“MODELLING AND LQR BASED LINEARIZED CONTROL STRATEGY OF QUAD ROTOR AIRCRAFT”

In partial fulfilment of the requirements for the degree of Masters in Mechatronics Engineering

By

Tehseen Akhtar

2010-NUST-MS PhD-MTS-25

Project Supervisor

Lt. Col. Dr. Knuwar Faraz

NUST,College of Electrical& Mechanical Engineering
Department of Mechatronics

Preface

This report is presented by a student from the department of mechatronics at college of electrical and mechanical engineering, NUST. This document contains a detailed report of MS thesis project in relations to Unmanned Arial Vehicle.

There were three semesters on campus and one off campus in the MS program in mechatronics. The purpose of on campus semesters was to give a brief theoretical knowledge to the students about different aspects of mechatronic systems and its applications.

The goal of this project is to derive a viable states space model of Quad-X and design a controller which is able to make the Quad-X hover. This report is divided in to five main chapters: Introduction, System Description, Design and Strategy, Modelling and Controller design. All the references are listed at the end and all the MATLAB code and Quad-X simulation models are all given in the project CD.

The author wishes to give special thanks to all those people who gave their time for the successful completion of the project namely Lt. Col. Dr. Knuwar Faraz and Sir Shafiq.

Contents

List of Figures

Abstract

In this paper, a non-linear modal of quad-rotor aircraft is derived (which is an under-actuated with six degrees of freedom (6DOF) is derived) using the basic laws of physics. The derived model is highly unstable, so first the model is linearized and then a linear multivariable control strategy is developed for the purpose of first stabilizing the aircraft and secondly forcing the aircraft to follow a certain fix reference value. The performance of this linear control strategy is evaluated using simulations on Simulink/MATLAB and the results show the effectiveness of the LQR controllers.

CHAPTER 1

INTRODUCTION

This is an introductory chapter which will give the reader some background knowledge about the project and also help the reader to quickly understand the basic aims of the project. The first part of this chapter gives a brief explanation about the theoretical background and important terms related to the project. The part of this chapter states the problem statement and the main goal which we are planning to achieve.

UAV's (Unmanned Aerial Vehicles) have recently become very popular. Now a day's low price and small scale UAV's are available on different websites and are easily purchasable. Quad copters also fall under the category of UAV's. It has a cross like frame with four rotors mounted at each corner as shown in the figure 1.1.

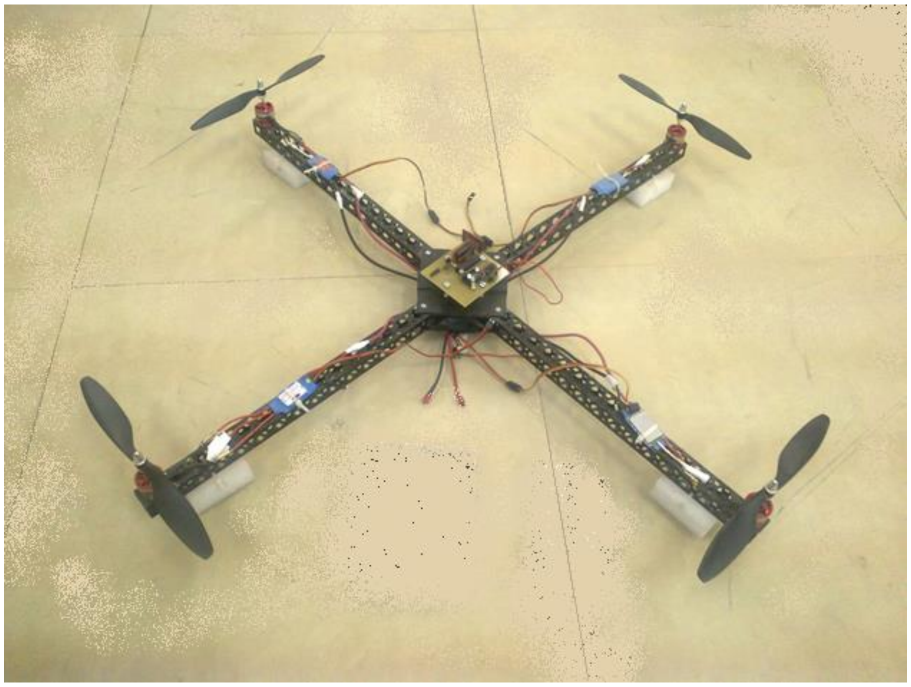

Fig 1.1: Quad-X Frame

Different quad rotor platforms are commercially available on the website and now a day like Turbo Ace X830-D [1], ArduCopter [2], ARDrone quadcoptor [3] and many others. These aircrafts are bought by hobbyists, professional flyers or teaching institutes for different purposes like entertainment, flying competitions or research. Due to the unavailability of the readymade frames in Pakistan and high shipment cost I have designed and built a custom quad rotor aircraft for this project and named it 'Quad-X'.

There are several applications of such aircrafts and some of them are listed below:

1. A Research platform for the students
2. Traffic monitoring and aerial surveillance
3. In military for search operations
4. Inspection of disaster hit areas and many others

The list of applications described above each requires different levels of control. For some applications high precision and accuracy is mandatory on the other hand some applications may work with large tolerances.

1.1 Problem Statement

Here I will describe the problem statement of the project and the goals that are set and the necessary steps to achieve those goals.

The goal of this project is divided in to three parts which will be implemented separately step by step. The first two parts are mandatory and the third part is optional and will be implemented provided if the time resources are available. The first part is the design and built of the Quad-X frame. The second part is related to the development of a control scheme which we will be implementing on the mathematical model of the Quad-X frame and check the results of the developed controller using MATLAB. The third part is the implementation of the controller on the actual frame.

These three parts can be further divided into sub parts and the completion of all the sub-parts is mandatory for the achievement of the main goal. The collective sub-parts are listed below which serves as the checklist for the development of the project.

- Mechanical design, procurement of components and assembly of the Quad-X frame.
- Modelling and development of a viable state's space modal of the Quad-X frame.
- Design, implementation and simulations of a stable hovering controller on the developed model using MATLAB.
- Analysing the controller's ability to stabilize the model using MATLAB.
- Analysing the controller's ability to follow a fixed reference path using MATLAB.
- Implementation of the controller on the Quad-X frame and analysing the results.

1.2 Chapter Summery

The main aims and objectives of the project have been discussed in this chapter. Some theoretical background has also been explained in order for the reader to get along with the project. The labs used throughout the project development are CNC machines lab and projects lab at Wah Engineering College. The main focus of the next chapters would be how to build the frame and develop its mathematical model. Design and implementation of a controller for stable hovering and fixed input following would be discussed next.

CHAPTER 2

SYSTEM DESCRIPTION

This chapter deals with the description of the labs used along with the facilities in the development of the project. The first part of the chapter describes the hardware components of the Quad-X frame used in developing the aircraft and the later part gives a little knowledge about the theory of aircraft specifically its behaviour in mid-air with respect to some given inputs which will help the reader to understand about the manoeuvrability of the aircraft.

2.1 Quad-X Hardware

This section will present the hardware description of the Quad-X aircraft. The Quad-X aircraft have been designed, assembled and maintained in the robotics lab of Wah Engineering College.

In figure 2.1 the hardware description of the Quad-X frame is shown. It contains two aluminium square pipes, a square core made up of Plexiglas material, four BL-DC motors, four propellers (two normal and two with counter pitch), Li-Po battery packs and main circuit board. We can also see three small gyroscopes placed besides the battery packs. An additional component which is missing in this fig and will be explained later is the 7 channel RC made by Futaba with a TX and RX modules [4].

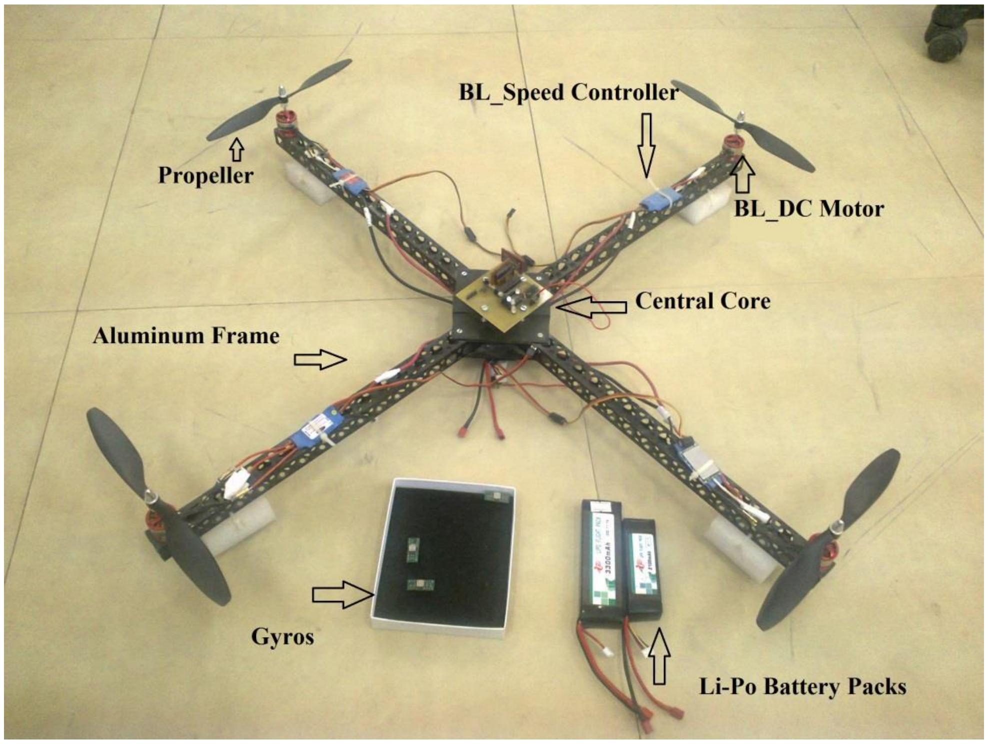

Fig 2.1: Hardware description of the Quad-X Frame

2.2 Quad-X Remote Control and Manoeuvring

The size of the Quad-X frame depicts that its manoeuvrability is normal neither too low nor too high. This section presents some of the most common manoeuvres of a quad rotor aircraft with some predefined inputs. Most of the complex movements of a quad rotor are simply a combination of these basic manoeuvres. Moreover it was thought to be of great importance that the reference frames used in the modelling part should be introduced here as well.

2.2.1 Reference frames

For the better understanding of the Quad-X manoeuvres two reference frames are defined here one called the earth frame **E** which is fixed with the real world and other is the body frame **B** which is fixed at the centre of the aircraft. The orientation and placement of the frames is shown in fig 2.2. These frames are basically the 3D coordinates systems in which the quad rotor manoeuvres. The coordinates system will be used as a reference in the modelling part of this aircraft. The vectors in the earth frame **E** will be referred as $\boldsymbol{a^E}$and those in the body frame **B** will be referred to as $\boldsymbol{a^B}$.The Quad-X is able to rotate freely around the three body axis and signs are defined based on the inputs given positive inputs means clockwise rotation and negative inputs means counter clockwise rotation.

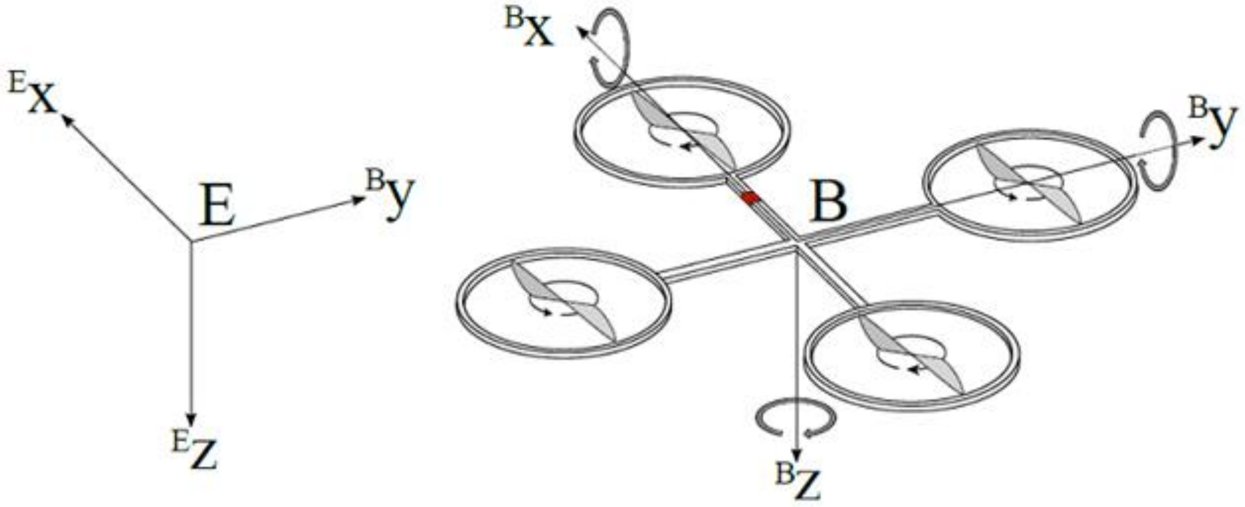

Fig 2.2: Description of the coordinates system used in the project [5]

2.2.2 Quad-X Control Inputs

The Quad-X uses four rotors to generate lift. Two of them are rotating in the clockwise direction and the other two are rotating in the counter clockwise direction. These are fixed pitched rotors as opposed to conventional helicopters. The main principle behind the movement of a Quad-X aircraft is the speed difference between the rotors i.e. the system can be completely controlled by changing the rotational speed of the four rotors thus varying the thrust and torques produced. Fig 2.3 describes the change in the Quad-X attitudes with the given remote control inputs. There are two sticks on the main remote control module and both of them can either move up-down or left-right thus giving an attitude changes to the Quad-X frame as depicted in fig 2.3

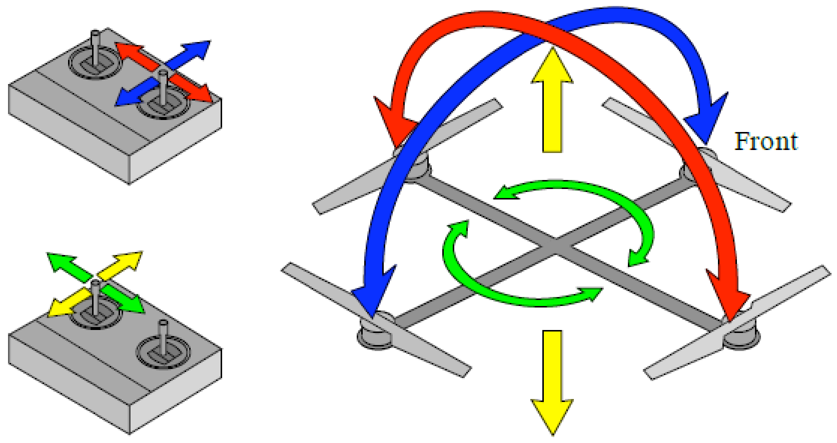

Fig 2.3: Quad-X sketch with wireless remote control interface [5]

2.2.3 Quad-X Manoeuvring

After the complete description of the reference frames and the control inputs of the Quad-X frame the manoeuvring possibilities are now discussed in detail.

The upward forces of the four rotors are generated by their angular speeds which are controlled by the wireless remote control as shown in fig 2.3. When the rotors rotate it splashes through the air and due to the frictional force between the air particles and the rotor motion two forces are generated. One is the lift force and the other is the drag force. The lift force lifts the quad rotor in the upward direction as shown in fig 2.4. The drag force results in counter moments on the rotors known as the induced moments ($\tau_1\ to\ \tau_4$) which are in the opposite direction of the rotation of the rotors as shown in fig 2.4. On a conventional helicopter tail rotor is specifically placed to counter the effect of this induced moment in order to stop the helicopters body from spinning, but in our case we can achieve the same results by simply rotation the rotor in pairs with two of them rotation in one direction and the other two rotation in the opposite direction as shown in fig 2.4.

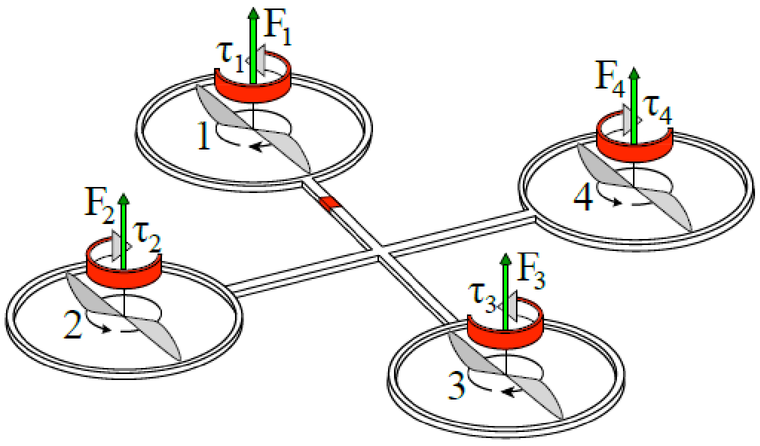

Fig 2.4: Forces and Torques of the Quad-X frame while hovering [5]

2.2.4 Mathematical description of Forces and Torques

Starting from the Newton's second law for translational and rotational systems we get the two famous relations as shown:

$$\sum F = ma \tag{2.1}$$

$$\sum \tau = J\alpha \tag{2.2}$$

Where **'F'** is the total force applied to any rigid body **'m'** is the mass of the body and **'a'** is the linear acceleration of the body. Similarly **'τ'** is the total torque exerted to the body **'J'** is the moment of inertia and 'α' is the angular acceleration.

While in hovering mode the Quad-X is in Force and Torque balance. Equal thrust and torques are generated by the rotors at equal speeds because of the similarity in their design which means that if the four rotors are rotation at equal speeds the total torque exerted on the Quad-X frame about the Z-body axis should be equal to zero. Similarly at equal speeds the thrust forces generated by each rotor would be same resulting in zero angular acceleration about the X- and Y-body axis. The only issue now is the huge amount of stresses being exerted at the central core of the Quad-X frame. In order to deal with the stress the material of the core and its thickness should be selected wisely. This part is discussed in detail in the mechanical design section of this report which will come later on.

2.2.5 Roll, Pitch and Yaw movements

Fig 2.5(a), (b), (c) depicts the roll, pitch and yaw movements of the Quad-X frame from left to right respectively along with their control inputs.

Roll means the rotation of the Quad-X frame about the X-body axis. This could be achieved by changing the speeds of rotor 2 and rotor 4 while maintaining the speeds of rotor 1 and rotor 3. For positive or clockwise movement about X-body axis the speed of rotor 2 is increased and the speed of rotor 4 is decreased as shown in fig 2.5(a).

Pitch means the rotation of the Quad-X frame about the Y-body axis. This could be achieved by changing the speeds of rotor 1 and rotor 3 while maintaining the speeds of rotor 2 and rotor 4. For positive or clockwise movement about Y-body axis the speed of rotor 1 is increased and speed of rotor 3 is decreased as shown in fig 2.5(b).

Yaw means the rotation of the Quad-X frame about the Z-body axis. As we already know that the induced moments generated by the rotors movement exert torques on the frame. The net torque exerted on the aircraft is zero as long as the torques exerted by the pair of rotors rotating in the clockwise direction is equal to the torques exerted by the pair of rotors rotating in counter clockwise direction. Thus generating a net zero torque but if we want to achieve a yaw movement we need to mismatch the pair of torques generated by the two pairs for example if we want to achieve a positive yaw rotation about the Z-body axis we will simply increase the speeds of rotor 2 and rotor 4 and decrease the speeds of rotor 1 and rotor 3 as shown in fig 2.5(c).

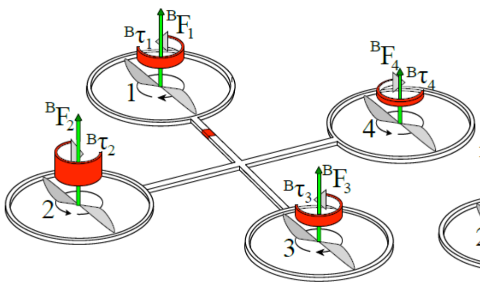

Fig 2.5(a): Roll movement [5]

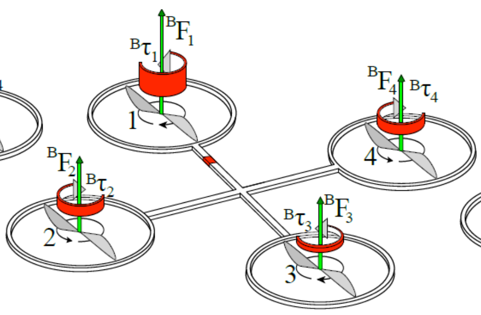

Fig 2.5(b): Pitch movement [5]

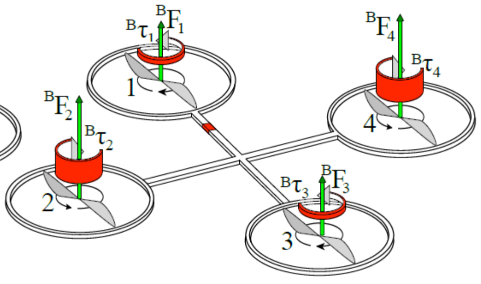

Fig 2.5(c): Yaw movement [5]

2.3 Chapter Summery

In this chapter the overall system analysis is carried out. Different hardware components used are discussed. System parameters are also discussed and system behaviour to different inputs is discussed.

In the next chapter a detailed mechanical design analysis is carried regarding the frame of Quad-X aircraft.

CHAPTER 3

DESIGN AND STRATEGY

This chapter describes some basic calculations related to the mechanical design of the Quad-X frame and the modelling and control strategies used in order to achieve stable hover.

3.1 Mechanical design of the Quad-X frame

The Mechanical Design has been categorized in the following sections:

a. Basic Design

b. Material Selection

c. Design Calculations

3.1.1 Basic Design

The basic design consists of two links attached together in a cross formation. The links are reinforced at the center by plates. Motors are mounted at the end of the links as shown in fig 3.1.

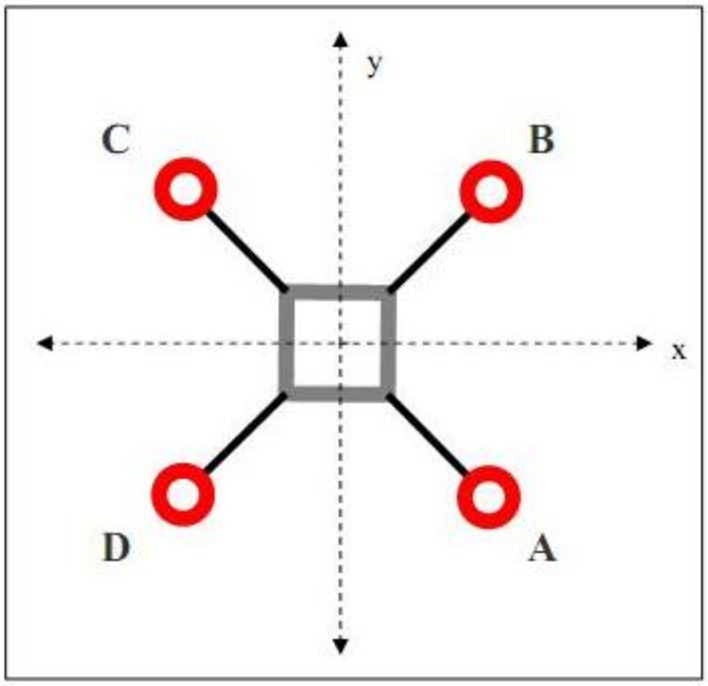

Fig 3.1: The basic Quad copter schematic

3.1.2 Material Selection

The criterion for the selection of material was:

- Durability
- Strength
- Machine ability
- Light Weight
- Availability
- Cost

So the material selected was Aluminum

Physical Properties	**Metric System**
Density	2560kg/m^3
Mechanical Properties	
Ultimate Tensile Strength	70MPa
Modulus Of Elasticity	70GPa

Table 3.1: Properties for Aluminum Alloy 1100-H14

3.1.3 Design Calculations

3.1.3.1 Lift Calculations

As the blades have a natural twist to make the lift force constant along the length of the blade.

Angle of attack θ =16.6 ^{0}

The formula for lift is:

$$L = \frac{1}{2}\,(\rho . V^2 . A . C_L) \tag{3.1}$$

Where:

L=Lift force

p=Density of air

V=Linear velocity of rotors

A=Area swept by the rotors

C_L=Coefficient of lift

According to the datasheet of the motors, the maximum lift force they can provide with 12X6 propeller is = 550g X 9.81 = 5.4N

For the Arms we selected Square pipes of Aluminum 12.5 X 12.5 mm which are easily available in the market. The material selected for the central part (base plate) of the frame is Plexiglas of 2.2mm. The stress calculations for above selected material are as follows:

The critical points in the frame are the joints of the arms with the base plate. Considering the arms as cantilevers, so the moment generated by the lift force is shown in fig 3.2:

3.1.3.2 Bending Stress Calculations:

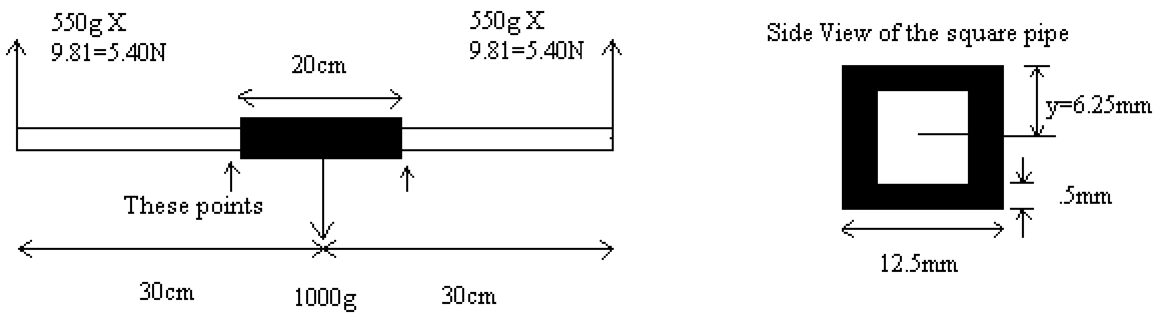

Fig 3.2: Orthographic Views of the Aluminum pipe

Moment is given by:

$$M = \frac{30x5.40}{100} = 1.62Nm$$

The moment of inertia is:

$$I = \frac{(0.0125^4 - 0.0105^4)}{12} = 1.0216x10^{-9}m^4$$

Centroid is:

$$y = 6.25mm$$

So the bending stress is:

$$\sigma_b = \frac{M.y}{I} = 9.92MPa \ll 70MPa\ (Max. Bending\ Stress)$$

3.1.3.3 Maximum Deflection

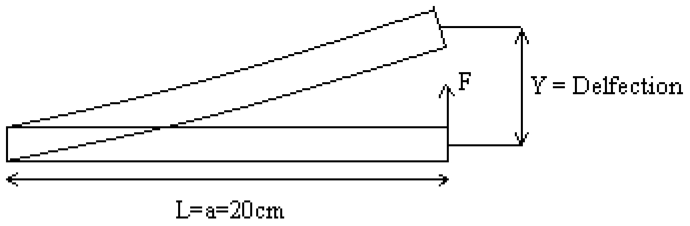

Fig 3.3: Beam Deflection

$Force = F = 5.40N$
$Arm\ Length = L = a = 20cm$
$Modulus\ of\ Elasticity = E = 70GPa$
So,

$$Beam\ deflection = Y = \frac{F.a^2.(3L - a)}{6.E.I} = 0.201mm$$

3.1.3.4 Bearing Stresses on Bolts and Base Plate

As bending stress is:

$\sigma_b = 9.92MPa$
And area is:

$A = 12.5mmx12.5mm - 11.5mmx11.5mm = 24x10^{-6}m^2$

So force will be:

$F = 9.92x10^6x24x10^{-6} = 238.08N$

Bolt diameter is:

$d = 3mm$

Area of the bolt under stress is:

$A_{bolt} = \pi.r^2 = 3.14x1.5^2mm^2 = 7.065mm^2$

So bearing stress on bolt is:

$$\sigma_{bearing}(bolt) = \frac{F}{A_{bolt}} = \frac{238.08N}{7.065mm^2} = 33.7MPa \ll 70MPa$$

Hence it's safe to use the bolts.

Now to calculate the bearing stress on the base plate:

Area of the plate in contact with the bolt is:

$$A_{plate} = 2.\pi.r.l = 2x3.14x1.5x2.2mm^2 = 20.9124mm^2$$

$$\sigma_{bearing}(base\ plate\ ) = \frac{F}{A_b} = \frac{238.08N}{20.9124mm^2} = 11.38MPa \ll 70MPa$$

Hence it is safe to use the plate.

3.2 Chapter Summery

In this chapter a detailed mechanical design analysis is carried out in order to make the frame of Quad-X which can bear all the stresses being applied due to different forces being generated during flight. A general block diagram and a dimensioned diagram of the main components of Quad-X frame are also given.

The main aim of the next chapter is to draw a viable state space model if Quad-X aircraft for control purpose.

CHAPTER 4

MODELLING

In this chapter a non-linear model of Quad-X is derived from the simple laws of Physics. The mathematical modal basically consists of equations of motion of the aircraft in three dimensions. The non-linear model is first simulated in MATLAB to see its behaviour for some given inputs. Finally the non-linear model is linearized for the purpose of control and the linear model is then validated by comparing its results with the non-linear model.

4.1 Quad rotor dynamics

The block diagram of the approximated system model is given below [5, page 24 fig 4.1].

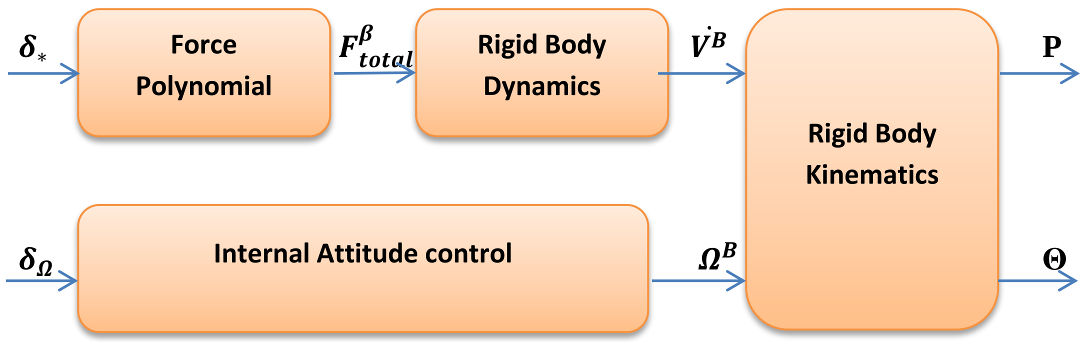

Fig 4.1: Approximated System Model

The three reference angular inputs to the quad rotor which are given as δ_ϑ, δ_ξ and δ_ϖ and the dynamic system of equations assumes the structure as shown below [5, page 25 eq. 4.1-4.3].

$$\Omega^B = \delta_\Omega \,.\, \lambda_\Omega \tag{4.1}$$

Where,

$$\lambda_\Omega = \begin{bmatrix} \lambda_\vartheta \\ \lambda_\xi \\ \lambda_\varpi \end{bmatrix} = \begin{bmatrix} 0.0011574 \\ 0.00092612 \\ 0.0016476 \end{bmatrix} \tag{4.2}$$

Also,

$$\Omega^B_\vartheta = \delta_\vartheta.\lambda_\vartheta, \Omega^B_\xi = \delta_\xi.\lambda_\xi \; and \; \Omega^B_\varpi = \delta_\varpi.\lambda_\varpi \tag{4.3}$$

Where,

$\delta_\vartheta \,, \delta_\xi \; and \; \delta_\varpi \; are \; the \; refference \; input \; values \; for \; roll, pitch \; and \; yaw \; respictively \; and$

$\lambda_{\vartheta}, \lambda_{\xi}$ *and* λ_{ϖ} *are the constants derived form experimental data as shown in* [5, *Appendix I*]

Two important equations relating translational and angular velocities in the body frame to the earth frame are as follows:

$$V^E = T_B^E(\varnothing) \, . V^B \text{ and } \Omega^E = A_B^E(\varnothing) \, . \Omega^B \tag{4.4}$$

Where,

$T_B^E(\varnothing) = \textit{Transformation matrix for linear velocities from body to earth frame and}$

$A_B^E(\varnothing) = \textit{Transformation matrix for angular velocities from body to earth frame}$

The values of the Transformation matrices are taken from [5, Appendix J]

Propellers exhibit the main forces and torques on the frame. Propeller 1, 3 are rotating in clockwise direction and 2, 4 are rotating in anti-clockwise direction. Total upward force is thus given by:

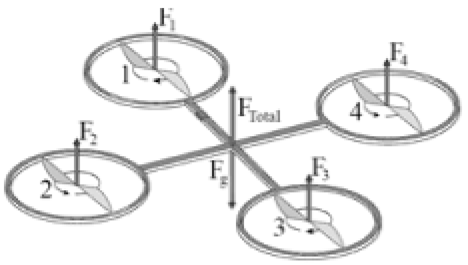

Fig 4.2: Net vertical force

$$F_{total}^{\beta} = F_1^{\beta} + F_2^{\beta} + F_3^{\beta} + F_4^{\beta} + F_g^{\beta} \tag{4.5}$$

$$F_{total}^{\beta} = F_{prop.}^{\beta} + F_g^{\beta} \tag{4.6}$$

Where,

$F_1^{\beta} - F_4^{\beta} = \textit{Forces generated by the propellers in the } z - \textit{body axis}$

And,

$F_g^{\beta} = \textit{gravitation force in the body frame.}$

The gravitational force in the body frame is calculated below by using Newton's second law $\boldsymbol{F = ma}$ and a transformation matrix from earth frame to body frame $\boldsymbol{T_E^B}$ as shown below in equation 4.7

$$F_g^E = m.g \text{ and } F_g^B = T_E^B \, . F_g^E \rightarrow F_g^B = T_E^B \, . m.g \tag{4.7}$$

And g = $\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$

Furthermore the translational acceleration $\boldsymbol{\dot{V}^E}$ as seen from the earth can also be calculated using Newton's second law.

$$F = m.a \;\rightarrow F^E_{total} = m\,.\dot{V}^E \;\rightarrow \dot{V}^E = \; F^E_{total}/m \tag{4.8}$$

The translational velocity $\boldsymbol{V^E}$ as seen from earth frame can also be written as follows:

$$V^E = \; T^E_B(\obslash)\,.V^B \tag{4.9}$$

Differentiating equation (4.9) from both sides we get:

$$\dot{V}^E = \; T^E_B(\obslash)\,.\dot{V}^B + \; \dot{T}^E_B(\obslash)\,.V^B \tag{4.10}$$

In the above equation the derivative of the transformation matrix can be written as the cross product as shown below in equation 4.11 [6, page 24 equation 4.4]

$$\dot{T}^E_B(\obslash).V^B = \; -T^B_E(\obslash)\,.(\Omega^B x V^B) \tag{4.11}$$

Where,

$\Omega^B = Angular\ velocity\ as\ seen\ from\ the\ body\ frame$

Putting (4.11) into (4.10), we get:

$$\dot{V}^E = \; T^E_B(\obslash)\,.\dot{V}^B - T^B_E(\obslash)\,.(\Omega^B x V^B) \tag{4.12}$$

Now after comparing the two equations (1.8) and (1.12) and simplifying yields the translation acceleration of the air craft as seen from body frame as shown below:

$$F^E_{total}/m \; = \; T^E_B(\obslash)\,.\dot{V}^B - T^B_E(\obslash)\,.(\Omega^B x V^B) \tag{4.13}$$

$$T^E_B(\obslash)\,.\dot{V}^B \; = \; F^E_{total}/m + T^B_E(\obslash)\,.(\Omega^B x V^B) \tag{4.14}$$

Multiplying both sides by $\boldsymbol{T^B_E}(\obslash)$ we get:

$$T^B_E(\obslash)(T^E_B(\obslash)\,.\dot{V}^B) \; = \; T^B_E(\obslash)(F^E_{total}/m + T^B_E(\obslash)\,.(\Omega^B x V^B)) \tag{4.15}$$

$$\dot{V}^B \; = \; T^B_E(\obslash)(F^E_{total}/m) + T^B_E(\obslash)(T^B_E(\obslash)\,.(\Omega^B x V^B)) \tag{4.16}$$

$$\dot{V}^B \; = \; F^B_{total}/m + \; (\Omega^B x V^B) \tag{4.17}$$

As,

$T_E^B(\varnothing)\,.T_E^B(\varnothing) = Identity$

For $\boldsymbol{F_{prop.}^{\beta}}$ the following cubic polynomial is considered [5, Appendix H]

$$F_{prop.}^{\beta}(\delta_*) = \begin{bmatrix} 0 \\ 0 \\ 3.2174.10^{-10}.\delta_*^3 - 1.9531.10^{-6}.\delta_*^2 + 5.4632.10^{-4}.\delta_* - 0.6698 \end{bmatrix} \tag{4.18}$$

Where,

$\delta_* = is\ the\ control\ input\ for\ thrust\ force\ generation$

Equation (4.18) shows that for the given range of inputs ($\boldsymbol{\delta_{*L}} - \boldsymbol{\delta_{*H}}$) there is a range of net upward thrust ($\boldsymbol{F_{prop.(L)}^{\beta}}(\boldsymbol{\delta_{*L}}) - \boldsymbol{F_{prop.(H)}^{\beta}}(\boldsymbol{\delta_{*H}})$).

Where,

$\delta_{*L} = lowest\ possible\ value\ of\ thrust\ input$

$\delta_{*H} = Heighest\ possible\ value\ of\ thrust\ input$

$F_{prop.(L)}^{\beta}(\delta_{*L}) = lowest\ possible\ value\ of\ the\ thrust\ generated\ \boldsymbol{and}$

$F_{prop.(H)}^{\beta}(\delta_{*H}) = Heighest\ possible\ value\ of\ the\ thrust\ generated$

Also equation (4.18) shows that whenever a constant input is applied a constant acceleration in the negative z-direction is generated where as in actual the aircraft settles at a constant velocity after initial acceleration and does not accelerates indefinitely. Based on the work of [7] we add an additional term to the model this which simulates the induced inflow through the quad rotor aircraft during upward movement.

$$F_{inflow}^{B} = I_f.V_Z^B \tag{4.19}$$

Induced flow is basically the total perpendicular airflow to the propellers minus the generated airflow by the rotors circular motion. Induced airflow is a negative constant and its value found by [7, Appendix A] is used. The upward force $\boldsymbol{F_{prop.}^{\beta}}(\boldsymbol{\delta_*})$ the gravitational force $\boldsymbol{F_g^E}$ and the induced inflow force $\boldsymbol{F_{inflow}^{B}}$ results in a dynamic system equation as shown below:

$$F_{total}^{\beta} = F_{prop.}^{\beta} + F_g^{\beta} + F_{inflow}^{B} \tag{4.20}$$

$$F_{total}^{\beta} = F_{prop.}^{\beta}(\delta_*) + T_E^B\,.m.g + I_f.V_Z^B \tag{4.21}$$

This concludes the modeling section of the quad rotor aircraft. Next we will simulate this non-linear modal in MATLAB Simulink.

4.2 Non-Linear Model Simulations

In order to simulate the non-linear model of quad rotor aircraft its system of non-linear equations are implemented on Simulink/Matlab by creating its non-linear simulation model. First the complete system of equations is gathered below (eq. 4.22 to 4.26) for clear visualization.

$$\Omega^B = \delta_\Omega \,.\, \lambda_\Omega \tag{4.22}$$

$$\lambda_\Omega = \begin{bmatrix} \lambda_\vartheta \\ \lambda_\xi \\ \lambda_\varpi \end{bmatrix} = \begin{bmatrix} 0.0011574 \\ 0.00092612 \\ 0.0016476 \end{bmatrix} \tag{4.23}$$

$$V^E = T_B^E(\varnothing) \,.\, V^B \; and \; \Omega^E = A_B^E(\varnothing) \,.\, \Omega^B \tag{4.24}$$

$$\dot{V}^B = F_{total}^B / m + (\Omega^B x V^B) \tag{4.25}$$

$$F_{total}^\beta = F_{prop.}^\beta(\delta_*) + T_E^B \,.\, m.\, g + I_f . V_Z^B \tag{4.26}$$

The non-linear simulation model of quad-rotor is shown below in Fig 4.3. The transformation matrices of translational and angular velocities from body frame to earth frame and vice versa are discussed in detail in [5, Appendix J]. The model is simulated with zero initial conditions i.e. [0, 0, 0] for x, y, z and [0, 0, 0] for pitch, roll and yaw respectively.

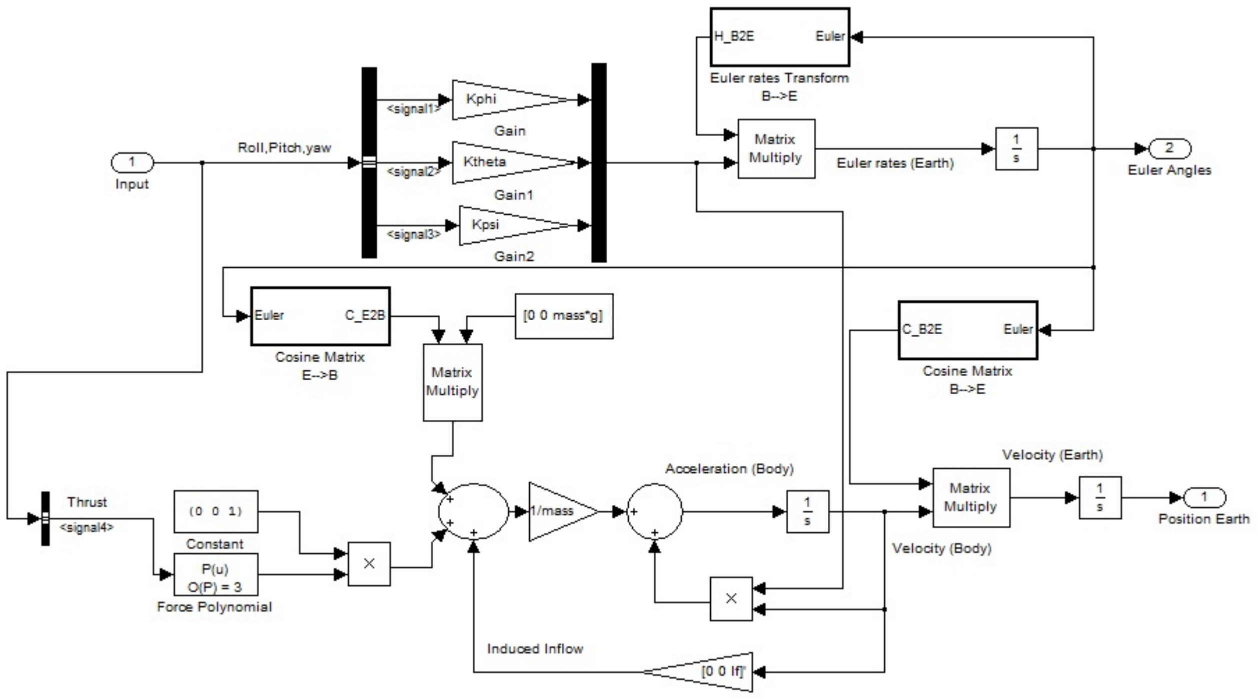

Fig 4.3: Non-linear Simulation Modal

The inputs given to the above model and their corresponding outputs are shown below:

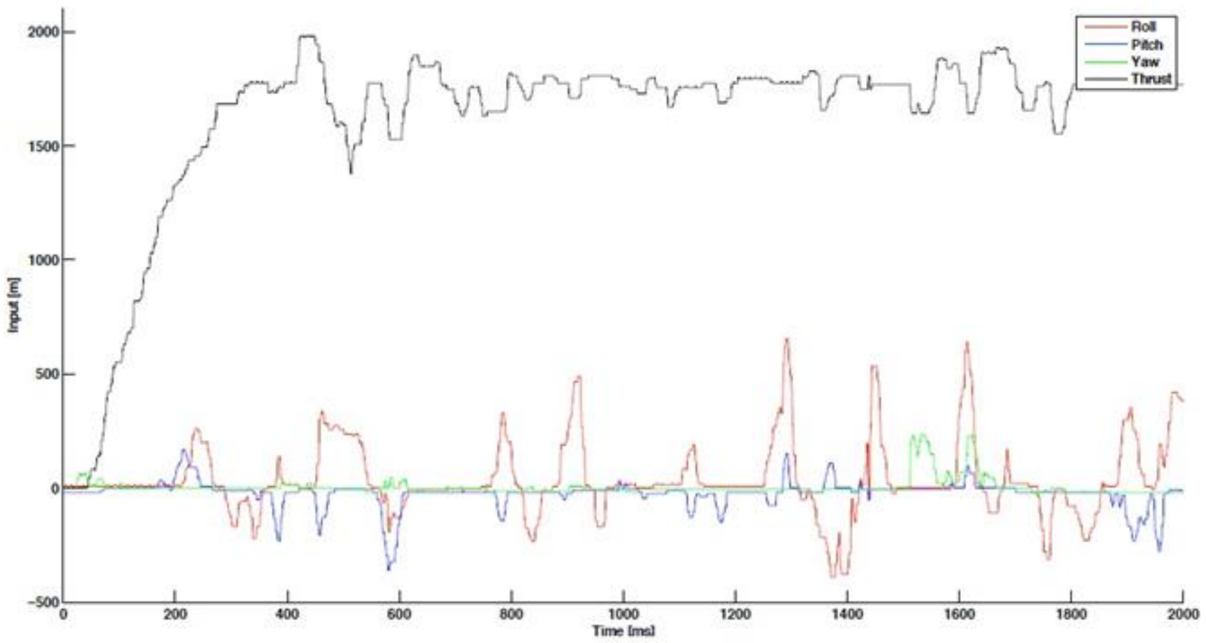

Fig 4.4: Input Signals to the Non-Linear Model

The output graphs generated are shown below:

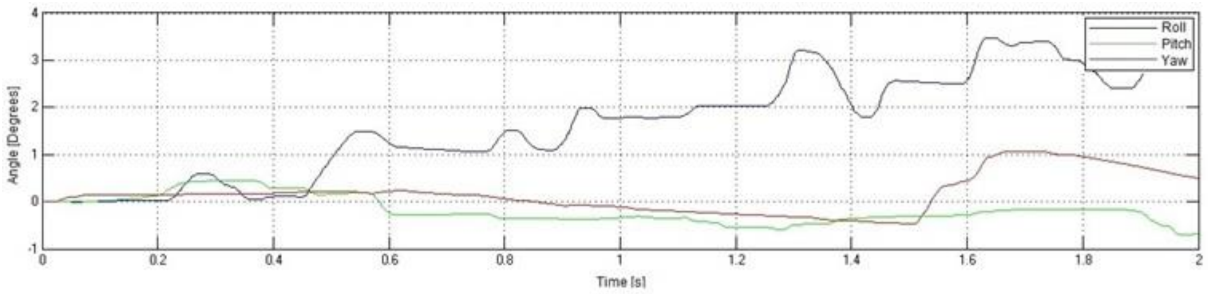

Fig 4.5: Estimated Attitude of the Non-Linear Model of Quad-X

We see in fig 4.5 that the air craft maintains its orientation for very short interval in after that it starts to deviate and the deviation increases further after that and after 2 seconds the air craft is significantly tilted. Keeping these results in mind we can easily justify the divergence of the translational movements of the quad rotor aircraft because when the platform is tilted the Thrust force $\boldsymbol{F}_{\boldsymbol{Total}}^{\boldsymbol{B}}$ is now pointing not only in the z-direction but it is making components in the x and y directions as well. Following figure shows the estimated positions of the Quad-X model.

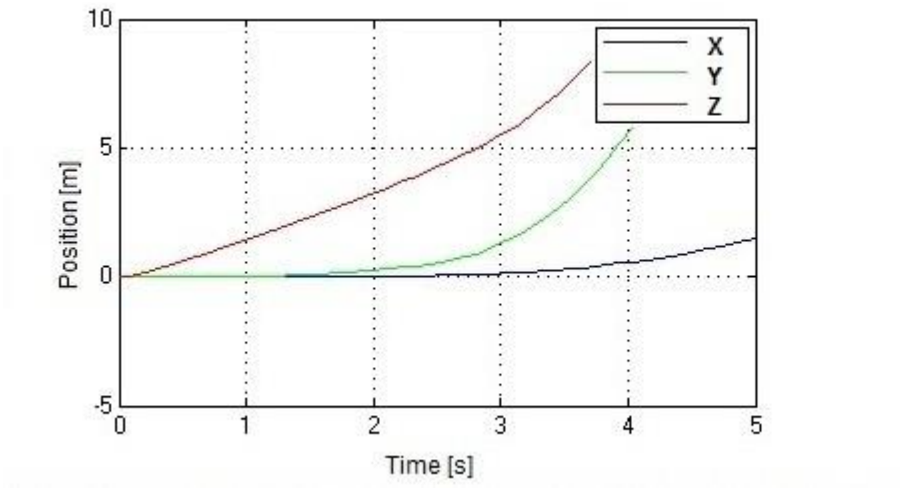

Fig 4.6: Estimated positions of the Non-Linear Model of Quad-X

4.3 Linearization of the Quad rotor Model

The controller chosen is linear and is suitable for multivariable control. In order to implement this controller on the quad rotor model it needs to be linearized first and after linearization its viable state space model must be constructed for the implementation of this controller.

4.3.1 Linearization Using Taylor Series approximation

The system of non-linear dynamic equations of quad rotor aircraft is linearized using Taylor series approximation about a certain operating point in which the functions are approximated by an infinite sum of terms containing the derivatives of the function. First Order Taylor Series approximation is used for the linearization of the Quad-X model in which only the first order term is calculated and the rest of the terms are ignored. Taylor Series approximation with one variable is shown in equation 4.27.

$$f(x) \approx f(\bar{x}) + \frac{\dot{f}(\bar{x})}{1!}(x-\bar{x})^1 + \frac{\ddot{f}(\bar{x})}{2!}(x-\bar{x})^2 + \frac{\dddot{f}(\bar{x})}{3!}(x-\bar{x})^3 + \cdots + \frac{f^n(\bar{x})}{n!}(x-\bar{x})^n \qquad (4.27)$$

The First Order Taylor Series approximation is shown in equation 4.28.

$$f(x) \approx f(\bar{x}) + \frac{\dot{f}(\bar{x})}{1!}(x-\bar{x})^1 \Big|_{n=1} \qquad (4.28)$$

The linearization of any system is usually done around the point where system behaves normally. This is the case when the x,y-body plane is aligned with the x,y-earth plane and Quad-X is in perfect hover and the yaw angle is zero. So the linearization can thus be approximated as $[\vartheta, \xi, \varpi]^T \approx [0, 0, 0]^T$ which means that the attitude is maintained close to zero. So the linearization is done about the angles $\Theta = 0 + \bar{\Theta} \mid \Theta \in \vartheta, \xi, \varpi$ where $\bar{\Theta}$ indicates small angular deviation. The parameters of our system will be affected by this assumption of the operating point as follows:

$$\Omega^B \approx 0 \; and \; V^B \approx 0 \tag{4.29}$$

Now using the First Order Taylor approximation is applied on the trigonometric functions around the operating point zero as shown in equations 4.30 to 4.32.

$$sin(\Theta) \approx sin(0) + cos(0).\bar{\Theta} \approx \bar{\Theta} \tag{4.30}$$

$$cos(\Theta) \approx cos(0) + (-sin(0)).\bar{\Theta} \approx 1 \tag{4.31}$$

$$tan(\Theta) \approx sin(0) + \frac{1}{(cos(0))^2}.\bar{\Theta} \approx \bar{\Theta} \tag{4.32}$$

4.3.2 Linearization of Kinematic and Dynamics of Quad-X

Referring to equation 4.4 linear velocities and angular velocities or earth frame and body frame are as follows:

$$V^E = T_B^E(\oslash)\,.V^B \; and \; \Omega^E = A_B^E(\oslash)\,.\Omega^B \tag{4.33}$$

The linearization of equation 4.33 means simplifying the direction cosine matrix $T_B^E(\oslash)$ and transformation matrix $A_B^E(\oslash)$ respectively. Both these matrices in their original form and linearized form are shown below.

$$T_B^E(\oslash) = \begin{bmatrix} c\xi c\varpi & s\vartheta s\xi s\varpi - c\vartheta s\varpi & c\vartheta s\xi c\varpi + s\vartheta s\varpi \\ c\xi s\varpi & s\vartheta s\xi s\varpi + c\vartheta c\varpi & c\vartheta s\xi s\varpi - s\vartheta c\varpi \\ -s\xi & s\vartheta c\xi & c\vartheta c\xi \end{bmatrix} \tag{4.34}$$

$$\approx \begin{bmatrix} 1 & \vartheta\xi\varpi - \varpi & \xi + \vartheta\varpi \\ \varpi & \vartheta\xi\varpi + 1 & \xi\varpi - \vartheta \\ -\xi & \vartheta & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & -\varpi & \xi \\ \varpi & 1 & -\vartheta \\ -\xi & \vartheta & 1 \end{bmatrix} \tag{4.35}$$

After substituting linearized cosine matrix in equation 4.33 we get the following relationship if translational velocities in earth and body frames.

$$V^E = T_B^E(\oslash)\,.V^B \tag{4.36}$$

$$\approx \begin{bmatrix} 1 & -\varpi & \xi \\ \varpi & 1 & -\vartheta \\ -\xi & \vartheta & 1 \end{bmatrix}.V^B \tag{4.37}$$

$$\approx \begin{bmatrix} 1 & -\varpi & \xi \\ \varpi & 1 & -\vartheta \\ -\xi & \vartheta & 1 \end{bmatrix} . \begin{bmatrix} V_x^B \\ V_y^B \\ V_z^B \end{bmatrix} \tag{4.38}$$

$$\approx \begin{bmatrix} V_x^B - \varpi . V_y^B + \xi . V_z^B \\ \varpi . V_x^B + V_y^B - \vartheta . V_z^B \\ -\xi . V_x^B + \vartheta . V_y^B + V_z^B \end{bmatrix} \approx \begin{bmatrix} V_x^B \\ V_y^B \\ V_z^B \end{bmatrix} = V^B \tag{4.39}$$

Similarly we can follow the same procedure to simplify the transformation matrix $A_B^E(\obslash)$.

$$A_B^E(\obslash) = \begin{bmatrix} 1 & t\xi s\vartheta & t\xi c\vartheta \\ 0 & c\vartheta & -s\vartheta \\ 0 & s\vartheta / c\xi & c\vartheta / c\xi \end{bmatrix} \approx \begin{bmatrix} 1 & \xi\vartheta & \xi \\ 0 & 1 & -\vartheta \\ 0 & \vartheta & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & \xi \\ 0 & 1 & -\vartheta \\ 0 & \vartheta & 1 \end{bmatrix} \tag{4.40}$$

And after substituting it in equation 4.33 following results are obtained.

$$\Omega^E = A_B^E(\obslash) . \Omega^B \tag{4.41}$$

$$\approx \begin{bmatrix} 1 & 0 & \xi \\ 0 & 1 & -\vartheta \\ 0 & \vartheta & 1 \end{bmatrix} . \begin{bmatrix} \Omega_\vartheta^B \\ \Omega_\xi^B \\ \Omega_\varpi^B \end{bmatrix} \approx \begin{bmatrix} \Omega_\vartheta^B + \xi . \Omega_\varpi^B \\ \Omega_\xi^B - \vartheta . \Omega_\varpi^B \\ \vartheta . \Omega_\xi^B + \Omega_\varpi^B \end{bmatrix} \approx \begin{bmatrix} \Omega_\vartheta^B \\ \Omega_\xi^B \\ \Omega_\varpi^B \end{bmatrix} = \Omega^B \tag{4.42}$$

Now referring to equation 4.21 related to the forces affecting the Quad-X frame are presented and linearized. The method of linearization is same as before. First we will write the original equation as follows:

$$F_{total}^{\beta} = \underbrace{F_{prop.}^{\beta}(\delta_*)}_{(1)} + \underbrace{T_E^B \ . m . g}_{(2)} + \underbrace{I_f . V_Z^B}_{(3)} \tag{4.43}$$

Where,

$(1) = Rotor\ Force$

$(2) = Gravitational\ Force$

$(3) = Induced\ Inflow$

The only non-linear part is the gravitational force with a direction cosine matrix so we will focus on linearizing this part as we did with the translational and angular velocities. The only additional step is that here we need the transpose of the previously used direction cosine matrix.

$$T_B^E(\obslash)^T = T_E^B(\obslash) \tag{4.44}$$

$$F_g^{\beta} = T_E^B(\varnothing).\begin{bmatrix}0\\0\\g\end{bmatrix}.m = \begin{bmatrix}1 & \varpi & -\xi\\ -\varpi & 1 & \vartheta\\ \xi & -\vartheta & 1\end{bmatrix}.\begin{bmatrix}0\\0\\g\end{bmatrix}.m = \begin{bmatrix}-\xi\\ \vartheta\\ 1\end{bmatrix}.m.g \tag{4.45}$$

Now referring to equation 4.17 and including the assumption of equation 4.29 we get:

$$\dot{V}^B = F_{total}^B/m + (\Omega^B x V^B) \tag{4.46}$$

$$\dot{V}^B \approx \frac{F_{total}^B}{m} = \frac{F_{prop.}^{\beta}(\delta_*)+F_g^{\beta}+F_{inflow}^B}{m} \tag{4.47}$$

When we write the translational acceleration in vector form we will note that the force generated by the propellers $F_{prop.}^{\beta}$ and the induced inflow would be placed in the z-component only as shown.

$$\dot{V}^B = \begin{bmatrix}\dot{V}_x^B\\ \dot{V}_y^B\\ \dot{V}_z^B\end{bmatrix} = \begin{bmatrix}-\xi.g\\ \vartheta.g\\ \frac{F_{prop.}^{\beta}(\delta_*)+mg}{m}+\frac{I_f}{m}.V_Z^B\end{bmatrix} \tag{4.48}$$

The reason for placing these forces in the z-component is because they are directed always towards the z-direction of the body frame. The term $F_{prop.}^{\beta} + mg$ can be replaced by the term F_Z which is the net-upward force acted on the Quad-X frame as explained earlier in section 4.1 and figure 4.2.

So we have a complete linear system of equation which could be written collectively as:

$$\begin{matrix}\dot{x} = V_x^B\\ \dot{y} = V_y^B\\ \dot{z} = V_z^B\end{matrix} \quad and \quad \begin{matrix}\dot{V}_x^B = -\xi.g\\ \dot{V}_y^B = \vartheta.g\\ \dot{V}_z^B = \frac{F_Z}{m}+\frac{I_f}{m}.V_Z^B\end{matrix} \quad and \quad \begin{matrix}\dot{\xi} = \delta_{\xi}.\lambda_{\xi}\\ \dot{\vartheta} = \delta_{\vartheta}.\lambda_{\vartheta}\\ \dot{\varpi} = \delta_{\varpi}.\lambda_{\varpi}\end{matrix} \tag{4.49}$$

Using the above system of equations we can drive the system's state space model which is the main focus of the coming section.

4.4 State Space Derivation

The main aim of this section is to convert the equations derived in section 4.3 of the Quad-X to their respective state space form as the controller which we will be implementing requires the state space model for its utilization. First the linearized set of equations is converted to state space in continuous time domain and then using MATLAB it model is discretized.

4.4.1 System States

Although a lot of variables were available but for simplicity we chose the following state and input vectors.

$$x_s = [x \quad y \quad z \quad \vartheta \quad \xi \quad \varpi \quad V_x^B \quad V_y^B \quad V_z^B] \tag{4.50}$$

$$I_s = [\delta_\vartheta \quad \delta_\xi \quad F_Z \quad \delta_\varpi] \tag{4.51}$$

A more complex model could also have been derived by adding the translational acceleration terms in the state vector but it would have increased our work so we will stick to the above mentioned states. Here x, y, z are the positions of Quad-X centre of mass in 3D space. ϑ, ξ, ϖ are the 3-2-1 Euler angles and V_x^B, V_y^B, V_z^B are the linear velocities of the Quad-X in $x, y\ and\ z$ direction respectively. From now on we would refer equation 4.50 as the state vector throughout the rest of the chapter.

4.4.2 Matrices Derivation

In state space model we describe our system in the form of a differential equation. There are two equations in a state space model one related to the system states and the other related to the system inputs.

The general form of the state space model in continuous time domain is shown below in equations 4.52 and 4.53.

$$\dot{x}_s(t) = A_c.x_s(t) + B_c.I(t) \tag{4.52}$$

$$y(t) = C_c.x_s(t) + D_c.I(t) \tag{4.53}$$

Where,

$x_s(t) = system\ states$

$\dot{x}_s(t) = state\ derivatives$

$A_c = state\ transition\ matrix\ in\ continuous\ time\ domain$

$B_c = input\ matrix\ in\ continuous\ time\ domain$

$I(t) = system\ inputs$

$y(t) = system\ outputs$

$C_c = output\ matrix\ in\ continuous\ time\ domain$

$D_c = Direct\ transmission\ matrix\ in\ continuous\ time\ domain$

By arranging the above linearized system of equations we can extract the following A_c, B_c, C_c matrices. The direct transmission matrix D_c is zero because there is no direct transmission of system inputs to system outputs.

$$A_c x_s(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_f/m \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \vartheta \\ \xi \\ \varpi \\ V_x^B \\ V_y^B \\ V_z^B \end{bmatrix} \tag{4.54}$$

$$B_c I(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \lambda_\varpi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{m} \\ 0 & 0 & 0 & 0 & \lambda_\xi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_\vartheta & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \begin{bmatrix} \delta_\vartheta \\ \delta_\xi \\ F_Z \\ \delta_\varpi \end{bmatrix} \tag{4.55}$$

$$C_c x_s(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \vartheta \\ \xi \\ \varpi \\ V_x^B \\ V_y^B \\ V_z^B \end{bmatrix} \tag{4.56}$$

The first task of driving the continuous state space model from the linearized system model has been achieved and in the coming section this continuous time domain model is discretized for controller implementation purpose.

4.4.3 Continuous to Discrete Conversion

Another restriction for the implementation of our controller is that our state space model should be in discrete time domain. Let’s start with the continuous state space model derived in the previous section and convert it to discrete form.

$$\dot{x}_s(t) = A_c . x_s(t) + B_c . I(t) \tag{4.57}$$

$$y(t) = C_c . x_s(t) + D_c . I(t) \tag{4.58}$$

As we know that equation 4.52 is a simple first order non-homogeneous differential equation and its general solution is given by:

$$x_s(t) = e^{A(t-t_0)} x_s(t_0) + \int_{t_0}^{t} e^{A(t-\tau)} . B_c . I(\tau) d\tau \tag{4.59}$$

Now we can convert equation 4.59 from its continuous time representation to discrete time representation by simply replacing $(t_0 = kT \ and \ t = kT + T)$.

Rewriting equation 4.59 we get the following new equation as:

$$x_s(kT+T) = e^{AT}x_s(kT) + \int_{kT}^{kT+T} e^{A(kT+T-\tau)}.B_c.I(\tau)d\tau \tag{4.59}$$

What we are doing above is simply using zero order hold (ZOH) in the above expression in which the value of the continuous response is taken after a specified time interval kT and this value remains constant for the whole interval until the next time inter $kT + T$ is reached. The input function becomes:

$$I(\tau) = I(kT),\ kT \leq \tau \leq kT + k \tag{4.60}$$

For simplicity we will introduce a dummy or intermediate term to make the system less complex.

$$\nexists = kT + T - \tau \tag{4.61}$$

Inserting $\nexists$ in equation 4.59 and rewriting it as:

$$x_s(kT+T) = e^{AT}.x_s(kT) + \left(\int_0^T e^{A\nexists}d\nexists\right).B_c.I(kT) \tag{4.62}$$

Looking at the above equation 4.62 we can obtain $A_d, B_d \ and \ C_d$ matrices as follows:

$$A_d = e^{AT}, B_d = \left(\int_0^T e^{A\nexists}d\nexists\right).B_c \ and \ C_d = C_c \tag{4.63}$$

Now we can easily construct the state space model of Quad-X in discrete time domain as shown equations 4.64 and 4.65.

$$x_s(k+1) = A_d.x_s(k) + B_d.I(k) \tag{4.64}$$

$$y(k) = C_d.x_s(k) \tag{4.65}$$

As the output is not directly affected by the inputs, this makes the matrix D_d zero and hence it is omitted in equation 4.65.

For the computation of the above equation 4.63 we will use the built-in function of MATLAB $c2d(sys, T, 'zoh')$. The function takes three parameters, system state space model in continuous time domain "sys", the sampling time "T" and the method of sampling which in our case is simply zero order hold in which the value between the two samples is considered constant and is equal to the value of the first point out of the two. The discrete state space matrices that we obtain after the implantation of the above mentioned function are:

$$A_d = \begin{bmatrix} 1.000 & 0 & 0 & 0 & -0.0005 & 0 & 0.01 & 0 & 0 \\ 0 & 1.000 & 0 & 0.0005 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 1.000 & 0 & 0 & 0 & 0 & 0 & 0.01 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.0981 & 0 & 1.000 & 0 & 0 \\ 0 & 0 & 0 & 0.0981 & 0 & 0 & 0 & 1.000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9914 \end{bmatrix} \quad (4.54)$$

$$B_d = \begin{bmatrix} 0 & -1.5136x10^{-9} & 0 & 0 \\ 1.8917x10^{-9} & 0 & 0 & 0 \\ 0 & 0 & 1.225510^{-4} & 0 \\ 1.157410^{-5} & 0 & 0 & 0 \\ 0 & 9.261210^{-6} & 0 & 0 \\ 0 & 0 & 0 & 1.647610^{-5} \\ 0 & -4.541010^{-7} & 0 & 0 \\ 5.675110^{-7} & 0 & 0 & 0 \\ 0 & 0 & 2.447610^{-2} & 0 \end{bmatrix} \quad (4.55)$$

$$C_d = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.56)$$

4.5 Chapter Summery

In this chapter modelling of Quad-X has been carried out and a viable state space model is derived. The model being non-linear is first simulated for a specific set of input and then it is linearized using First Order Taylor Series approximation. The linear model achieved is then converted from continuous to discrete time domain which is the requirement of our controller.

The main focus of the next chapter is to use this derived stat space model and derive a viable system control.

CHAPTER 5

CONTROLLER DESIGN

As we have seen in the previous chapter that the Quad-X model is non-linear and time-variant as the battery reduces with fly time. However for this project we are considering two design approaches for the controller one is stable hovering of the Quad-X and second is constant reference tracking. The key objective of this chapter is to present a complete procedure and the necessary design steps taken in developing the controller.

The controller chosen for the stabilization of the Quad-X model is the linear state feedback controller due to its unique property of handling MIMO systems efficiently with great accurately.

5.1 State Feedback

The word feedback describes the property of any system to check if the required outputs of the system are met or not and correcting them at the same time by linking this feedback with the input. Using feedback we can design controllers that can make our system robust even in the presence of external disturbance or any irregularity within a system generated due to any reason.

When we present a system in state space form the state vector contains all the information needed to compute the future behaviour of the system due to which our controller becomes memory-less. The only assumption that we have to make now is that all the states at the time *t* are measureable or can be estimated using different techniques.

So in general we can write the desired controller which is a function of the states at time *k* as:

$$I(k+1) = f(x(k), k) \tag{5.1}$$

Where,

$I(k+1) = inputs\ at\ time\ k+1$

$x(k) = states\ at\ time\ t$

We can derive a controller from the above equation 5.1 in which the inputs at time *k* +1 are linear combination of the states at time *k* as shown below.

$$I(k+1) = -K(k).x(k); \quad K \in R^{mxn} \tag{5.2}$$

Where,

$K(k) = controller\ gain\ at\ time\ k$

For time invariant models the controller designed is also time invariant i.e. $K(k) = K$ which means that the controller is just a multiplication gain of constant values and in our case we are going to consider this and our main goal of the remaining chapter will be to calculate that gain.

The linear feedback can also take the following form with reference considered.

$$I(k+1) = -K.x(k) + K_r.r(k); \quad K \in R^{mxn}, K_r \in R^{mxr} \tag{5.3}$$

Where,

$K_r = constant\ gain$

$r(k) = reference\ states\ at\ time\ k$

The open loop dynamics is given by the following equations 5.4 and 5.5 and figure 5.1 as:

$$x(k+1) = A_d.x(k) + B_d.I(k) \tag{5.4}$$

$$y(k) = C_d.x(k) \tag{5.5}$$

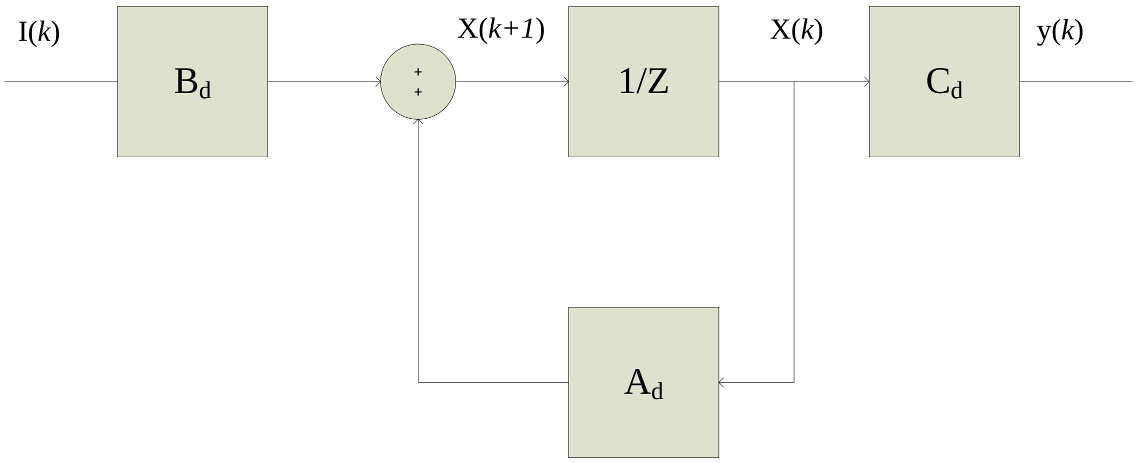

Fig 5.1: Open Loop

Now by creating the close loop the state transition matrix (A_d) changes to $(A_d - B_d.K)$. Now for the desired behaviour of the close loop the only thing now required is the careful making of the gain matrix K.

Now the close loop dynamics are shown in the following equations 5.6 and 5.7 and in figure 5.2 as:

$$x(k+1) = (A_d - B_d.K).x(k) + K_r.r(k) \tag{5.6}$$

$$y(k) = C_d.x(k) \tag{5.7}$$

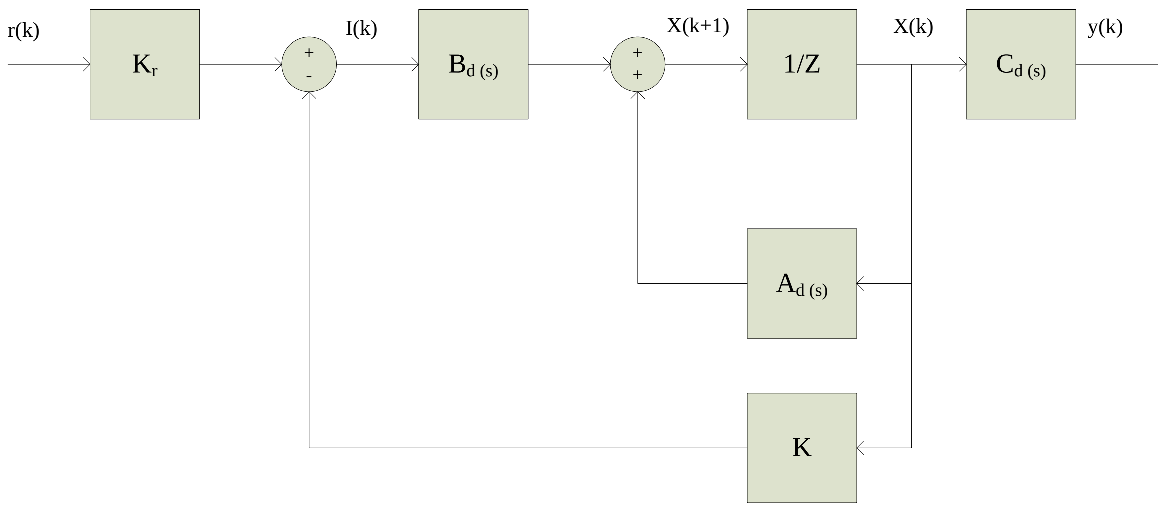

Fig 5.2: Close Loop with State Feedback

5.1.1 State Feedback Methods

Determining a controller for any system is same as determining a suitable feedback gain matrix K. Now for SISO systems we can design a controller purely based on the positions of the closed loop poles. But as we know that it can be quiet hectic to find the right set of pole values because even after finding the right set of poles we may encounter a situation where the control signal is outside the physical limits that we can actually produce even for small or reasonable disturbance and reference signals.

Now if we talk about the MIMO systems like the one under study the model of Quad-X it becomes even more hectic and tedious to relate the elements of the gain matrix K to that of the close loop poles. The parameters are more in number then the constraints and an under-determined system has to be solved which means there is no unique solution K for a set of poles, and to determine the optimum value is not trivial.

For systems like the model presented here of the Quad-X, Linear-Quadratic methods are more suitable which falls under the category of Optimum control. Optimum Control gives provides us with a design technique which instead of putting effort in finding the positions of poles focuses on the minimization of a certain performance function. We will see in the coming sections that it is basically a systematic method for finding an optimum value of the state feedback gain matrix.

5.2 Linear Quadratic Methods

5.2.1 Optimum Control Problem

As explained earlier that in optimum control we find a control law I by minimizing a certain function. This function is known as the cost function Ψ of the states and control variables and our job is to balance this cost function in a meaningful manner.

The general expression for the non-linear, discrete, optimum control problem is as follows:

$$x(k+1) = f(x(k), I(k), k) \tag{5.8}$$

The performance function or index of performance **ί** is the sum of performance and cost criterion *Ψ(k)* over time horizon *N*, which starts form k_0=0.

We can further divide this in to two different problems:

1. Finite time horizon *N*.

$$\iota = \sum_{k=0}^{N-1} \Psi(x(k), I(k), k) + \omega(x(k)) \tag{5.9}$$

2. Infinite time horizon *N*=∞.

$$\iota = \sum_{k=0}^{\infty} \Psi(x(k), I(k), k) \tag{5.10}$$

Our key purpose is to optimize the above performance function with the sequence of control inputs *I*. The final optimized performance index will depend on the system dynamics *f* and the initial states x_0 and the inputs. For the case of finite horizon, there is an additional term as we can see in equation 5.9 ω which basically puts a gain at the state error for the final time step.

5.2.2 LQ Control for Discrete Time Systems

We will start with the following linear model which we derived in the previous chapter.

$$x(k+1) = A_d.x(k) + B_d.I(k); \quad A_d \in R^{nxn}, B_d \in R^{nxm} \tag{5.11}$$

For this project we will drive the controller matrix from the cost function as discussed above and in our case we will consider that our performance function **ί** which is a function of *Ψ* is quadratic in *I(k)* and *x(k).* In general several other cases could also be considered and some of them are in fact explored in the current research.

Our performance function will take the following form:

$$\iota = \sum_{k=0}^{N-1} [x(k)^T.Q_1.x(k) + I(k)^T.Q_2.I(k)] + x(N)^T.Q_N.x(N) \tag{5.12}$$

$$\iota = \sum_{k=0}^{\infty} [x(k)^T.Q_1.x(k) + I(k)^T.Q_2.I(k)] \tag{5.13}$$

The matrices Q_1 and Q_2 are symmetric and they are the parameters of our control problem. For the complex cases these matrices could be time-variant, but we will stick to constant matrices not changing with time as the time-variant case is beyond the scope of the project. Further in the linearization problem two constraints are used, one is that the state weight matrices Q_1 and Q_N are positive-semi definite, and the actuation weight matrix Q_2 is strictly positive definite, which means that the weights in of the diagonal states could be zero, but the command scaling weights are always positive.

The above constraint helps us in making the controller a linear function of the states and after optimization we will achieve linear state feedback solution.

$$I^{*}(k) = -L(k).x(k) \tag{5.14}$$

Where,

$I^{*}(k)$ *indicates the command sequence that optimizes the performance function.*

The LQ state feedback gain matrix from now on will be denoted by L.

As our performance function is of the quadratic form, for optimization purpose we need to minimize this function *ï=min(í(k))*. The solution will be dependent on the state dynamics and as defined by the calculated constant real matrices *(A_d, B_d)*, and the initial states x_0. After solving the optimization problem the result which we would be interesting in would be the gain matrix *L(k)* and the command sequence $I^{*}(k)$, while the actual value of the performance index after minimization *ï=min(í(k))* would be less interesting.

When we calculate the state feedback gain matrix *L(k)* using the finite horizon equation we get time-variant matrix but for practical situations where operation time is undetermined as in our case constant gain matrix is preferred as the flight time is not determined and our only objective is to make the Quad-X hover irrespective of the flight time. For this we will use the infinite horizon optimization strategy that will produce time-invariant controller *L*.

5.2.3 Infinite Horizon Optimization

The solution of Infinite horizon controller in used in this project in which the performance function is an infinite sum of only the positive terms, due to its quadratic nature and for this sum to minimize it must converge. So for this performance function to converge the optimum controller sequence $I^{*}(k)$ will drive the performance and cost criterion to zero. Also the states *x(k)* and commands *I(k)* must also tend to zero because the matrices Q_1 and Q_2 are constant and positive definite.

$$\ddot{\imath}(x_0) = min_I\big(\acute{\imath}(I)\big) = min_I \sum_{k=0}^{\infty} \Psi(k) = constant \tag{5.15}$$

$$\lim_{k\to\infty} \Psi(k) = \lim_{k\to\infty}\big(x^T(k).Q_1.x(k) + I^T(k).Q_2.I(k)\big) = 0 \tag{5.16}$$

Where,

Q_1 *and* Q_2 *are constant matrices in equation* 5.16

$$\lim_{t\to\infty} x(k) = 0;\ \lim_{t\to\infty} I(k) = 0 \tag{5.17}$$

The most important thing here is to specify the correct system model in order to map the problem presented here which means that in hover state our Quad-X model must fulfil the condition described in equation 5.17. We will see in the coming sections of the chapter that after the introduction of the reference signal we will be able to meet this condition because the quantities being weighted in the performance index are three positions and three angles. Now for position the performance index will minimize the difference: x-r_x, y-r_y, z-r_z and in this form it will satisfy the condition given in 5.17. In hovering state the Roll, Pitch and Yaw commands are naturally zero or very close to zero as the Quad-X will not be making any angular movements ideally. One more important factor is the thrust force which will not be zero rather it will have some non-zero constant value. So in order to cater for this problem we will not take in account the thrust force but the net vertical upward force F_Z. The net upward force in hover is zero so in this way the condition of equation 5.17 will be satisfied.

The general solution to infinite horizon problem guaranties a unique input command sequence solution I^* to the optimization provided certain conditions are met like (A_d, B_d) are controllable and $Q_2>0$ and (A_d, Q_2) is observable. These conditions are sufficient but more relaxed criteria may exist.

Moreover the optimizing command sequence $I^*(k)$ is described by just a constant state feedback matrix $L(k) = L$, and can be proved that it is stable in close loop. Thus a single matrix can easily be computed offline and used for an infinite time horizon of control which will bring our system from an initial arbitrary state to zero with minimum cost.

This time-invariant feedback matrix obtained as a result of infinite horizon optimization problem is the main focus of the rest of the chapter.

5.3 Linear Quadratic Algorithm

The method used to find the state feedback static gain is to first formulate the expression for time varying LQ-controller using the finite-time horizon algorithm and backward induction principle described in the next session which is known dynamic programming. After that we put N=∞ and perform iterations for a variable number of time which depends upon the convergence of the values of the required controller matrix.

Step by step the process of derivation the expression will be explained which is used in the final algorithm and its pseudo-code is also given. The algorithm is based on the optimal control notes [8] and the description of the Standard Regulator problem for discrete systems in [6]. Furthermore, reference is introduced and the system changes needed to cope with it are explained and some guidelines for selecting Q_1 and Q_2 for optimum results are discussed.

5.3.1 Dynamic Programming

Dynamic programming is basically a procedure of breaking complex function into series of simpler functions easy to solve.

The optimization process takes the advantage of this fact, that:

Suppose that we have derived a control strategy which is optimal in the interval [0; N] it must also be optimal in any interval [k; N] with 0 ≤ k ≥ N.

It makes sense because if we are able to improve the efficiency of our controller in the interval $[k;\ N]$ this would also improve the efficiency of the controller in the entire interval $[0;\ N]$.

So based on the above assumption we will split our optimization and introduce the following notation;

$$\text{ἰ}_0^N = \sum_{k=0}^{N} \Psi(x(k), I(k)) \tag{5.18}$$

Since the performance function is dependent only on the initial state vector and sequence of control signals so the above equation can be modified as:

$$\text{ἰ}_0^N = \text{ἰ}_0^N(x(0), I(0), I(1), I(2), \dots, I(N)) \tag{5.19}$$

This makes sense because the next state say $x(1)$ is determined by the previous state $x(0)$ and the sequence of inputs like $I(0)$ etc.

Now for the minimum obtainable performance function we will write:

$$\text{ϊ}_0^N\big(x(0)\big) = min_{I(0),\dots,I(N)} \text{ἰ}_0^N(x(0), I(0), I(1), I(2), \dots, I(N)) \tag{5.20}$$

Now introducing the dynamic programming part which was stated at the beginning of the section, the performance function in the sub interval $[k;\ N]$ is stated as:

$$\text{ἰ}_k^N = \sum_{j=k}^{N} \Psi(x(j), I(j)) \tag{5.21}$$

$$\text{ἰ}_k^N = \text{ἰ}_k^N(x(k), I(k), I(k+1), \dots, I(N)) \tag{5.22}$$

And similarly to equation 5.20 we will write the minimum obtainable performance function for the above equation 5.22 as:

$$\text{ϊ}_k^N\big(x(k)\big) = min_{I(k),\dots,I(N)} \text{ἰ}_k^N(x(k), I(k), I(k+1), \dots, I(N)) \tag{5.23}$$

$$\text{ϊ}_k^N\big(x(k)\big) = min_{I(k),\dots,I(N)} \sum_{j=k}^{N} \Psi(x(j), I(j)) \tag{5.24}$$

$$\text{ϊ}_k^N\big(x(k)\big) = min_{I(k),\dots,I(N)} \Big[\Psi\big(x(k), I(k)\big) + \sum_{j=k+1}^{N} \Psi(x(j), I(j))\Big] \tag{5.25}$$

$$\text{ϊ}_k^N\big(x(k)\big) = min_{I(k)} \Big[\Psi\big(x(k), I(k)\big) + min_{I(k+1),\dots,I(N)} \sum_{j=k+1}^{N} \Psi(x(j), I(j))\Big] \tag{5.26}$$

$$\text{ϊ}_k^N\big(x(k)\big) = min_{I(k)} \Big[\Psi\big(x(k), I(k)\big) + \text{ϊ}_{k+1}^N\big(x(k+1)\big)\Big] \tag{5.27}$$

We see that using simple substitution and manipulation we calculated the above expression which shows that if we have the performance function in the interval $[k;\ N]$, the performance function in the interval $[k-1;\ N]$ can be calculated. So in order to calculate the performance function in the interval $[0;\, N]$, we need to calculate all the terms backwards from $\acute{\imath}_N^N\big(x(N)\big)\ to\ \acute{\imath}_0^N\big(x(0)\big)$ according to the following algorithm.

STEP 0:

$$\ddot{\imath}_N^N\big(x(N)\big) = \Psi(x(N), I(N)) \tag{5.28}$$

$I(N)$ is mostly zero as it is the last command input and does not affect any state vector and any non-zero value will just increase the performance function.

STEP 1:

$$\ddot{\imath}_{N-1}^N\big(x(N-1)\big) = min_{I(N-1)}\big[\Psi\big(x(N-1), I(N-1)\big) + \ddot{\imath}_N^N\big(x(N)\big)\big] \tag{5.29}$$

This is written in accordance with the equation 5.27 and a minimized command signal $I^*(N-1)$ is obtained.

STEP i:

$$\ddot{\imath}_{N-i}^N\big(x(N-i)\big) = min_{I(N-i)}\big[\Psi\big(x(N-i), I(N-i)\big) + \ddot{\imath}_{N-i+1}^N\big(x(N-i+1)\big)\big] \tag{5.30}$$

Extending our approach used in step 1 and through iteration calculating the performance function and command signal $I^*(N-i)$

STEP N:

Finally the performance function in the interval $[0; N]$ will be obtained as:

$$\ddot{\imath}_0^N\big(x(0)\big) = min_{I(0)}\big[\Psi\big(x(0), I(0)\big) + \ddot{\imath}_1^N\big(x(1)\big)\big] \tag{5.31}$$

So, we can calculate the whole command signal from $I^*(N)$ to $I^*(0)$.

Thus by using this algorithm, just by knowing the initial states we can completely calculate the command sequence from $I^*(0), \dots, I^*(N)$ which will bring the plant from its initial states to zero. Although this method could also be applied to the open-loop systems but due to disturbances the results may not be very pleasing. A close-loop control would have good results. The next section is about the calculation of the feedback matrix gain.

5.3.2 Matrix Algebra

Assuming that we have enough knowledge about our system and about the performance index we shall start this section with the following equation which is nothing but equation 5.27:

$$\ddot{\imath}_k^N\big(x(k)\big) = min_{I(k)}\big[\Psi\big(x(k), I(k)\big) + \ddot{\imath}_{k+1}^N\big(x(k+1)\big)\big] \tag{5.31}$$

Now we will make two simple assumptions:

$$I(k) = -L(k).x(k) \tag{5.32}$$

$$and \quad \ddot{\imath}_k^N\big(x(k)\big) = x^T(k).S(k).x(k) \tag{5.33}$$

The first assumption is simple but for the second assumption it needs some explanation. As we have already seen that the performance function was originally quadratic function of the states and the inputs, but actually it can be just written as a function of states only because we shall see that when we minimize our performance function with respect to input *I* we will derive an expression for input sequence I^* and back substitute it in the original performance function there by elimination the input and the performance function then will completely become function of states as shown in equation 5.33.

We will insert equation 5.33 in 5.31 for the expression $\ddot{\imath}_{k+1}^N\big(x(k+1)\big)$ and search for the control signal *I(k)* which will give us the minimal performance index $\ddot{\imath}_k^N\big(x(k)\big)$:

$$\ddot{\imath}_k^N\big(x(k)\big) = min_{I(k)}[x^T(k).Q_1.x(k) + I^T(k).Q_2.I(k) + x^T(k+1).S(k+1).x(k+1)] \tag{5.34}$$

$$\ddot{\imath}_k^N\big(x(k)\big) = min_{I(k)}[x^T(k).Q_1.x(k) + I^T(k).Q_2.I(k) + (A_d.x(k) + B_d.I(k))^T.S(k+1).(A_d.x(t) + B_d.I(k))] \tag{5.35}$$

We can make the above expression simpler by removing the k argument from the input and the states vector but we will use it with the S matrix because that is the matrix which is under consideration at the moment.

$$\ddot{\imath}_k^N(x) = min_I[x^T.Q_1.x + I^T.Q_2.I + (A_d.x + B_d.I)^T.S(k+1).(A_d.x + B_d.I)] \tag{5.36}$$

Now we will differentiate the above performance function with respect to *I* and find the optimal control signal I^* using the matrix derivative [9, page 10, eq. 70] as shown:

$$\frac{\partial \ddot{\imath}_k^N(x)}{\partial I} = Q_2.I + Q_2^T.I + 2.B_d^T.S(k+1).(A_d.x + B_d.I) \tag{5.37}$$

$$= 2.Q_2.I + 2.B_d^T.S(k+1).(A_d.x + B_d.I) = 0 \tag{5.38}$$

The optimal control signal I^* can only be found iff $[Q_2 + B_d^T.S(k+1).B_d]$ is invertible. Reintroducing the argument k our expression for optimum control would become:

$$I^*(k) = -[Q_2 + B_d^T.S(k+1).B_d]^{-1}.B_d^T.S(k+1).A_d.x(k) \tag{5.39}$$

Now comparing equation 5.32 and 5.39 we get the proportionality matrix $L(k)$:

$$L(k) = [Q_2 + B_d^T.S(k+1).B_d]^{-1}.B_d^T.S(k+1).A_d \tag{5.40}$$

Next we will insert $I^* = -L.x$ in the expression of the performance index $\ddot{\imath}_k^N(x)$ and obtain:

$$\ddot{\imath}_k^N(x) = x^T.Q_1.x + (-Lx)^T.Q_2.(-Lx) + (A_d.x - B_dLx)^T.S(k+1).(A_d.x - B_dLx) \tag{5.41}$$

After some manipulation we will get:

$$\text{ï}_k^N(x) = x^T[Q_1 + L^T.Q_2.L + (A_d - B_dL)^T.S(k+1).(A_d - B_dL)]x \tag{5.42}$$

$$\text{ï}_k^N\big(x(k)\big) = x^T.S(k).x \tag{5.43}$$

So we have proved our assumption that the performance function is quadratic in $x(k)$ i.e. the states of our model. So we can formulate a recursive expression for the $S(k)$ matrix by simply comparing equation 5.42 and 5.43 as shown.

$$S(k) = Q_1 + L^T(k).Q_2.L(k) + (A_d - B_dL(k))^T.S(k+1).(A_d - B_dL(k)) \tag{5.44}$$

We shall now derive the simpler expression for the $S(k)$ matrix which would be non-negative and definite because $Q_1$is non-negative and definite.

For the derivation of $S(k)$ we will omit the argument k again and insert the value of $L(k)$.

$$S(k) = Q_1 + L^TQ_2L + (A_d - B_dL)^T\ S(k+1)\ (A_d - B_dL) \tag{5.45}$$

Now by inserting the value of $L(k)$ and simplifying we get the simple final form for $S(k)$ as:

$$S(k) = Q_1 + A_d^T.S(k+1).[A_d - B_d.L(k)] \tag{5.46}$$

The expressions for $S(k)$ and $L(k)$ calculated so far are time varying and mostly time invariant matrices are preferred which are $L(0)$ and $S(0)$ by taking $N \to \infty$.

Which means that $L(0) = L(k) = L(k+1) = L$ and this steady state value of L will correspond to the steady state value of $S(0) = S(k) = S(k+1) = S$.

So our equations will simplify further to the following final form:

$$L = [Q_2 + B_d^T.S.B_d]^{-1}.B_d^T.S.A_d \tag{5.47}$$

$$S = Q_1 + A_d^T.S.[A_d - B_d.L] \tag{5.48}$$

The above equation is the Algebraic Riccati equation. Solving it will yield the matrix S which intern will give us the matrix L which is actually required for this project.

5.3.3 MATLAB Code for Calculating L Matrix

Following is the MATLAB code that was implemented for the extraction of the static feedback gain matrix:

```
While (norm (conv) > 10e − 10);
  L = (Q2 + B_D.' * S * B_D)^(−1) * B_D.' * S * A_D;
  S_next = Q1 + A_D.' * S * (A_D − B_D * L);
  Conv = S − S_next;
  $S = S\_next;$
  $norm\ (conv)$
end
```

5.3.4 Weight Matrices Selection

The only design parameters for the implementation of our control algorithm which is the ***LQ-method*** are the two time-invariant matrices Q_1 and Q_2. Although the number of parameter from which we have to select become very large i.e. $m^2 + n^2$ but as a reward we will get a unique controller gain $\boldsymbol{L}$. However if we limit the matrices to diagonal matrices we can reduce the no of choices to just $m + n$ which is a huge reduction.

As described earlier while describing the performance index that the weight matrices are quadratic and symmetric. Q_1 and Q_N were positive semi-definite (nxn) and Q_2 was positive definite(mxm). We also noted that Q_1 affected the current state, Q_N the final state and Q_2 the input state.

The problem for selecting the appropriate weights for a given controller design problem is naturally up to a system's engineer, and mainly depends upon the system's dynamics. After the careful selection of the weight matrices we need to solve the Riccati equation and find the L-matrix and test it either by simulation or practically.

After testing we shall note if some states $x_i(k)$ has a slow/fast response then we can adjust it by increasing/decreasing the weight matrix $Q_1(i,i)$ and on the other hand if the input signal $I_j(k)$ become too large/small and fall out of our practical range then we can adjust it by increasing/decreasing the weight matrix $Q_2(j,j)$.

One of many techniques used for the purpose of finding the gain matrices is the ***Bryson's Rule*** which states that acceptable value of weights for the states $x(k)$ and control input $I(k)$ would be to choose the diagonal matrices Q_1 and Q_2 as:

$$Q_1(i,i) = \frac{1}{x_{i\ ,max}^2} \tag{5.49}$$

$$Q_1(i,i) = \frac{1}{I_{j\ ,max}^2} \tag{5.50}$$

Where $x_{i,max}$ is the maximum acceptable value for $x_i(k)$ and $I_{j,max}$ is the maximum acceptable value for $I_j(k)$.

5.3.5 Reference Tracking

The controller designed and calculated in the above sections was for the purpose to bring our system from an initial disturbed state to zero state. Suppose if we want to track a certain reference signal like step, ramp, sin or any other signal we need to model the controller accordingly and obtain a suitable gain ***L***. Different procedures for calculating the controller gain to track such reference along with disturbances are discussed in detail in [12, chapter 5, 6 and 7]. For our project we only selected to follow constant reference (step reference) and it has been proved in [8] that the gain matrix ***L*** that we calculated to bring the states back to zero can be used to force the system to follow the step reference as the model does not affect the value of ***L***. the block diagram with reference signal becomes as follows:

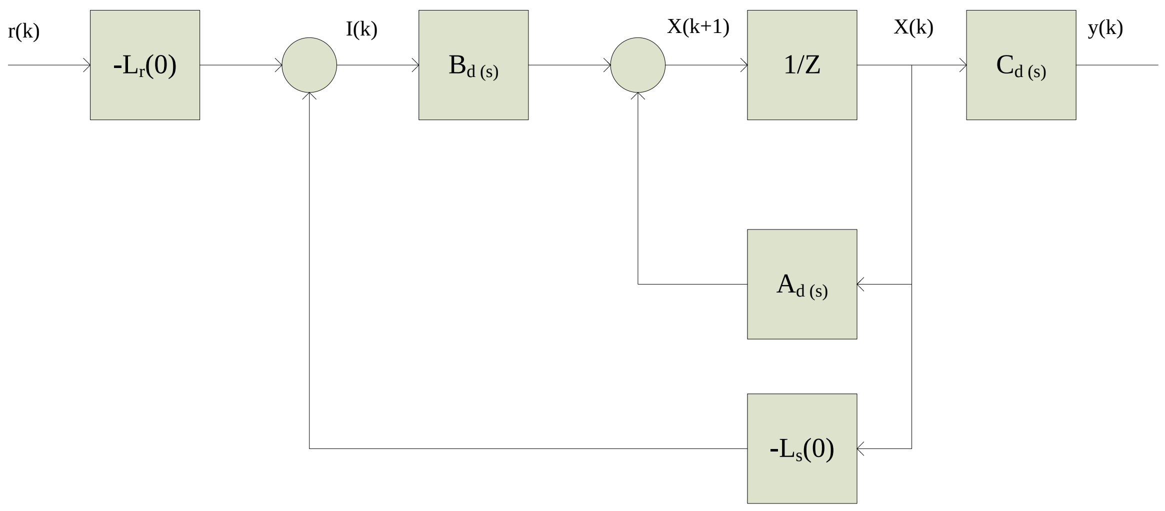

Fig: 5.3: Block Diagram of Optimal Control System With Constant Reference

5.4 LQ Implementations and Simulations

The controller is calculated offline using the MATLAB code as given in the previous section and the obtained controller matrix is used in the Simulink model of our Quad-X and offline results are generated. The Simulink model is shown below in figure 5.4.

Now we will simulate few results offline to check the performance of our controller.

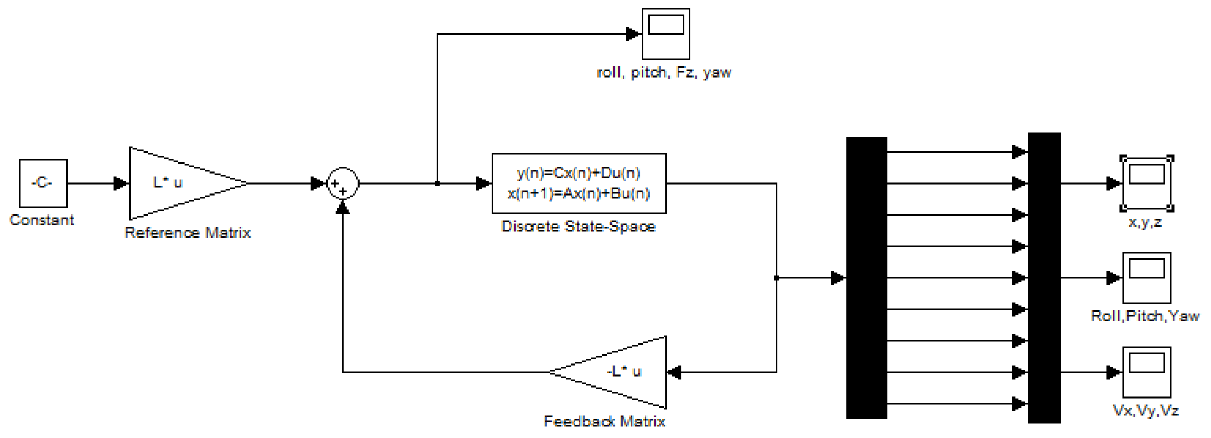

Fig 5.4: Linear-Discrete Simulink model of Quad-X with Constant Reference

5.4.1 Simulation Results

The offline simulations were run for reference states x_r = [0 0 0 0 0 0 0 0 0] and initial state $x_{0=}$[1 0 0 0 0 0 0 0 0].

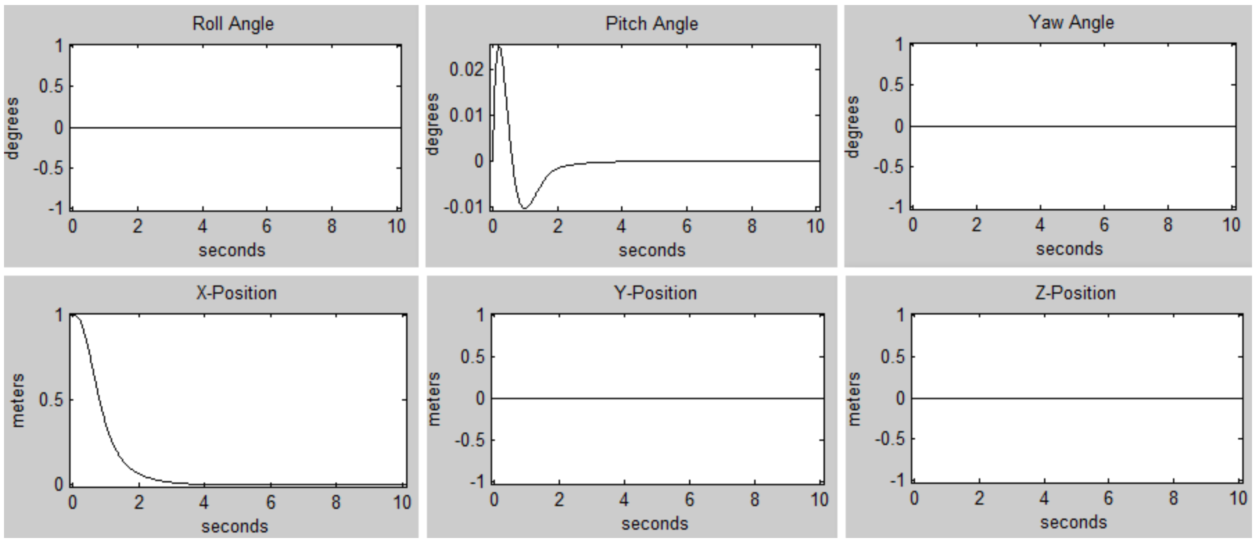

Fig 5.5: Step on X-Axis

The above simulations show that our controller brings the Quad-X from an initial disturbed position in X-Axis to the origin in less than 4 seconds.

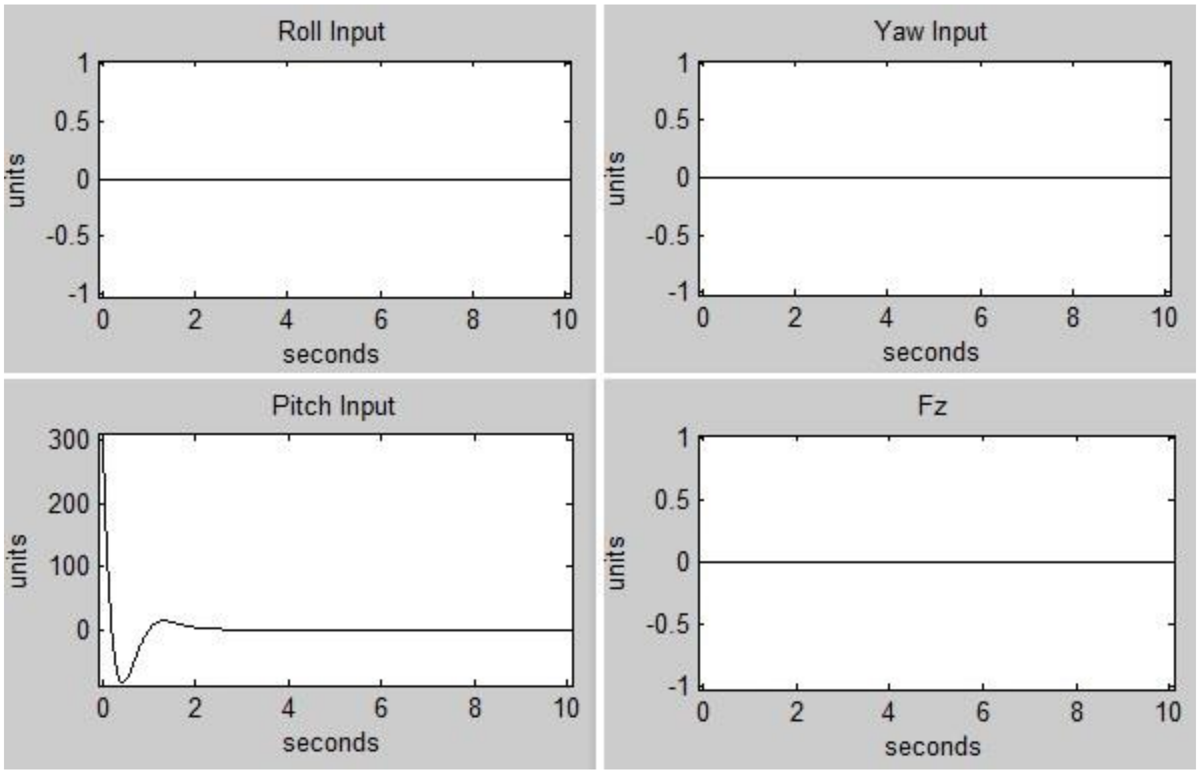

Fig 5.6: Generated Control Inputs

These are the control inputs generated to bring it to the initial state.

Let’s see another example for a constant reference tracking. The initial position of the Quad-X aircraft is $x_0 = [0\ 0 - 2\ 0\ 0\ 0\ 0\ 0\ 0]$ and reference signal $x_r = [0\ 0 - 1\ 0\ 0\ 0\ 0\ 0\ 0]$. The output graphs are as follows:

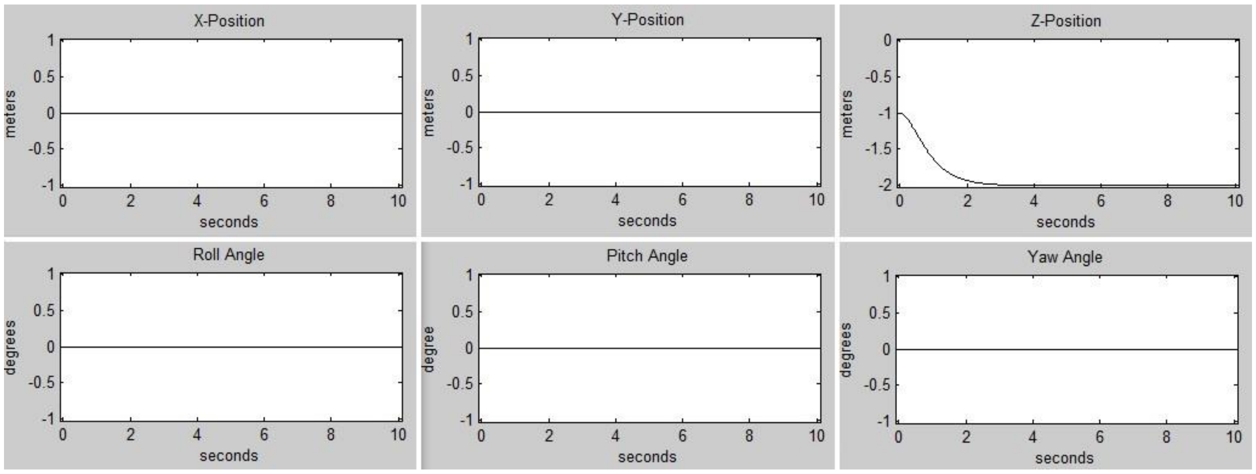

Fig 5.7: Step on Z-Axis

The above graphs illustrates that in less than four seconds the Quad-X moves from -1 to -2 meters height vertically upward as positive Z-Axis is in the downward direction.

The control sequence needed for the Quad-X to perform this manoeuvre is depicted in the following fig 5.8.

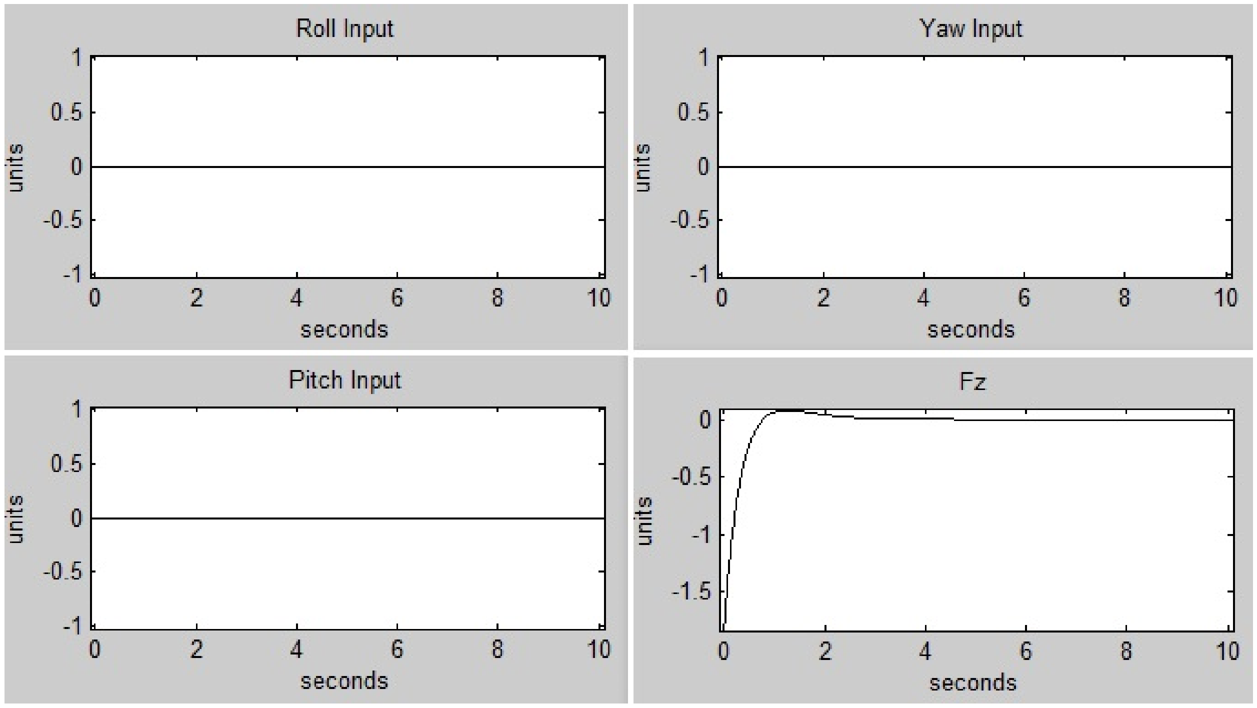

Fig 5.8: Control Sequence for Z-Axis

As a reference the truncated values of the weight matrices Q_1 and Q_2 and the feedback and reference gains are given here.

$$Q_1 = \begin{bmatrix} x & y & z & roll & pitch & yaw \\ 2.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 82.054 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} roll & pitch & F_z & yaw \\ 2.5e-5 & 0 & 0 & 0 \\ 0 & 2.5e-5 & 0 & 0 \\ 0 & 0 & 0.73463 & 0 \\ 0 & 0 & 0 & 2.5e-5 \end{bmatrix}$$

$$L = \begin{bmatrix} & x & y & z & roll & pitch & yaw & V_x & V_y & V_z \\ roll & 0 & 303.61 & 0 & 7021.9 & 0 & 0 & 0 & 284.8 & 0 \\ pitch & -304.71 & 0 & 0 & 0 & 7991.7 & 0 & -295.97 & 0 & 0 \\ F_z & 0 & 0 & 1.8 & 0 & 0 & 0 & 0 & 0 & 1.36 \\ yaw & 0 & 0 & 0 & 0 & 0 & 1784.8 & 0 & 0 & 0 \end{bmatrix}$$

5.5 Chapter Summery

In this chapter a controller is designed which can perform two operations one is to bring the Quad-X model from any initially disturbed position to zero state and second is to make it follow a constant reference signal. The controller is actually a constant gain which is multiplied with the states of the Quad-X at time *t* and calculated the inputs for the next time step *t+1*. This controller is obtained by an iterative convergent technique with the help of MATLAB which guarantees the stability of the Quad-X model.

CHAPTER 6

CONCLUSION

This project was started with the initial aim to make the Quad-X frame, design a viable controller and implement it on the aircraft first to make it perform a stable hover and secondly to move the aircraft from one point to another i.e. to make it follow a constant reference input.

Due to shortage of time and as it was a solo project performed by me so all the objectives were not successfully achieved but the objectives achieved were considered enough for the successful completion of my MS Thesis.

The objectives achieved are as follows:

1. Quad-X model was successfully designed and built
2. A viable stat space model was built for the purpose of controller design
3. An LQ controller was successfully designed and simulated and the results were shown

6.1 Future Work

The future work includes the following points:

1. To implement this controller on the Quad-X frame
2. Design of more complex controller which can cater for external disturbances
3. Design of a controller which can follow dynamic paths like ramp input, sinusoidal input etc.

Bibliography:

[1] WOW hobbies.

http://www.wowhobbies.com/turboacex720multi-rotorpreorder-3-2-1-1-2-1.aspx

[2] ArduCopter_Quad

https://code.google.com/p/arducopter/wiki/ArduCopter_Quad

[3] AR Drone Quadcopter

http://labe.felk.cvut.cz/~tkrajnik/ardrone/

[4] Futaba

http://www.futaba-rc.com/systems/futk7000.html

[5] Claudia Mary, Luminita Cristiana Totu, Simon Konge Koldbæ. *Modelling and Control of Autonomous Quad-rotor*. AAU, 1st edition, 2010.

[6] John B. Moore and Brian D.O. Anderson. *Optimal Control: Linear Quadratic Methods*. Prentice Hall, 1990.

[7] T. Bak. *Lecture Notes - Modeling of Mechanical Systems*. AAU, 1st edition, 2002.

[8] Palle Andersen Ole Sø rensen. Optimal control, lecture notes. Department of Control Engineering, Institute of Electronic Systems, Frederik Bajers Vej 7, DK-9220 Aalborg, Denmark, 6. February 2010.

[9] Matrix cookbook by Kaare Brandt Petersen& Michael Syskind Pedersen. http://orion.uwaterloo.ca/~hwolkowi/**matrix**cookbook.pdf