be proposed over the existing technique is that, separately optimize for each of the distinct regions. Hence we get convex function for each region. Now least square technique can be used to construct single convex function from those set of convex function. This is an experimental concept which has not been implemented due to lack of time, and this concept can be taken as starting point for future work.
Another modification to our existing method which can be made is that, the convex boundary is designed using *interior point method*.But there are other Convex Optimization algorithms, which are not implemented due to lack of time. In future work, other algorithms can also be used so that comparative study can be made on performance of them and choice can be made for the best performing algorithm.

In the second problem, Convex Optimization is used to design the gain values of PD controller. Constraints for the problem is chosen considering practical constraints on satellite. The gain values are then used for attitude control of satellite along three axis. The simulation results, which describes the controller's performance under closed loop condition, is satisfactory. The results matches with the desired response at the same time maintaining all the constraints. But one problem which is not apparently visible, is the upper

limit on maximum control torque(T_{cmax}). For our design,the value for T_{cmax} is chosen .5 N-m, which is quite large from satellites' operation point of view. The choice of that value is made because of a technical issue. For maximum control torque below .5 N-m, there appears no optimum solution for gain values(when other constraints are kept as such). Now control torque is related to hardware(depending on gyros specifications). Hence no experiments(vary the value of the constraint and check for existence of solution) can be with the value of T_{cmax}. The solution of the problem is to loose limits on constraints which are not hardware related and then use the optimization method to find gain values. This can be implemented in future .

Also for *'Three Axis Attitude Control'* of satellite, only the satellite model is considered under closed loop function. In future, wheel dynamics(reaction wheels, control moment gyros) can also be incorporated with that. Also effect of disturbances like solar radiation pressure, gravity effect due to geoid shape of Earth etc are not considered in the problem. But they have significant impact on satellite's attitude in practical case. All these can be modeled in equation and solved for attitude controller design for three axis stabilization. These will be the leading point for future progress on this problem.

Bibliography

[1] E. K. Chong and S. H. Zak, *An introduction to optimization*. John Wiley & Sons, 2013, vol. 76.

[2] S. Saxena, "A simple introduction to karmarkar's algorithm for linear programming," *arXiv preprint arXiv:1712.08328*, 2017.

[3] M. J. Sidi, *Spacecraft dynamics and control: a practical engineering approach*. Cambridge university press, 2000, vol. 7.

[4] J. R. Wertz, *Spacecraft attitude determination and control*. Springer Science & Business Media, 2012, vol. 73.

[5] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[6] "Normal-vector-of-plane-passing-through-point," [Accessed on 10-07-2018]. [Online]. Available: https://www.researchgate.net/profile/Yashavant_Patel/publication/269710905/figure/fig1/AS:392054246002699@1470484424732/Normal-vector-of-plane-passing-through-point.png

[7] "Convex function," [Accessed on 10-07-2018]. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c7/ConvexFunction.svg/1200px-ConvexFunction.svg.png

Appendix A

MATLAB Source Code

A.1 'Chandrayan-2 Lander' Convex Boundary Design

A.1.1 Karmarkar's Method

Karmarkar's Aritificial Problem

```
% clc;
% clear all;
% close all;
% A=[1 0 ;0 1;1 1];
% b=[4 6 8]';
% c=[-2 -5]';
function [A_,b_,c_,x_]= artificial(A,b,c)
[m,n]=size(A);
xo=ones(1,n)';
lo=ones(1,m)';
uo=ones(1,n)';
vo=ones(1,m)';
c_=[zeros(2*m+2*n,1)',1]';
A_=[c' -b' zeros(n,1)' zeros(m,1)' (-c'*xo+b'*lo); A zeros(m,m)
    zeros(m,n) -eye(m) (b-A*xo+vo);zeros(n,n) A' eye(n) zeros(n,m) (c
    -A'*lo-uo)];


b_ =[0 b' c']';
x_=[xo' lo' uo' vo' 1]';
end
```

Karmarkar's Canonical form

```
% clc;
% clear all;
% close all;
function [Ac,cc,y]=canonical(A,b,c,x,ao)
 c=c-min(c);
% A=[1 0 ;0 1;1 1];
% b=[4 6 8]';
% c=[-2 -5]';
% a=[1 1]';
[m,n]=size(A);
Ac=zeros(m,n+1);
for i=1:n
    Ac(:,i)=A(:,i)*ao(i);
end
Ac(:,n+1)=-b;
cc=zeros(1,n+1)';
for i=1:n
    cc(i)=c(i)*ao(i);
end
cc(n+1)=0;
y=zeros(1,n+1);
 sum=x'*ao+1;
 for i=1:n
    y(i)=x(i)/(sum*ao(i));
 end
 y(n+1)=1/sum;
 y=y';
end
```

Karmarkar's Algorithm

```
%% Karmarkar's algorithm to solve problem
%% Problem should be minimization problem of kind Ax >=b
    **********************(convert accordingly{)
function [X_opt,f,Z]=karma_opt(A,b,c)
[M,N]=size(A);
[A_,b_,c_,x_]= artificial(A,b,c);
ao=x_;
n=length(ao);
q=8;
[Ac,cc,y]=canonical(A_,b_,c_,x_,ao);
[xk,z]=karmarkar(Ac,cc,y,q);
```

```
for i=1:n
    X(i)=ao(i)*xk(i)/xk(n+1);
end
   X_opt=X(1:N)';
   f=c'*X_opt;

  %% Taking data for plotting
%      z=z(1:N,:);
    [p,q1]=size(z);
    for i=1:q1
        for j=1:n
         Z(j,i)=ao(j)*z(j,i)/z(n+1,i);
        end
    end
     Z=Z(1:N,:);
end
```

A.1.2 Problem Analysis

Boundary Design For Intersection of Line and Parabola

```
clc;
clear all;
close all;

%% Problem Formulation
M=100;
x=0:M;
y1=30*x-300;
y=.05*x.^2;
a=[zeros(1,10)' zeros(1,10)';x(11:101)' y(11:101)';x(11:101)' y1
    (11:101)'];


%% Optimization
X=a(:,1);
Y=a(:,2);
N=length(X);

A= [ones(N,1) X Y X.^2 Y.^2 X.*Y;-eye(6)];
B= [10*ones(N,1);zeros(6,1)];
```

```
Cost=[N sum(X) sum(Y) sum(X.^2) sum(Y.^2) sum(X.*Y)]';
AM = [ones(N,1) X Y X.^2 Y.^2 X.*Y];
BM = [10*ones(N,1)];
CM = [N sum(X) sum(Y) sum(X.^2) sum(Y.^2) sum(X.*Y)]';
%[X_opt,f,Z]=karma_opt(A,B,Cost);
X_opt=fmincon(@(C)(CM(1)*C(1)+CM(2)*C(2)+CM(3)*C(3)+CM(4)*C(4)+CM(5)*
    C(5)+CM(6)*C(6)),ones(6,1)/6,AM,BM,[],[],-inf*(ones(6,1)),inf*(
    ones(6,1)),@detm);

Z=X_opt;
Z=Z./max(Z);
c=Z(1);
g=Z(2)/2;
f=Z(3)/2;
a=Z(4);
b=Z(5);
h=Z(6)/2;
a*b-h^2
for i=1:length(X)
    D=[Z(1) Z(2)*X(i) Z(3) Z(4)*X(i)^2 Z(5) Z(6)*X(i)];
    c1=(D(1)+D(2)+D(4));
    b1=D(3)+D(6);
    a1=D(5);
    Y1(i)=(-b1+sqrt(b1^2-4*a1*c1))/(2*a1);
    Y2(i)=(-b1-sqrt(b1^2-4*a1*c1))/(2*a1);
end

 p=[];
q=[];
l=[];
w=[];
for i=1:N
if(imag(Y1(i))==0)
w=[w Y1(i)];
l=[l X(i)];
end
end

for i=1:N
if(imag(Y2(i))==0)
q=[q Y2(i)];
p=[p X(i)];
```

```
end
end
scatter(p,q)
hold on;
scatter(l,w,'r*')
hold on;
scatter(X,Y,'g')
x = [-1000:1000];
y = [-1000:1000];
for i=1:length(x)
for j=1:length(y)
z(i,j) = Z(1) + Z(2)*x(i) + Z(3)*y(j) + Z(4)*x(i)^2 + Z(5)*y(j)^2 + Z
    (6)*x(i)*y(j);
end
end
figure,mesh(y,x,z)
```

Boundary Design for Intersection of 3D Plane and Elliptic Paraboloid

```
clc;
clear all;
close all;
%% Problem formulation
a1=2;
b1=2;
c1=0.08;
d=1;
z=@(x,y)(x.^2+y.^2);
x=linspace(-10,10,100);
y=linspace(-10,10,100);
[xx,yy]=meshgrid(x,y);
surf(xx,yy,z(xx,yy));
shading interp;
hold on;
point=[xx(:),yy(:)];
Z_=point(:,1).^2+point(:,2).^2;
%
z1=@(x1,y1)(-b1*y1-a1*x1-d)/c1;
x1=linspace(-10,10,100);
y1=linspace(-10,10,100);
[xx1,yy1]=meshgrid(x1,y1);
surf(xx1,yy1,z1(xx1,yy1));
shading interp;
```

```
point1=[xx1(:),yy1(:)];
Z11=(-b1*point1(:,2)-a1*point1(:,1)-d)/c1;


vec1=[point Z_];%eliptic paraboloid
vec2=[point1 Z11]; %plane
len=length(vec1);
p_=[];
q_=[];

for i=1:len
    t= a1*vec1(i,1)+b1*vec1(i,2)+c1*vec1(i,3)+d;
    if(t<=0)
        p_=[p_; vec1(i,1) vec1(i,2) vec1(i,3)];
    end
end

for i=1:len
    t1=vec2(i,3)-vec2(i,2)^2-vec2(i,1)^2;
    if(t1>=0)
        q_=[q_; vec2(i,1) vec2(i,2) vec2(i,3)];
    end
end

cascade=[p_;q_];
X=cascade(:,1);
Y=cascade(:,2);
Z=cascade(:,3);

N=length(X);

%% Optimization
AM=-[ones(N,1) X Y Z X.^2 Y.^2 Z.^2 X.*Y Y.*Z Z.*X];
BM=[6*ones(N,1)];
CM =[N sum(X) sum(Y) sum(Z) sum(X.^2) sum(Y.^2) sum(Z.^2) sum(X.*Y)
     sum(Y.*Z) sum(X.*Z)]';

% [X_opt,f]=karma_opt(AM,BM,CM);
% OPTIONS =optimoptions('fmincon','Algorithm','trust-region-
    reflective');
  X_opt=fmincon(@(C)(CM(1)*C(1)+CM(2)*C(2)+CM(3)*C(3)+CM(4)*C(4)+CM
      (5)*C(5)+CM(6)*C(6)+CM(7)*C(7)+CM(8)*C(8)+CM(9)*C(9)+CM(10)*C
```

```
        (10)),ones(10,1),AM,BM,[],[],-inf*(ones(10,1)),inf*(ones(10,1))
        );
Zo=X_opt;
Zo=Zo./max(Zo);
c=Zo(1);
g=Zo(2)/2;
f=Zo(3)/2;
a=Zo(4);
b=Zo(5);
h=Zo(6)/2;
a*b-h^2
for i=1:length(X)
    D=[Zo(1) Zo(2)*X(i) Zo(3)*Y(i) Zo(4) Zo(5)*X(i)^2 Zo(6)*Y(i)^2
        Zo(7) Zo(8)*X(i)*Y(i) Zo(9)*Y(i) Zo(10)*X(i)];
    c1=(D(1)+D(2)+D(3)+D(6)+D(5)+D(8));
    b1=D(4)+D(9)+D(10);
    a1=D(7);
    Z1(i)=(-b1+sqrt(b1^2-4*a1*c1))/(2*a1);
    Z2(i)=(-b1-sqrt(b1^2-4*a1*c1))/(2*a1);
end


p=[];
q=[];
l=[];
w=[];
r=[];
r1=[];
for i=1:N
if(imag(Z1(i))==0)
w=[w Z1(i)];
l=[l Y(i)];
r=[r X(i)];
end
end
% scatter(l,w)
for i=1:N
if(imag(Z2(i))==0)
q=[q Z2(i)];
p=[p Y(i)];
r1=[r1 X(i)];
end
```

```
 end
 figure;
% scatter(p,q)
scatter3(r1,p,q)
hold on;
scatter3(r,l,w,'r*')
 hold on;
scatter3(X,Y,Z,'g')
```

Boundary Design For 'Chandrayan-2 Lander' Dataset

```
 clear all
close all

%% Loading Chandrayan-2 Lander Dataset and Processing
 a = load('DataOut.FBMMM_580');
wi=1;ni=1;
mean = [ zeros(length(a),1) zeros(length(a),1)  zeros(length(a),1)
    ...
          zeros(length(a),1)  zeros(length(a),1)  zeros(length(a),1)
              ...
          zeros(length(a),1)  zeros(length(a),1)  -516000*ones(length(
              a),1) ...
          19000*ones(length(a),1)  zeros(length(a),1)  zeros(length(a)
              ,1) ...
          zeros(length(a),1)  zeros(length(a),1)  760*ones(length(a)
              ,1) ...
          1100*ones(length(a),1)  zeros(length(a),1) zeros(length(a)
              ,3)];

b = a+0*mean;

magic_number = 37;
itr = 1;
for i=1:length(b)/magic_number

    cv(i) = sum(b((i-1)*37+1:i*37,17));
    if (cv(i) == 0)
        ncvds(i,:) = b((i-1)*37+1,[9 10 16 17]);
    else
        okp  = b((i-1)*37+1:i*37,17);
        for k = 1:magic_number
            if (okp(k) ~= 0)
```

```
                cvds(itr,:) = b((i-1)*37+k,[9 10 16 17]);
                itr = itr+1;
            end
        end
    end

end
figure,grid on,hold on,
plot3(ncvds(:,1),ncvds(:,2),ncvds(:,3),'y*')
plot3(cvds(:,1),cvds(:,2),cvds(:,3),'k*')


%% Group Each dataset corresponding to given downrange

drset = cvds(:,1);
drsetdiff = drset(2:end)-drset(1:end-1);

itr = 1;
for i = 1:length(drsetdiff)
    if (drsetdiff(i) ~= 0)
        partition(itr) = i;
        itr = itr+1;
    end
end

%% Get a group data for a given downrange

% dataset = cvds(partition(7)+1:partition(8),:);
% figure,grid on,hold on,
% plot(dataset(:,2),dataset(:,3),'k*')
bpts = zeros(1,3);
for i = 2:length(partition)
    if (i == 1)
        dataset = cvds(1:partition(i),:);
    else
        dataset = cvds(partition(i-1)+1:partition(i),:);
    end
    pts = getSurface(dataset);
    for k = 1:length(pts)
        bpts(end+1,:) = pts(k,:);
    end
end
```

```
figure,grid on,hold on,
%plot3(cvds(:,1),cvds(:,2),cvds(:,3),'g*')
plot3(bpts(:,1),bpts(:,2),bpts(:,3),'k*');xlabel('dr');ylabel('ht');
    zlabel('m')

itr = 1;
for i=1:length(bpts)
    if (bpts(i,3) < 20)
        bptsUp(itr,:) = bpts(i,:);itr = itr +1;
    end
end

figure(27),grid on,hold on,
% plot3(cvds(:,1),cvds(:,2),cvds(:,3),'g*')
plot3(bptsUp(:,1),bptsUp(:,2),bptsUp(:,3),'k*');xlabel('dr');ylabel('
    ht');zlabel('m')

%% Least Square Method For Boundry Design
fMat = [ones(length(bptsUp),1) bptsUp(:,1) bptsUp(:,2) bptsUp(:,3)
    bptsUp(:,1).^2 bptsUp(:,2).^2 bptsUp(:,3).^2 bptsUp(:,1).*bptsUp
    (:,2) bptsUp(:,2).*bptsUp(:,3) bptsUp(:,3).*bptsUp(:,1)];
[U S V] = svd(fMat);
coeff = V(:,end)';

err = fMat*coeff';
figure,plot(err)

ac  = coeff(7);
bc  = coeff(4) + coeff(9)*bptsUp(:,2) + coeff(10)*bptsUp(:,1);
cc  = coeff(1) + coeff(2)*bptsUp(:,1) + coeff(3)*bptsUp(:,2) + coeff
    (5)*bptsUp(:,1).^2 + coeff(6)*bptsUp(:,2).^2 + coeff(8)*bptsUp
    (:,1).*bptsUp(:,2);

r1 = (-bc + sqrt(bc.^2 - 4*ac*cc))/2/ac;
r2 = (-bc - sqrt(bc.^2 - 4*ac*cc))/2/ac;

%% Constrained Optimization Method For Boundary Design
%figure(27),hold on,grid on,plot3(bptsUp(:,1),bptsUp(:,2),r1,'g*')
%bptsUp=load('datadeb');
% figure(27),hold on,grid on,plot3(bptsUp(:,1),bptsUp(:,2),real(r2),'
```

```
    r*')

X=bptsUp(:,1);
Y=bptsUp(:,2);
Z=bptsUp(:,3);
N=length(X);

AM=-[ones(N,1) X Y Z X.^2 Y.^2 Z.^2 X.*Y Y.*Z Z.*X];
BM=[4*ones(N,1)];
CM =[N sum(X) sum(Y) sum(Z) sum(X.^2) sum(Y.^2) sum(Z.^2) sum(X.*Y)
     sum(Y.*Z) sum(X.*Z)]';

% [X_opt,f]=karma_opt(AM,BM,CM);
% OPTIONS =optimoptions('fmincon','Algorithm','trust-region-
    reflective');
X_opt=fmincon(@(C)(CM(1)*C(1)+CM(2)*C(2)+CM(3)*C(3)+CM(4)*C(4)+CM(5)*
    C(5)+CM(6)*C(6)+CM(7)*C(7)+CM(8)*C(8)+CM(9)*C(9)+CM(10)*C(10)),
    ones(10,1),AM,BM,[],[],-inf*(ones(10,1)),inf*(ones(10,1)),@det3m)
    ;

Zo=X_opt;
Zo=Zo./max(Zo);
c=Zo(1);
g=Zo(2)/2;
f=Zo(3)/2;
a=Zo(4);
b=Zo(5);
h=Zo(6)/2;

a*b-h^2
for i=1:length(X)
    D=[Zo(1) Zo(2)*X(i) Zo(3)*Y(i) Zo(4) Zo(5)*X(i)^2 Zo(6)*Y(i)^2
        Zo(7) Zo(8)*X(i)*Y(i) Zo(9)*Y(i) Zo(10)*X(i)];
    c1=(D(1)+D(2)+D(3)+D(6)+D(5)+D(8));
    b1=D(4)+D(9)+D(10);
    a1=D(7);
    Z1(i)=(-b1+sqrt(b1^2-4*a1*c1))/(2*a1);
    Z2(i)=(-b1-sqrt(b1^2-4*a1*c1))/(2*a1);
end
p=[];
q=[];
l=[];
```

```
w=[];
r=[];
r1=[];
for i=1:N
if(imag(Z1(i))==0)
w=[w Z1(i)];
l=[l Y(i)];
r=[r X(i)];
end
end
% scatter(l,w)
for i=1:N
if(imag(Z2(i))==0)
q=[q Z2(i)];
p=[p Y(i)];
r1=[r1 X(i)];
end
 end
 figure;
scatter(p,q)
scatter3(r1,p,q)
hold on;
scatter3(r,l,w,'r*')
 hold on;
scatter3(X,Y,Z,'g')
```

A.2 PD Controller Gain Design and Validation

A.2.1 Gain Design

Gain Design Constraints Function

```
function [c,ceq]=nonconst(y,zeta,wn,Ts,J)
%    J=100;
theSat = [5];
wSat = 1.5*pi/180;
  TcMax = 4/sqrt(3)*0.05;
%    TcMax = 1.575;
CoastingRate = 1.5;
thei = 1;
```

```
 c=[zeta(1)-(y(2)/(2*sqrt(y(1)*J)));...
  (  y(2)/(2*sqrt(y(1)*J)))-zeta(2);...
  wn(1)-sqrt(y(1)/J);...
  sqrt(y(1)/J)-wn(2);...
  Ts(1)-8*J/y(2);...
  8*J/y(2)-Ts(2);...
  y(1)/y(2)*theSat(thei)-(CoastingRate);...
  -y(1)/y(2)*theSat(thei)+(CoastingRate);...
%   (0.04/y(1)-0.42*pi/180) ; % disturbance torque
%   -(0.04/y(1) +0.39*pi/180) ; % disturbance torque
    (y(1)*theSat(thei)*pi/180+y(2)*wSat)-TcMax   ];

% ceq=[y(1)/y(2)*theSat(thei)-(CoastingRate)];
ceq =[];
end
```

Gain Design Function

```
function [k,flag]=gain(zeta,wn,Ts,J)
%% zeta,wn, Ts are in range and J is moment of inertia of the system
Co=[9,30]';

fun=@(x)myfun(x);
noncoln=@(x)nonconst(x,zeta,wn,Ts,J);
  options = optimoptions(@fmincon,'MaxFunEvals',300000,'MaxIter'
      ,100000);

%  options = optimoptions(@fmincon,'Algorithm','sqp','MaxIter',1500);
[z,~,flag]=fmincon(fun,Co,[],[],[],[],zeros(2,1),inf*ones(2,1),
    noncoln,options );
wn1=sqrt(z(1)/J);
zet1=z(2)/(2*wn1*J);
Ts1=4/(zet1*wn1);
k=[z' wn1 zet1 Ts1];
end
```

Run Function(for evaluation gain at given design specification)

```
function [a,flag]=run()
zeta=[.5  .8];
wn=[.2  .5];
```

```
J=100;
Ts=[0  100];
[a,flag]=gain(zeta,wn,Ts,J);
end
```

A.2.2 Single Axis Attitude Control

Runge-Kutta(RK4) Function

```
function [nxtstate]= rk4dynamics(t,f,x,h,u,params)

    dn=f(t,x,u,params);
    k1=h*dn;
    x1=x+.5*h*k1;
    k2=h*f(t+0.5*h,x1,u,params);
    x2=x +.5*h*k2;
    k3=h*f(t+0.5*h,x2,u,params);
    x3=x+h*k3;
    k4=h*f(t+h,x3,u,params);
    nxtstate= x + (1/6)*(k1+2*k2+2*k3+k4);

end
```

Dynamics Function(Double integrator for single axis control)

```
function [wdot]= dynamics(t,w,Tau,params)
%       Im=params.I;
%       I1=inv(Im);
%       cr=-cross(w,Im*w);
%       wdot=I1*cr;
Im=params.J;
    I1=inv(Im);
    cr=-cross(w,Im*w);
    wdot=I1*(Tau+cr);

end
```

Code of Single Axis Attitude Control

```
clc;
clear;
close all;
%%%% Physical Parameters
params.J = 100;
```

```
%%%%% Simulation Parameters
starttime = 0;
stoptime = 150;
dStp = 0.01;
N = stoptime/dStp;
thetar=0;
wr=0;
thetaSat=5*pi/180;
wSat=10*pi/180;
%TcMax = 4/sqrt(3)*0.05;
TcMax =1.575;
kp=9;
kd=30;

%%%%%% Initial Conditions
xi=[60*pi/180 0]';
t=0;
x=xi;
xs=[];
time=[];
torques=[];
thetas=[];

%%%% Adding saturation and Dynamics simulation
for i=1:N
    thetaer=thetar-x(1);
    wer=wr-x(2);
    if(abs(thetaer)>thetaSat)
        theta=sign(thetaer)*thetaSat;
    else
        theta=thetaer;
    end
    thetas = [thetas theta];
    w = wer;
    Tc=kp*theta+kd*w; %***********Control torque
    if(abs(Tc)>TcMax)
        Torque=sign(Tc)*TcMax;
    else
        Torque=Tc;
    end
    handle=@dynamics;
```

```
        xn= rk4dynamics(t,handle,x,dStp,Torque,params);
        x=xn;
        xs=[xs x*180/pi];
        time=[time t];
        torques=[torques Torque];
        t=t+dStp;
    end
    figure ;
    plot(time,xs(1,:)),hold on,plot(time,-thetas*180/pi,'r')
    grid on;
    grid minor;
    figure;
    plot(time,xs(2,:))
    grid on;
    grid minor;
    figure;
    plot(time,torques)
    grid on;
    grid minor;
```

A.2.3 Three Axis Attitude Control

Quaternion Rate Function

```
    function [der]=quat_der1(t,q,w,params)
    p=qmult(q,w);
    der=.5*p;
    end
```

Spacecraft Dynamics Function

```
    function [wdot]= dynamics(t,w,Tau,params)
       Im=params.J;
        I1=inv(Im);
        cr=-cross(w,Im*w);
        wdot=I1*(Tau+cr);

    end
```

Code of Three Axis Attitude Control

```
    clc
    clear all
```

```
close all
%%%%%%%%%% Code for three axis stabilzation
%% Physical Parameters
params.J = [100 0 0; 0 100 0; 0 0 100];

%%%%% Simulation Parameters
starttime = 0;
stoptime = 150;
dStp = 0.01;
N = stoptime/dStp;
qr=[1 0 0 0]';
wr=[0 0 0]';
% qSat=ones(3,1)*.5;
qSat=[0.9972     0.0416     0.0454     0.0416]';
% qSat = 0.05*ones(4,1);
%wSat=ones(3,1)*1.5*pi/180;
kp=9;
kd=30;
Tcmax=1.575*ones(3,1);

%% Initial Conditions
% qi=0*ones(4,1);
qi=[0.7745     0.1585     0.5915     0.1585]';
wi=[0.5 0.5 0.5]'*pi/180;
t=0;
% x=xi;
q=qnorm(qi);
w=wi;
xs1=[];
xs2=[];
time=[];
qd=zeros(4,1);
wd=zeros(3,1);
count=1;
% Q=[quat(theta)];
torque=zeros(3,1);
T=[];%torque array

%% Error Calculation
% quati=eul2quat(xi');
for i=1:N
    qer1=qmult(qr,qconj(q));
```

```
    qer=qnorm(qer1);
     wer=wr-w;
%% Control Torque Formulation
    for j=2:4
       if(abs(qer(j))>qSat(j))
           qd(j)=sign(qer(j))*qSat(j);
        else
            qd(j)=qer(j);
        end
    end
    qd(1)=sqrt(1-(q(1)^2+q(2)^2+q(3)^2+q(4)^2));
    Tc=2*kp*qd(2:4)+kd*wer;% torque equation

    for j=1:3
        if(abs(Tc(j))>Tcmax(j))
            torque(j)=sign(Tc(j))*Tcmax(j);
        else
            torque(j)=Tc(j);
        end
    end
    T=[T torque];

    %% Updating parameters
    handle1=@dynamics;
     xs1=[xs1 w];
    Xn= rk4dynamics(t,handle1,w,dStp,torque,params);
    w=(Xn);
%   handle2=@qprod;
%   quater=(quat(theta));
    handle2=@quat_der1;
    count
    count=count+1;
    W=[0;w];
    xs2=[xs2 q];
   Yn= rk4dynamics(t,handle2,q,dStp,W,params); % **** quatmultiply
       inputs are m*4 type
   q=qnorm(Yn);
%  theta=(quat2eul(Yn))';
%  Q=[Q Yn];
%  theta=eul(Yn);
   time=[time t];
   t=t+dStp;
```

```
end
figure ;
plot(time,xs1*180/pi);grid on;
xlabel('time'); ylabel('w');
grid minor;
figure ;
plot(time,xs2);grid on;
xlabel('time'); ylabel('quaternion');
grid minor;

figure ;
plot(time,T);grid on;
xlabel('time'); ylabel('Control-Torque');
grid minor
```

s

Appendix B

Derivations

B.1 Relation of Rotational Velocity & Quaternion Rate

We know, that *unit quaternions* are quaternions with unit norm. Thorugh out the analysis, consider q be unit quaternion,i.e. $\|q\| = 1$

Unit quaternion can be used to represent the attitude of a rigid body. Consider $\vec{x}$ be a constant vector in fixed reference frame and $\vec{x'}$ be same vector in rotating reference.Let us denote,$z = \begin{bmatrix} \vec{x} \\ 0 \end{bmatrix}$ and $z' = \begin{bmatrix} \vec{x'} \\ 0 \end{bmatrix}$ Then we have

$$\begin{aligned} z' &= \bar{q} \otimes z \otimes q \\ z &= q \otimes z' \otimes \bar{q} \end{aligned} \tag{B.1}$$

Where $\bar{q}$ represents *conjugate* of q, and as q is also unit norm, hence $\bar{q}$ is same as q^{-1}(inverse quaternion).
So applying time derivative to z, and using the relation $\dot{z'} = 0$(where dot represents time derivative),

$$\begin{aligned} \dot{z} &= \dot{q} \otimes z' \otimes \bar{q} + q \otimes z' \otimes \dot{\bar{q}} \\ \implies \dot{z} &= \dot{q} \otimes \bar{q} \otimes z \otimes q \otimes \bar{q} + q \otimes \bar{q} \otimes z \otimes q \otimes \dot{\bar{q}} \\ (using\ z'\ value\ from\ B.1) & \\ \implies \dot{z} &= \dot{q} \otimes \bar{q} \otimes z + z \otimes q \otimes \dot{\bar{q}} \\ As\ we\ know\ q \otimes \bar{q} &= I(unity) \end{aligned} \tag{B.2}$$

Now consider the term $\dot{q} \otimes \bar{q}$,(q is of the form,$q = (q_4, \vec{q})$)

$$\dot{q}\bar{q} = (-\dot{q_4}\vec{q} + q_4\dot{\vec{q}} - \dot{\vec{q}} \times \vec{q}, \dot{q_4}q_4 + \dot{\vec{q}}.\vec{q})$$

Using quaternion multiplication formula
Now consider the scalar part of $\dot{q}\bar{q}$, which is equals to $\dot{q_4}q_4 + \dot{q_1}q_1 + \dot{q_2}q_2 + \dot{q_3}q_3$.
Now,

$$\dot{q_4}q_4 + \dot{q_1}q_1 + \dot{q_2}q_2 + \dot{q_3}q_3 = q \otimes \dot{q} = 0 \tag{B.3}$$

The above relation is derived as follows,

$$\begin{aligned} \|q\| &= 1 (as\ \ q\ \ is\ \ unit\ \ quaternion) \\ \implies q \otimes q &= \|q\|^2 = 1 \\ \implies q \otimes \dot{q} + \dot{q} \otimes q &= 0\ \ (taking\ \ derivative\ \ of\ \ above\ \ equation) \\ \implies 2q \otimes \dot{q} &= 0 \\ \implies q \otimes \dot{q} &= 0 \end{aligned} \tag{B.4}$$

hence we have,

$$\dot{q} \otimes \bar{q} = (\vec{v}, 0) \tag{B.5}$$

where $\vec{v} = -\dot{q_4}\vec{q} + q_4\dot{\vec{q}} - \dot{\vec{q}} \times \vec{q}$

Again from eqn (B.2),

$$q\dot{\otimes}\bar{q} = (\dot{q_4}\vec{q} - q_4\dot{\vec{q}} + \dot{\vec{q}} \times \vec{q}, \dot{q_4}q_4 + \dot{\vec{q}}.\vec{q})$$

Using quaternion multiplication formula. which implies,

$$q\dot{\bar{q}} = (-\vec{v}, 0) \tag{B.6}$$

So the eqn (B.2) can be written as,

$$\begin{aligned} \dot{z} &= (\vec{v}, 0) \otimes z + z \otimes (-\vec{v}, 0) \\ \implies \dot{\vec{x}} &= 2(\vec{v} \times \vec{x}) \end{aligned} \tag{B.7}$$

Also ,

$$\begin{aligned}
\|\dot{z}\| &= \sqrt{\|(\vec{v},0)\otimes z\|^2 + \|z\otimes(-\vec{v},0)\|^2 + 2\|(\vec{v},0)\otimes z\|\|z\otimes(-\vec{v},0)\|} \\
\implies \|\dot{z}\| &= \sqrt{\|(\vec{v},0)\|^2\|z\|^2 + \|z\|^2\|(-\vec{v},0)\|^2 + 2\|(\vec{v},0)\|\|z\|\|z\|\|(-\vec{v},0)\|} \\
&\textit{using property } \|a\otimes b\| = \|a\|\|b\|, \textit{where } a,b \textit{ are quaternions} \\
\implies \|\dot{\vec{x}}\| &= \sqrt{\|\vec{v}\|^2\|\vec{x}\|^2 + \|\vec{x}\|^2\|-\vec{v}\|^2 + 2\|\vec{v}\|\|\vec{x}\|\|\vec{x}\|\|-\vec{v}\|} \\
\implies \|\dot{\vec{x}}\| &= \sqrt{4\|\vec{v}\|^2\|\vec{x}\|^2} \\
\implies \|\dot{\vec{x}}\| &= 2\|\vec{v}\|\|\vec{x}\|
\end{aligned} \tag{B.8}$$

From eqns (B.7) and (B.8), we can conclude that $\vec{v} \perp \vec{x}$
Again if rotation of $\vec{x}$ is considered to be pure rotation with *angular v*elocity ω, then it can be written,

$$\dot{\vec{x}} = \vec{\omega} \times \vec{x},\ \ \vec{\omega} \perp \vec{x} \tag{B.9}$$

So using eqns (B.7), (B.8), (B.9), we can write: $\vec{\omega} = 2\vec{v}$
hence

$$\omega = (\vec{\omega},0) = 2(\vec{v},0) = 2\dot{q}\otimes\bar{q} \tag{B.10}$$

multiplying both sides by, q we get:

$$\begin{aligned}
2\dot{q}\otimes\bar{q}\otimes q &= \omega\otimes q \\
\implies \dot{q} &= \frac{1}{2}\omega\otimes q
\end{aligned} \tag{B.11}$$

now let ω' be angular velocity in body frame,then: $\omega' = \bar{q}\otimes\omega\otimes q$ Putting ω from (B.10) we get,

$$\begin{aligned}
\omega' &= 2\bar{q}\otimes\dot{q} \\
&\textit{multiplying both sides by } q, \\
\implies q\otimes\omega' &= 2\dot{q} \\
\implies \dot{q} &= \frac{1}{2}q\otimes\omega'
\end{aligned} \tag{B.12}$$

Hence we have,

$$\boxed{\dot{q} = \frac{1}{2}\omega \otimes q}, (in\ \ case\ \ of\ \ Inertial\ \ Reference\ \ frame)$$

$$\boxed{\dot{q} = \frac{1}{2} q \otimes \omega'}, (in\ \ case\ \ of\ \ Body\ \ Reference\ \ frame)$$

B.2 Derivation of Euler Moment Equation

Euler moment equation is used to describe the spacecraft dynamics discussed in chapter 7. The equation is given below:

$$\vec{\tau} = \dot{\vec{h_I}} = \dot{\vec{h_B}} + \vec{\omega} \times \vec{h} \tag{B.13}$$

where $\vec{\tau}$ defines net external torque acting which comprises of control torque and disturbance torque in general (if we consider satellite model), $\dot{\vec{h_I}}$ derivative of angular momentum in inertial frame, $\dot{\vec{h_B}}$ is derivative of angular momentum in body frame.
Above equation can be directly derived from the relation of time-derivative of a vector in an inertial frame and non-inertial frame.The relation is discussed and derived below followed by derivation of Euler momet equation from that equation.
Consider $\vec{x}$ be a vector,whose time derivative is to be found.$\vec{x_I}$ denotes the vector represented in inertial frame, $\vec{x_B}$ represents the same vector represented in body frame.

Figure (B.1) shows the two coordinate axes with vector $\vec{x}$ replaced by angular velocity $\vec{\omega}$. Suppose, at any instant ϕ, θ, ψ amount rotation required along X, Y, Z (inertial frame) axes respectively to achieve the body frame orientation.
The rotation can be described by rotation matrix R_I^B(inertial to body), which is given for particular sequence of rotation(X-Y-Z),

$$R_I^B = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi + s\psi c\phi & -c\psi s\theta c\phi + s\psi s\phi \\ -s\psi c\theta & -s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi + c\psi s\phi \\ s\theta & -c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{B.14}$$

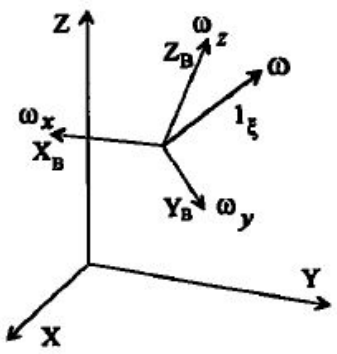

Figure B.1: Coordinate System of X, Y, Z inertial frame and X_B, Y_B, Z_B body frame

(where c- means cosine and s- means sine)
If the rotations along axes are infinitesimal small say $d\phi, d\theta, d\psi$, then the rotation matrix simplifies to:

$$\delta R_I^B = \begin{bmatrix} 1 & \Delta\psi & -\Delta\theta \\ -\Delta\psi & 1 & \Delta\phi \\ \Delta\theta & -\Delta\phi & 1 \end{bmatrix} \tag{B.15}$$

Now at time instant 't', the following relation holds:

$$\begin{aligned} \vec{x_B}(t) &= R_I^B \vec{x_I}(t) \\ \implies \vec{x_I}(t) &= (R_I^B)^T \vec{x_B}(t) \\ \implies \vec{x_I}(t) &= R_B^I \vec{x_B}(t) \end{aligned} \tag{B.16}$$

where $R_B^I = (R_I^B)^{-1} = (R_I^B)^T$,as rotation matrix is *orthogonal*
$\vec{x_B}$ and $\vec{x_I}$ denotes vector $\vec{x}$ in body frame and inertial frame respectively
Simillary we can write,

$$\delta R_B^I = (\delta R_I^B)^{-1} = (\delta R_I^B)^T \tag{B.17}$$

Now consider, in small time Δt rotates by an amount $\Delta\phi, \Delta\theta, \Delta\psi$ about x, Y, Z axes respectively,Hence

$$\vec{x_I}(t + \Delta t) = \delta R_B^I . R_B^I . \vec{x_B}(t + \Delta t) \tag{B.18}$$

Hence change in rotation matrix is given by,

$$
\begin{aligned}
&\Delta R_B^I = \delta R_B^I . R_B^I - R_B^I \\
\implies &\Delta R_B^I = (\delta R_B^I - I) R_B^I \\
&\Delta R_B^I = \begin{bmatrix} 0 & -\Delta\psi & \Delta\theta \\ \Delta\psi & 0 & -\Delta\phi \\ -\Delta\theta & \Delta\phi & 0 \end{bmatrix} . R_B^I (\ using\ eqns\ (B.15), (B.17))
\end{aligned}
\tag{B.19}
$$

Hence dividing above equation by Δt we get,

$$
\frac{\Delta R_B^I}{\Delta t} = \begin{bmatrix} 0 & -\frac{\Delta\psi}{\Delta t} & \frac{\Delta\theta}{\Delta t} \\ \frac{\Delta\psi}{\Delta t} & 0 & -\frac{\Delta\phi}{\Delta t} \\ -\frac{\Delta\theta}{\Delta t} & \frac{\Delta\phi}{\Delta t} & 0 \end{bmatrix} . R_B^I \tag{B.20}
$$

Under limit $\Delta t \mapsto 0$,the above eqn becomes,

$$
\begin{aligned}
&\lim_{\Delta t \to 0} \frac{\Delta R_B^I}{\Delta t} = \lim_{\Delta t \to 0} \begin{bmatrix} 0 & -\frac{\Delta\psi}{\Delta t} & \frac{\Delta\theta}{\Delta t} \\ \frac{\Delta\psi}{\Delta t} & 0 & -\frac{\Delta\phi}{\Delta t} \\ -\frac{\Delta\theta}{\Delta t} & \frac{\Delta\phi}{\Delta t} & 0 \end{bmatrix} . R_B^I \\
\implies &\frac{dR_B^I}{dt} = \begin{bmatrix} 0 & -\frac{d\psi}{dt} & \frac{d\theta}{dt} \\ \frac{d\psi}{dt} & 0 & -\frac{d\phi}{dt} \\ -\frac{d\theta}{dt} & \frac{d\phi}{dt} & 0 \end{bmatrix} . R_B^I
\end{aligned}
\tag{B.21}
$$

Now say $\frac{dR_B^I}{dt} = \dot{R_B^I}$, also consider $\frac{d\phi}{dt} = \omega_X$, $\frac{d\theta}{dt} = \omega_Y$, $\frac{d\psi}{dt} = \omega_Z$,where $\omega_X, \omega_Y, \omega_Z$ are angular velocity of body frame about X, Y, Z axes respectedly.
Hence we have,

$$
\begin{aligned}
&\dot{R_B^I} = \begin{bmatrix} 0 & -\omega_Z & \omega_Y \\ \omega_Z & 0 & -\omega_X \\ -\omega_Y & \omega_X & 0 \end{bmatrix} . R_B^I \\
\implies &\dot{R_B^I} = C(\vec{\omega}) . R_B^I
\end{aligned}
\tag{B.22}
$$

where $C(\omega)$ is the *Cross-product matrix* for angular velocity vector $\vec{\omega} = [\omega_X, \omega_Y, \omega_Z]^T$ and
,

$$C(\vec{\omega}) = \begin{bmatrix} 0 & -\omega_Z & \omega_Y \\ \omega_Z & 0 & -\omega_X \\ -\omega_Y & \omega_X & 0 \end{bmatrix}$$

Now again consider last eqn of (B.16),

$$\begin{aligned} & \vec{x_I}(t) = R_B^I \vec{x_B}(t) \\ & \dot{\vec{x_I}}(t) = R_B^I \dot{\vec{x_B}}(t) + \dot{R_B^I} \vec{x_B}(t) \ (taking\ derivative\ w.r.t.\ time\ on\ both\ sides) \\ \implies & \dot{\vec{x_I}}(t) = R_B^I \dot{\vec{x_B}}(t) + C(\vec{\omega}) R_B^I \vec{x_B}(t) \ (using\ eqn(B.22)) \\ \implies & \dot{\vec{x_I}}(t) = R_B^I \dot{\vec{x_B}}(t) + C(\vec{\omega}) \vec{x_I}(t) \ (using\ last\ eqn\ of\ (B.16)) \end{aligned} \tag{B.23}$$

Now $\dot{\vec{x_I}}(t) = (\frac{d\vec{x}}{dt})_{Inertial}$ (derivative of $\vec{x}$ in inertial frame),
$R_B^I \dot{\vec{x_B}}(t) = (\frac{d\vec{x}}{dt})_{Body}$ (derivative of $\vec{x}$ in body frame),
and $C(\vec{\omega})\vec{x_I}(t) = \vec{\omega} \times \vec{x}$
Then last line of eqn(B.23) can be reformed as,

$$(\frac{d\vec{x}}{dt})_{Inertial} = (\frac{d\vec{x}}{dt})_{Body} + \vec{\omega} \times \vec{x} \tag{B.24}$$

This is the general equation relating derivative of $\vec{x}$ in an inertial and non-inertial frame(rotating with angular velocity $\vec{\omega}$)
Now we consider angular momentum vector of a rigid body be $\vec{h}$. Consider external torque acting on it be $\vec{\tau}$(which is equals to $\vec{\tau_c} + \vec{\tau_d},\ \vec{\tau_c}$ being control torque,$\vec{\tau_d}$ being disturbance torque),then

$$\vec{\tau} = \dot{\vec{h_I}} \ (\vec{h_I}\ is\ \vec{h}\ represented\ in\ inertial\ frame) \tag{B.25}$$

Using eqn(B.24) we have,

$$(\frac{d\vec{h}}{dt})_I = (\frac{d\vec{h}}{dt})_B + \vec{\omega} \times \vec{h} \tag{B.26}$$

Putting (B.26) in (b.25) we get,

$$\vec{\tau} = \dot{\vec{h}}_I = \dot{\vec{h}}_B + \vec{\omega} \times \vec{h} \tag{B.27}$$

Hence we get the famous Euler-Moment equation,

$$\boxed{\vec{\tau} = \dot{\vec{h}}_I = \dot{\vec{h}}_B + \vec{\omega} \times \vec{h}}$$