Dynamics Modelling for A Quanser Helicopter

Riwash Tuladhar, Taman Deep Singh

Department of Mechanical Engineering, University of Windsor

MECH-4672: Dynamics and Control of UAVs

Dr. Afshin Rahimi

July 24, 2023

Table of contents

List of Figures

List of Tables

Abstract

The project explores the control of the Quanser three degree-of-freedom (DOF) helicopter, a scaled-down mechanical device emulating the flight dynamics of a rotor helicopter. The goal is to develop controllers and estimators to maintain a stable and safe flight of the helicopter without considering travel motion. The paper presents the nonlinear and linearized equations of motion for the helicopter and investigates its stability, controllability, observability, stabilizability, and detectability for the given range of Lp (roll damping coefficient) and Mq (pitch damping coefficient) individually through the use of MATLAB built-in functions. State-space models and various controllers, including LQR and LQ Servo, are designed and simulated based upon the range of Lp and Mp value that meets the requirement of previous list. The study also evaluates different control strategies to design and improve the system performance, such as Dynamic Output Feedback (DOFB) controllers and estimators. A simulink model was then created that connects to the unit and control each function. Overall, by performing a comprehensive understanding of the flight dynamics and control of unmanned aerial vehicles (UAVs), the importance of designing effective controllers to ensure the safety and stability of complex mechanical systems like the Quanser 3 DOF helicopter is studied.

Introduction

Based on MIT OCW 16.30 Lab 1, a three-degree-of-freedom Quanser helicopter system was actuated using two rotor speeds: Vcyc, an electric voltage to differentially change the two rotor speeds, and Vcoll, an electric voltage to control the collective speed of the two propellers (B. Zheng and Y. Zhong). The system's output was presented in three angles: roll φ, pitch θ, and travel ψ The goal was to ensure that the system is controlled. Various pedagogies learned throughout the semester was used to be able to control the behavior of the system. To achieve this, the process began with linearizing the nonlinear equation of motion derived from Newton's second law around a steady-state hover operating condition. By disregarding one of the travel dynamics (ψ), a state-space model was created. Using state-space techniques, specific controllers were designed: an LQR controller for roll, an LQ Servo controller for pitch, two closed-loop optimal estimators using LQE, and dynamic output feedback (DOFB). Next, the system's response was analyzed through various step simulations, accounting for the impact of small sensor noise on the DOFB controllers. Each simulation was implemented in Simulink and recorded. Through these studies, valuable insights and data were generated to understand and effectively control the behavior of the system.

Literature Review and Background

Web resource: MIT Open CourseWare, specifically, Feedback Control System 16.30 and Aircraft Stability and Control,16.333 as well as MIT OCW 16.30 Lab 1 was preliminary used to understand the scope of the project. Dr. Rahimi's Lecture (3-11) provided us with project understanding and in-depth analysis of flight dynamics modelling.

Problem Definition

The initial goal of this assignment is to use dynamic modeling techniques to derive both nonlinear and linear equations of motion for the Quanser 3 degree-of-freedom (DOF) helicopter which then later is used to design various control strategies. The series of specific tasks of the project include:

1. Applying the dynamic modeling approach to derive the nonlinear and linearized equations of motion for the Quanser 3-DOF helicopter.

2. Performing stability, controllability, observability, stability, and detectability analysis for the system based on the derived equations and given numerical parameters.

3. Designing a candidate LQR (Linear Quadratic Regulator) controller for the roll dynamics of the helicopter using state-space techniques. Simulating the system's response to a 20° step input and analyzing the performance.

4. Designing a candidate LQ Servo controller for the pitch dynamics of the helicopter using state-space techniques. Simulating the system's response to a 20° step input and analyzing the performance.

5. Designing two closed-loop optimal estimators using LQE (Linear Quadratic Estimator) for the median-value pitch dynamics and roll dynamics of the helicopter. Simulating the system's response to a 10° step input and analyzing the performance.

6. Designing dynamic output feedback (DOFB) controllers for the pitch and roll dynamics of the helicopter using state-space techniques. Simulating the system's response to roll and pitch step inputs and analyzing the performance, considering small sensor noise.

7. Putting the DOFB controllers into MATLAB/Simulink form and running simulations to record and analyze the step responses of the system.

For the roll dynamics (neglecting travel dynamics):

$$I_{xx}\ddot{\phi} = \tau_{cyc}l_h - mgl_\phi sin\phi - L_p\dot{\phi} - I_r\omega_{rotor}(\dot{\theta}\cos\phi + \dot{\psi}\sin\phi) \tag{1}$$

For the pitch dynamics (neglecting travel dynamics):

$$I_{yy}\ddot{\theta} = \tau_{coll} l_{boom} \cos\phi - Mgl_{\theta}\, sin(\theta + \theta_{rest}) - Dl_{boom} \sin\gamma + \; I_r \omega_{rotor} \dot{\phi} - M_q \dot{\theta} \qquad (2)$$

For the yaw dynamics:

$$I_{zz}\ddot{\psi} = \tau_{coll} l_{boom} \sin\phi - Dl_{boom} \cos\gamma \qquad (3)$$

$$\text{Given in th equation, we have,} \quad D = \; K_D \dot{\psi} \qquad (4)$$

$$\tau_{coll} = \; K_{\tau} \omega_{coll} - K_v \dot{\psi} \qquad (5)$$

$$\tau_{cyc} = \; K_{\tau} \omega_{cyc} - K_v \dot{\psi} \qquad (6)$$

$$\dot{\omega}_{cyc} + 6\omega_{cyc} = 780 V_{cyc} \qquad (7)$$

$$\dot{\omega}_{coll} + 6\omega_{coll} = 540 V_{coll} \qquad (8)$$

Please note that, D is the induced drag, τ_{cyc} is the cyclic thrust, ω_{rotor} is the constant rotational speed and K_D, and K_v is the drag and voltage coefficients respectively.

The Quanser 3 degree-of-freedom (DOF) helicopter system involves various parameters and variables that play significant roles in its dynamics and control. These include moments of inertia (I_{xx}, I_{yy}, I_{zz}) representing the rotational mass distribution around the x, y, and z axes, respectively. The angles of roll, pitch, and yaw (ϕ, θ, ψ) describe the orientation of the helicopter in three-dimensional space. The mass of the rotor assembly (m) and the mass of the entire setup (M) influence the overall system dynamics. The acceleration due to gravity (g) provides a gravitational reference for the system. Length parameters such as l_{boom} (from pivot point to heli body), l_{ϕ} (length of the pendulum for roll axis), l_{θ} (length of the pendulum for pitch axis), and l_h (from pivot point to the rotor) determine the physical dimensions of the helicopter. The roll damping coefficient (L_p) and pitch damping coefficient (M_q) contribute to the system's damping characteristics. The coefficient of thrust (D) relates to the generation of lift and propulsion. The collective pitch torque

(τ_{coll}) and cyclic pitch torque (τ_{cyc}) represent control inputs that affect the rotor dynamics. The control gains (K_τ, K_v) determine the relationship between control input and control torque. The derivatives of cyclic pitch and collective pitch angular velocities ($\dot{\omega}_{cyc}$, $\dot{\omega}_{coll}$) represent the rate of change of these quantities. Finally, the inputs corresponding to cyclic pitch and collective pitch control (V_{cyc}, V_{coll}) directly influence the helicopter's motion and behavior. These parameters and variables collectively define the nonlinear equations of motion and form the basis for analyzing and controlling the Quanser 3-DOF helicopter system.

Table 1

Physical Parameter of the System

Parameter	Description	Value
m	System mass	1.15 kg
L	System length	3.57 m
θ_{rest}	Theta rest value	-25 deg.

Note. This is not a full list of parameters; you can find additional parameters in Appendix A

Theory

State-Space Model

The state-space model is a mathematical representation of a dynamic system using a set of first-order differential equations. It consists of state variables (x), input variables (u), output variables (y), and matrices (A, B, C, and D) that describe the system's behavior over time. The model is commonly written in the form of two equations:

$$\dot{x} = Ax + Bu \tag{9}$$

$$y = Cx + Du \tag{10}$$

Stability

The stability of a system represented by a state-space model can be determined by analyzing the eigenvalues of the A matrix. The eigenvalues provide important information about the system's behavior, and their properties indicate whether the system is stable, marginally stable, or unstable.

The equation to determine the eigenvalues of A matrix in a state space model is:

$$\det(\lambda I - A) = 0 \tag{11}$$

The eigen vector is a complex number composed of real and imaginary parts can been seen below.

$$\lambda = \sigma \pm i\omega \tag{12}$$

If the real part eigen vector $\lambda < 0$, the system is stable.

If the real part eigen vector $\lambda > 0$, the system is unstable.

If the real part eigen vector $\lambda = 0$, the system is marginally stable.

Controllability

Controllability is a property of a dynamic system that determines whether it is possible to steer or control the system's state from any initial condition to any desired state using appropriate control inputs. In other words, a system is said to be controllable if it can be manipulated in such a way that it can reach any desired state by applying suitable control signals. The system is controllable or not can be determined by following equation. If the matric determined is fully ranked or the determinant of the matix is non-zero, then it is controllable.

$$rank\ (M_C)\ =\ rank([B \quad AB \quad A^2B \ .. \ A^{n-1}N])\ =\ n \tag{13}$$

Observability

Observability is a property of a dynamic system that determines whether its internal states can be estimated or observed from the available output measurements. In other words, a system is said to be observable if it is possible to deduce or estimate the system's current state based solely on the knowledge of its input and output data. If the matric determined is fully rank or the determinant of the matix is non-zero, then it is observable.

$$rank(M_o) = \left\{ \begin{bmatrix} C \\ CA \\ CA^2 \\ \cdot \\ \cdot \\ \cdot \\ C\,A^{n-1} \end{bmatrix} \right\} = n \tag{14}$$

Stabilizability

Stabilizability is a fundamental property for controllability. A system can only be controllable if it is stabilizable, but stabilizability does not guarantee controllability. Controllability requires that the system can be driven to any desired state, while stabilizability only ensures that the system can be driven to a stable state or a specific target. If a single-input single-output (SISO) system's transfer function exhibits no zero-pole cancellations, the system is considered both controllable and observable. However, the presence of zero-pole cancellations renders the system either uncontrollable, unobservable, or even both uncontrollable and unobservable. To determine the stabilizability of a system, it is crucial to ensure that all unstable modes are controllable, meaning there are no uncontrollable unstable modes.

Detectability

A system is considered detectable if it is possible to design an observer or estimator that can reliably estimate the internal states based on the available output measurements. Detectable system has all its unstable modes observable, ensuring reliable estimation of internal states for control; no unobservable unstable modes exist, preventing undetected instability risks.

LQR Controller (Roll)

The LQR (Linear Quadratic Regulator) controller is a popular control design technique used in control theory to stabilize and regulate linear dynamic systems, minimizing a cost function while satisfying certain performance and stability criteria. The LQR is a controller with a reference input set to zero, emphasizing optimization. It is a pole-placement technique that connects states and control inputs eq. (15). The controller is designed to stabilize an unstable system by shifting the poles of the system's transfer function from the right-half plane (unstable region) to the left-half plane (stable region) of the complex plane. The integral includes the state cost and control cost terms. By adjusting the R matrix, closed-loop poles are optimized to minimize state errors.

$$J_{LQR} = \int_0^\infty [\vec{x}(t)^T Q\vec{x}(t) + \vec{u}(t)^T R\vec{u}(t)]\, dt \tag{15}$$

The $\vec{x}(t)^T Q\vec{x}(t)$ term is the state cost with the weight Q. Weight Q penalizes the state if it does not stabilize at the reference value, it is also known as the state pe. The $\vec{u}(t)^T R\vec{u}(t)$ term is called the control cost with weight R. R effectively sets the controller bandwidth and can be thought of as the amount of energy spent to control a system. Here,

$$Q = C^T \times C \tag{16}$$

$$R = \rho I \tag{17}$$

ρ is called the controller bandwidth usually set by the designer. Eq. (16) and (17) are good starting points when trying to minimize the cost function because it provides approximately equal weight to both output error and control effort.

$$\text{Feedback Gain: } K_{lqr} = R^{-1}B^T P \tag{18}$$

$$\text{Algebric Riccati Equation: } 0 = A^T P + PA + Q - PBR^{-1}B^T P \tag{19}$$

There exists an optimum gain “K_{lqr}” that minimizes the cost function, which can be found by eq. (18) (How & Frazzoli, 2010). P is a parameter in the Algebraic Riccati Equation, which if solved would give the best K_{lqr} that is required.

LQ Servo Controller (Pitch)

The LQ Servo Controller is a robust method that manipulate complex poles by penalizing integrated output. Both LQ servos and LQR share the cost function solving, but they differ in replacing Q with $\bar{Q}$ a new matrix combining the Q term and E.

$$\bar{Q} = \begin{bmatrix} Q & 0 \\ 0 & E \end{bmatrix} \tag{20}$$

E, known as the integration error penalty, is set by the designer and crucial in enabling the LQ Servo to achieve zero state error without relying on ($\overline{N}$), showcasing its robustness to modeling errors. However, a drawback of the LQ Servo is its sole reliance on the integrated error to respond to the reference, lacking a direct path from the reference input to the system, potentially leading to slower transients (How & Frazzoli, 2010**).**

LQE Closed-loop Optimal Estimators

When limited measurement is provided, a closed-loop optimal estimator provides estimation of a full state of a system. When all the state of a system is not directly accessible and is only inferred to noisy or partial measurement, then LQE plays a crucial role. Basically, the estimator uses feedback control to minimize the estimation error. It adjusts the estimator’s gain matrix or feedback the estimation output error to improve the estimate. LQE has two inputs, control input of “u” and output “y” that estimate the full state. The gain matrix “L” is used as key matrix to improve the convergence of the error. The other three covariance matrices, “Q”, “plays significant 14 role in stability and performance of the estimator, “Rww” to ensure the unmodeled disturbances are accounted and “Rvv” to find out any error or noise in the measurements.

$$L = QC^T(R_{vv})^{-1} \tag{21}$$

“Q” is a symmetrical matrix that is determined by Riccati Equation. Selecting the optimal values for R_{vv} (measured noise covariance) and R_{ww} (process noise covariance) is crucial. As the gain matrix L increases, the estimator becomes faster but also introduces more noise to the system. Hence, careful consideration and balancing of these parameters are necessary to achieve the desired trade-off between estimation speed and noise level.

Dynamic Output Feedback (DOFB) Controllers

A dynamic output feedback controller is a design of controller which is combination of estimators and regulators. The design of DOFB controllers involves carefully selecting the state estimator, tuning the control law parameters, and considering the trade-offs between performance, stability, and control effort.

$$\widehat{\dot{\vec{x}}} = (A - BK - LC)\hat{\vec{x}} + L\vec{y} \tag{22}$$

Methodology

L_p and M_q Range Test

To perform L_p and M_q range tests on a state-space model, we need to analyze the stability, controllability, observability, stabilizability, and detectability properties of the system. The state-space models were coded in MATLAB and with the help of two loop functions, the range of L_p and M_q values were iterated to perform each analysis. The stability of the matrix A is calculated by calculating the eigenvalues (λ), “*eig()*”. Similarly, controllability and observability were performed with the in-built function of “*cont = ctrb() and obs = obsv()*” respectively. The rank of each matrix were checked with the "*rank()*" matrix. Note: all the MATLAB functions ion Appendix B relating to L_p and M_q range test. The "*disp()*" function provides the result when each of the loop functions was processed.

LQR Controller Selection

After the testing of L_p and M_q values, by learning in theory, we came to know that the controllable system can only be an LQR controller. Therefore, an L_p value that fits this criterion should be used to make a state-space model for roll. As eq. (17) and (18) provides us with the need of finding weights Q and R.

An Lp value that fits this requirement is used to make a new state-space model for roll. The next step is the determination of the weights Q and R. Equation 19 can be simply done in MATLAB using matrix multiplication on MATLAB. Simply using the *"[K] = lqr(Ar,Br,Q,R)*" in MATLAB for a defined range of R, the response for 20deg step input was plotted. As R depends on the control bandwidth, a for loop is used to test various ρ values and then kept in R, meaning that its value constantly changes until the loop ends. The identity matrix was designed through "*eye()*" function. The in-built function for R was used as "*R = rho*eye(1)".*

LQ Servo Selection

Like LQR, the LQ Servo has a similar method, which determines an appropriate M_q value to fulfill the system requirements. Subsequently, a new pitch state space is constructed based on the chosen M_q value. The augmented system matrices Abar and Bbar are formed by adding a zero to the last column of A_p and B_p, respectively, using the MATLAB function: "*Abar = [Ap, zeros(3, 1); -Cp, 0]; Bbar = [Bp; 0]".* To design the cost function weighing matrix for the state errors, Q is created as the transpose of Cp multiplied by Cp. The LQR (Linear-Quadratic Regulator) control function "lqr()" was then utilized to compute the optimal gain matrix Kbar, the associated algebraic Riccati equation solution Pbar, and the eigenvalues Lamd_bar, which represent the stability of the closed-loop system. Furthermore, after the for-loop, a specific value of E (E = 550) is chosen as the integration penalty error.

LQE Estimators Selection

Two estimators were created by defining their corresponding state spaces. The median values of L_p and M_q were utilized to define the system matrices. The tuning parameter (rho), covariance matrix (R_{ww}), and the covariance of the measurement noise (R_{vv}) were also defined. Symbolic math commands in MATLAB were employed to define various variables for the covariance matrix. Subsequently, the gain matrix was utilized to facilitate the convergence of the estimation error. The state space was defined using the new L gain, and a graph was plotted to illustrate the relationship between the Roll Angle and Time.

Dynamic Output Feedback (DOFB) Controller Selection

As explained in theory that DOFB controller is an integrated estimator and regulator, first the LQ Servo design is formulated exactly as mentioned above. The DOFB controller combines a state-feedback controller with an observer (estimator) to estimate the unmeasured states. This controller is designed with a feedforward term (Nbar) to improve tracking performance. The feedforward term is determined to achieve a desired steady-state response. The matrix Anbarcl, Bnbarcl , and Cnbarcl represent the closed-loop system with both the controller and the estimator, along with the feedforward term Nbar. Note: These functions are used in Appendix G and H.

Results and Discussions

As each non-linear term is specified, the derivation of the linearized motor equation for the Quanser 3-DOF helicopter is provided as:

a. Roll direction linearization.

Substituting eq. (6) into eq. (1), we get

$$I_{xx}\ddot{\phi} = (K_\tau \omega_{cyc} - K_v \dot{\psi}) l_h - mgl_\phi sin\phi - L_p \dot{\phi} - I_r \omega_{rotor}(\dot{\theta}\cos\phi + \dot{\psi}\sin\phi)$$

Now differentiating concerning ϕ, we get,

$$I_{xx}\Delta\ddot{\phi} = K_\tau \Delta\omega_{cyc} l_h - K_v \Delta\dot{\psi} l_h - mgl_\phi cos\phi\Delta\phi - L_p \Delta\dot{\phi} + I_r \omega_{rotor}\dot{\theta}\sin\phi\Delta\phi$$

$$- I_r \omega_{r\dot{o}tor}\psi\cos\phi\Delta\,\dot{\theta} - I_r \omega_{rotor}\psi\cos\phi\Delta\phi - I_r \omega_{r\dot{o}tor}\psi\sin\phi\Delta\,\psi$$

Substituing trim values $cos0° = 1$ and $sin0° = 0$, we get,

$$\Delta\ddot{\phi} = \frac{K_\tau \Delta\omega_{cyc} l_h - K_v \Delta\dot{\psi} l_h - mgl_\phi \Delta\phi - L_p \Delta\dot{\phi} - -I_r \omega_{rotor}\,\Delta\dot{\theta} - I_r \omega_{rotor}\psi\Delta\phi}{I_{xx}} \quad (23)$$

b. Pitch direction linearization

Substituting eq. (5) into eq. (2) and differentiating with respect to θ and substituing trim values, $cos0° = 1$ and $sin0° = 0$, we get,

$$\Delta\ddot{\theta} = \frac{K_\tau l_{boom}\Delta\omega_{coll} - K_v l_{boom}\Delta\dot{\psi} - Mgl_\theta\cos(\theta_{rest})\Delta\,\theta - K_D\psi l_{bo\dot{o}m}\,\Delta\gamma + I_r \omega_{rotor}\Delta\phi - \dot{M}_q\Delta\dot{\theta}}{I_{yy}} \quad (24)$$

c. Yaw direction linearization

Substituting eq. (5) into eq. (3) and differentiating concerning ψ and substituing trim values, $cos0° = 1$ and $sin0° = 0$, we get,

$$\Delta\ddot{\psi} = \frac{-K_D\, l_{boom}\Delta\dot{\psi}}{I_{zz}} \quad (25)$$

Therefore, the given linearized equations in the matrix form is

Roll, neglecting travel dynamics;

Input Matrix

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\omega}_{cyc} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\dfrac{mgl_{\phi}}{I_x} & -\dfrac{L_p}{I_x} & \dfrac{K_{\tau} l_h}{I_x} \\ 0 & 0 & -6 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \omega_{cyc} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 780 \end{bmatrix} V_{cyc} \tag{26}$$

Output Matrix

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \omega_{cyc} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} V_{cyc} \tag{27}$$

Pitch, neglecting travel dynamics;

Input Matrix

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{\omega}_{coll} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\dfrac{Mgl_{\theta}\cos(\theta_{rest})}{I_y} & -\dfrac{M_q}{I_y} & \dfrac{K_{\tau} l_{boom}}{I_y} \\ 0 & 0 & -6 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega_{coll} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 540 \end{bmatrix} V_{coll} \tag{28}$$

Output Matrix

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega_{coll} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} V_{coll} \tag{29}$$

Full dynamics state space (neglecting only rotor interia);

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \\ \dot{\omega}_{cyc} \\ \dot{\omega}_{coll} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \dfrac{-mgl_{\phi}}{I_x} & 0 & 0 & \dfrac{-L_p}{I_x} & 0 & \dfrac{-K_v l_h}{I_x} & \dfrac{-K_{\tau} l_h}{I_x} & 0 \\ 0 & \dfrac{-Mgl_{\theta}\cos(\theta_{rest})}{I_y} & 0 & 0 & \dfrac{-M_q}{I_y} & \dfrac{-K_v l_{boom}}{I_y} & 0 & \dfrac{-K_{\tau} l_{boom}}{I_y} \\ \dfrac{K_{\tau} l_{boom}(\omega_{coll,0})}{I_z} & 0 & 0 & 0 & 0 & \dfrac{-K_D l_{boom}}{I_z} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \omega_{cyc} \\ \omega_{coll} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 780 & 0 \\ 0 & 540 \end{bmatrix} \begin{bmatrix} V_{cyc} \\ V_{coll} \end{bmatrix} \tag{30}$$

It was discovered after running the code in Appendix B that the roll and pitch are stable, observable, controllable, stabilizable, and detectable for the provided range of L_p and M_q values. The outcomes of the iteration are summarized in Tables (2) and (3).

Table 2

Summary Table of L_p Range

L_p value	Eigenvalue	Controllability	Observability	Stabilizability	Detectability
0.02	-0.2778 + 1.0846i -0.2778 - 1.0846i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.04	-0.5556 + 0.9720i -0.5556 - 0.9720i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.06	-0.8333 + 0.7477i -0.8333 - 0.7477i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.08	-1.1111 + 0.1376i -1.1111 - 0.1376i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.1	-0.5670 -2.2108 -6.0000	Controllable	Observable	Stabilizable	Detectable
0.12	-0.4321 -2.9013 -6.0000	Controllable	Observable	Stabilizable	Detectable
0.14	-0.3547 -3.5342 -6.0000	Controllable	Observable	Stabilizable	Detectable
0.16	-0.3026 -4.1418 -6.0000	Controllable	Observable	Stabilizable	Detectable
0.18	-0.2647 -4.7353 -6.0000	Controllable	Observable	Stabilizable	Detectable
0.2	-0.2356 -5.3199 -6.0000	Controllable	Observable	Stabilizable	Detectable

Table 3

Summary Table of M_q Range

M_q values	Eigenvalues	Controllability	Observability	Stabilizability	Detectability
0.100	-0.0538 + 0.7209i -0.0538 - 0.7209i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.185	-0.0995 + 0.7160i -0.0995 - 0.7160i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.270	-0.1452 + 0.7082i -0.1452 - 0.7082i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable

0.355	-0.1909 + 0.6972i -0.1909 -0.6972i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.440	-0.2366 + 0.6831i -0.2366 - 0.6831i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.525	-0.2823 + 0.6655i -0.2823 - 0.6655i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.610	-0.3280 + 0.6442i -0.3280 - 0.6442i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.695	-0.3737 + 0.6188i -0.3737 - 0.6188i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.780	-0.4194 + 0.5888i -0.4194 - 0.5888i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable
0.865	-0.4651 + 0.5534i -0.4651 - 0.5534i -6.0000 + 0.0000i	Controllable	Observable	Stabilizable	Detectable

The tables above show the L_p and M_q values for the range of the value given in Table (1) and (2) respectively, at some instances. At every value the eigen modes are stable as it shows the negative real part. For L_p values the oscillations are fading away after 0.1 as there are not imaginary parts.

Figure 1

Showing the result of different rho values

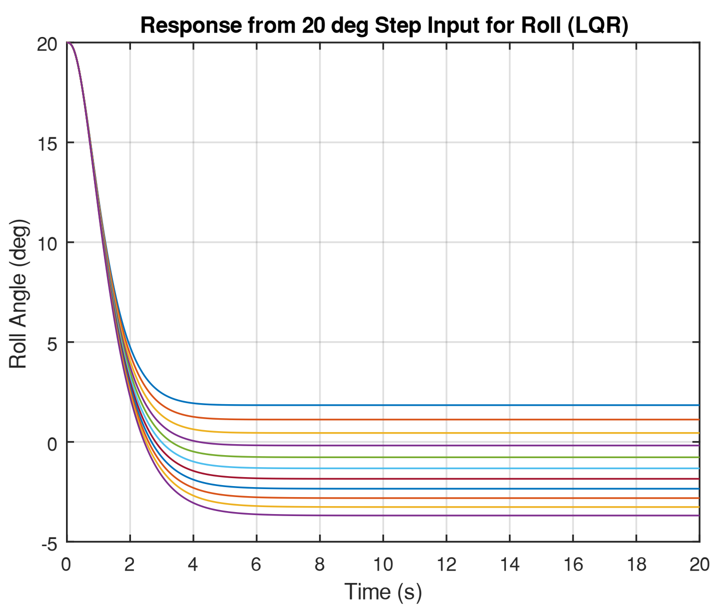


Fig. (1) shows 20-degree step response for roll at different rho values, higher values of ρ show steady state error. The best value of rho is selected to be 1.3 from Fig. (1) and the Roll angle-Time graph is shown in Appendix H

Figure 2

Vcyc vs Time (s) of Roll (LQR) for different rho values

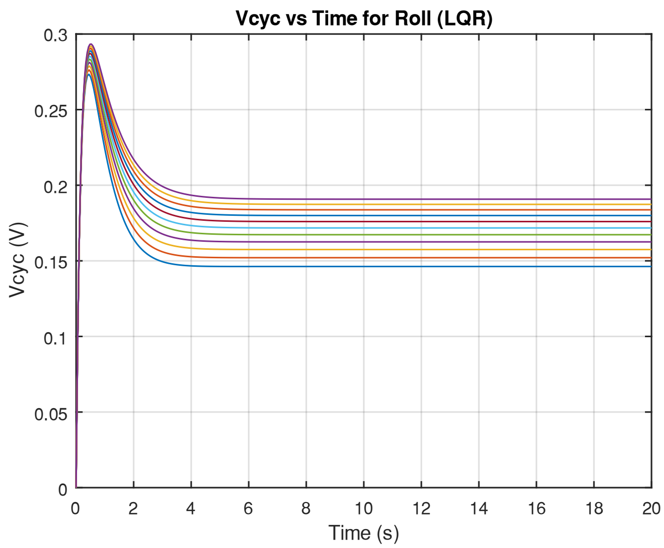


Fig. (2) shows the voltage vs time graph for roll during the 20-degree step response. The voltage is saturated within the constraint set of ± 5V. The voltage-time graph can be seen for rho = 1.3 in Appendix I

Figure 3

Response of Pitch for 20 Degree Step Input

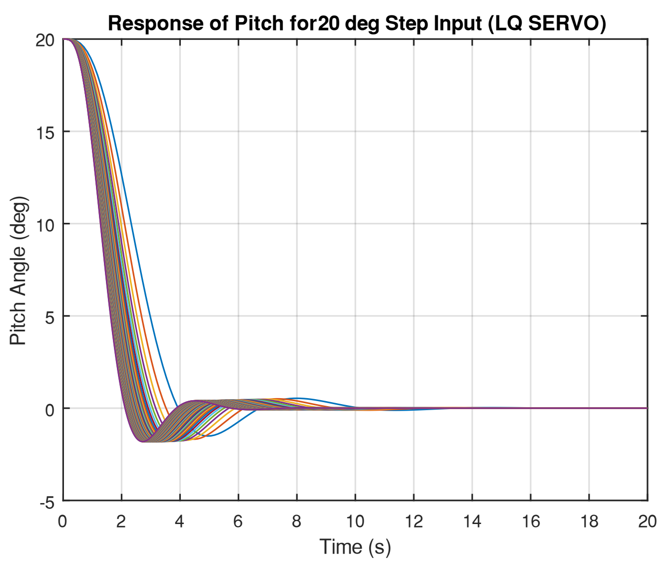


Fig. (3) shows this step response of LQ Servo controller for pitch. Error between the commanded and actual pitch angles is being added while plotting the graphs. The array of penalty integration factor (E) is used to plot graph. The optimal value of integration factor was found, which is E=550 where the system is reaching steady state faster, that is close to 10 seconds

Figure 4

Voltage (V_{coll}) Vs Time for Pitch Graph

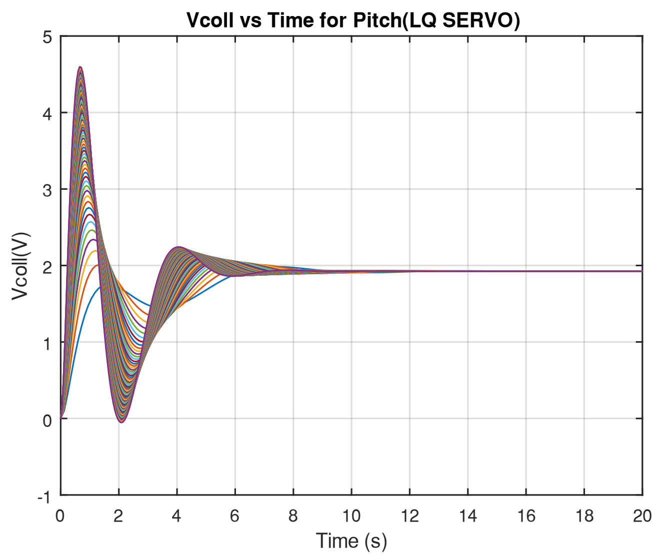


Fig. (4) shows the voltage vs time graph for pitch during the 20-degree step response for different values of integration error. The voltage is within the constraint set of +- 5V for integration error E=550.

Figure 5

Roll Estimator Graph

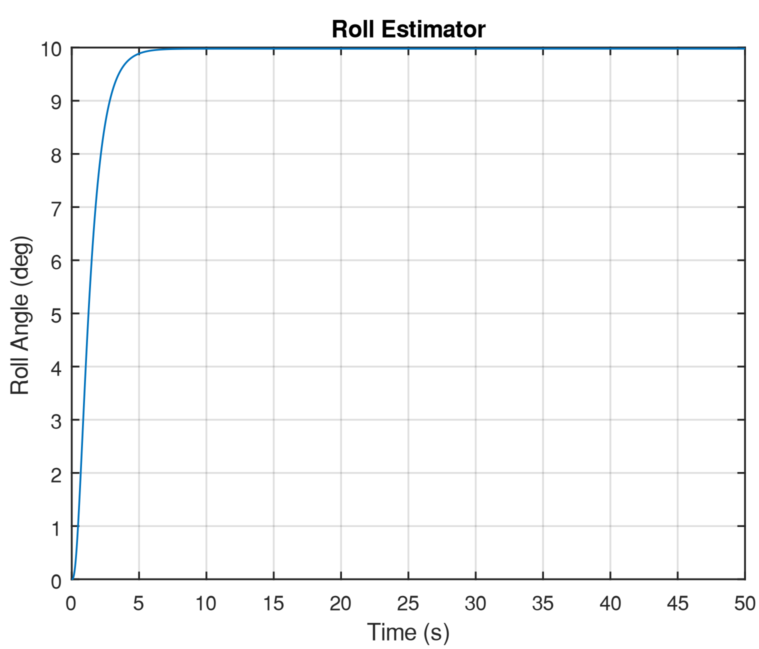
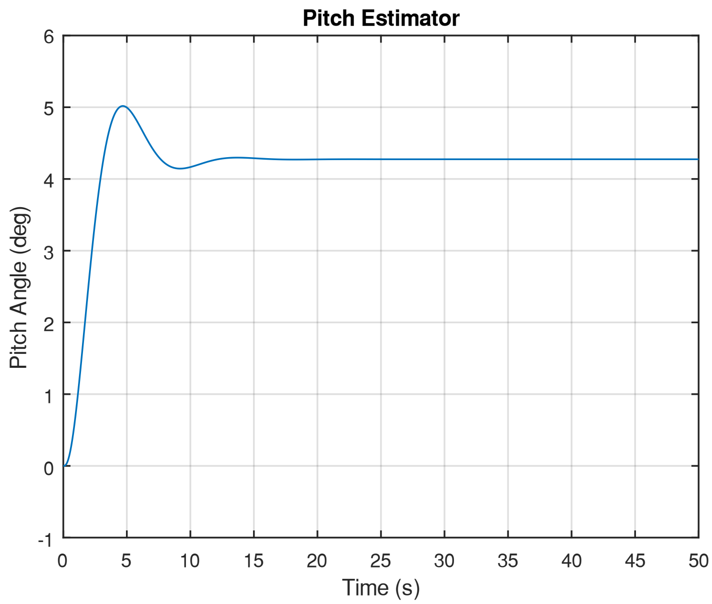


Figure 6

Pitch Estimator Graph



The figure (5) and (6) shows the stimulated response for a 10-degree step input to the system estimator for Roll and Pitch. For a 10deg step input, the roll and pitch degree, when stable, are close to 10 deg and 4.2 deg respectively. The appropriate values of R_{ww} for the Roll Estimator and Pitch Estimator were calculated using the MATLAB function 'covar' based on the range of L_p and M_q values. For the Roll Estimator, R_{ww} was determined to be 0.1695, and for the Pitch Estimator, R_{ww} was found to be 0.1026. The selection of R_{vv} for the Roll and Pitch Estimators was based on the hit and trial method to stabilize the system against disturbances. For the Roll Estimator, the chosen value was 0.3, and for the Pitch Estimator, the chosen value was 1.

The dynamic output feedback controller from part 5 was simulated into MATLAB/Simulink form to represent the simulation and responses were recorded. The graph obtained from MATLAB is presented below. The graph obtained from Simulink is generated in Appendix K (for Angles Vs Time) and Appendix L (for Voltages Vs Time).

Figure 7

Roll Angle Vs Time Graph

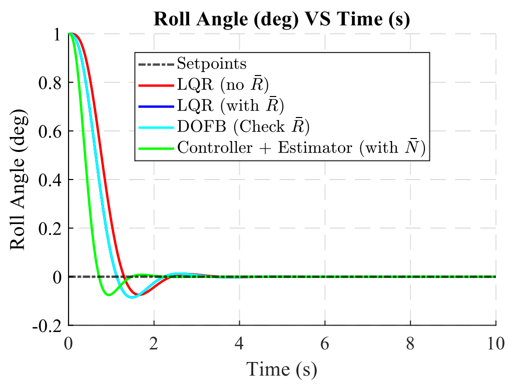


Figure (7) shows the dynamic output feedback controller for roll that is reaching to steady state for all set points.

Figure 8

Pitch Angle Vs Time Graph

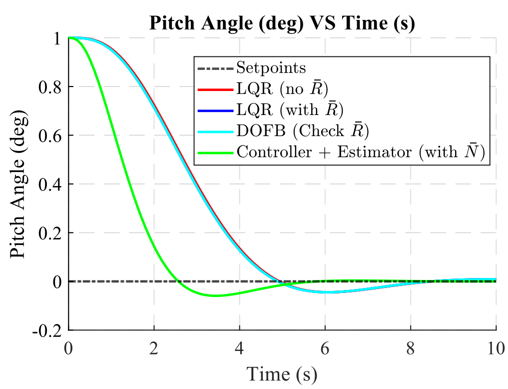


Figure (8) shows the dynamic output feedback controller for pitch that is reaching to steady state for all set points.

Conclusion

The project successfully explored the control of the Quanser three-degree-of-freedom DOF) helicopter, establishing its stability, observability, controllability, stabilizability, and detectability within the selected Lp and Mq range. State-space models and controllers, such as LQR and LQ Servo, were designed and simulated based on these parameters. The LQR controller demonstrated optimal stabilization and control for roll dynamics, while the LQ Servo achieved precise reference tracking and setpoint regulation for pitch dynamics. The study evaluated various control strategies, estimators, and DOFB controllers to enhance system performance. Improving the dynamic output of the Quanser helicopter can be achieved by fine-tuning the weighting matrices Q and R in the LQR controller. Incorporating integral action (E) in the LQ Servo controller (LQI control) can eliminate steady-state errors and improve tracking accuracy. Adjusting Rvv and Rww enhance the gain matrix L, led to better state estimation and control performance.

References

B. Zheng and Y. Zhong, "Robust Attitude Regulation of a 3-DOF Helicopter Benchmark: Theory and Experiments," in IEEE Transactions on Industrial Electronics, vol. 58, no. 2, pp. 660-670, Feb. 2011, doi: 10.1109/TIE.2010.2046579.

Beard, R. W., & McLain, T. W. (2012). Small unmanned aircraft: Theory and practice. Princeton university press.

Drela, M. (2014). Flight vehicle aerodynamics. MIT press.

How, J., & Frazzoli, E. (2010, September). Feedback Control Systems | Aeronautics and Astronautics MIT OpenCourseWare. MIT OpenCourseWare | Free Online Course Materials; MIT OpenCourseWare. https://ocw.mit.edu/courses/16-30-feedback-controlsystems-fall-2010/

How, J., & Frazzoli, E. (2010b, September). MIT16_30F10_lec12.pdf | Feedback Control Systems | Aeronautics and Astronautics | MIT OpenCourseWare. MIT OpenCourseWare | Free Online Course Materials; MIT OpenCourseWare. https://ocw.mit.edu/courses/16-30- feedback-control-systems-fall-2010/resources/mit16_30f10_lec12/

Valavanis, K. P., & Vachtsevanos, G. J. (Eds.). (2015). Handbook of unmanned aerial vehicles (Vol. 1). Dordrecht: Springer Netherlands.

3 DOF helicopter. Quanser. (2021, April 20). Retrieved July 23, 2023, from https://www.quanser.com/products/3-dof-helicopter/

Appendix A – Physical Parameter

Parameter	Description	Value
m	System mass	1.15 kg
L	System length	3.57 m
l_{boom}	Length from pivot point to Heli body	0.66 m
l_{ϕ}	Length of the pendulum for roll axis	0.004 m
l_{θ}	Length of the pendulum for pitch axis	0.014 m
l_h	Length from the pivot point to the rotor	0.177 m
I_{xx}	Moment of inertia about the x- axis	0.036 Nms^2
I_{yy}	Moment of inertia about the y- axis	0.93 Nms^2
K_{τ}	Coefficient of thrust	$4.25 \times 10^{-3}\ Ns$
θ_{rest}	Theta rest value	-25 deg.
g	Acceleration due to gravity	9.81 m/s^2
L_p	Roll damping coefficient	[0.02, 0.2] Nms
M_q	Pitch damping coefficient	[0.1, 0.9] Nms

Note. This is the full list of parameters.

Appendix B – Lp and Mq Values from State Space Models

```
clear;clc;close all;

%% Assigning Physical Parameters of 3-DOF unit

m = 1.15;                                % Mass of rotor assembly (kg)
M = 3.57;                                % Mass of the whole setup (kg)
l_boom = 0.66;                           % Length from pivot point to heli body (m)
l_phi = 0.004;                           % Length of the pendulum for roll axis (m)
l_theta = 0.014;                         % Length of the pendulum for pitch axis (m)
l_h = 0.177;                             % Length from pivot point to the rotor (m)
I_xx = 0.036;                            % Moment of inertia about x-axis (Nms^2)
I_yy = 0.93;                             % Moment of inertia about y-axis (Nms^2)
I_z = 0.93;                              % Moment of inertia about z-axis (Nms^2)
K_tau = 0.00425;                         % Coefficient of thrust (Ns)
Theta_rest = -25;                        % Theta rest value (degrees)
g = 9.81;                                % Acceleration due to gravity (m/s^2)
L_p = 0.02:0.02:0.2;                     % Roll damping coefficient (Nms), predefined
range
M_q = 0.1:0.085:0.9;                     % Pitch damping coefficient (Nms), prede-
fined range
K_D = 0;                                 % Coefficient of drag
w_coll0 = (M*g*l_theta*sin...            % Trim state angular acceleration
(Theta_rest))/(K_tau*l_boom);
K_v = 0.0125*w_coll0*K_tau*l_boom;       % Coefficient of v
n = length(L_p);                         % Array index for Lp for loop
o = length(M_q);                         % Array index for Lp for loop

%% Phase 1 - State-Space Models (checking stability, controllability, observability,
stabilizability, and detectability)

% Using for loop to integrate the values of the range of the L_P
for i = 1:1:n
  Set = i; disp("Iteration Number for Lp values: "),disp(Set)
    Lp(i) = L_p(i);

% Defining the system Matrixes for Roll
 Ar = [0, 1, 0; (-m*g*l_phi)/I_xx, -Lp(i)/I_xx, (K_tau*l_h)/I_xx; 0 0 -6];
 Br = [0; 0; 780];
 Cr = [1 0 0; 0 0 0; 0 0 0];
 Dr = 0;


 % Checking the eigen values for roll motion for Stability
 lambda_r = eig(Ar);
 if lambda_r < 0
   disp('For the Lp value: '), disp(Lp(i)); disp("The system is stable");
disp(lambda_r)
 elseif lambda_r == 0
   disp('For the Lp value: '), disp(Lp(i)); disp("The system is marginally stable")
 else
   disp('For the Lp value: '), disp(Lp(i)); disp("The system is not stable")
 end
```

```
% Checking if the system is controllable for Roll
cont = ctrb(Ar, Br);
Mc = rank(cont);
if Mc == 3
    disp('For the Lp value: '), disp(Lp(i))
    disp("The system is controllable")
else
    disp('For the Lp value: '), disp(Lp(i))
    disp('The system is uncontrollable')
end


% Checking if the system is Observable check for Roll
obs = obsv(Ar, Cr);
Mo = rank(obs);
if Mo == 3
    disp('For the Lp value: '), disp(Lp(i))
    disp("The system is observable")
else
    disp('For the Lp value: '), disp(Lp(i))
    disp("The system is not observable")
end
end

% Using for loop to interate the values of the range of the M_P
for j = 1:1:o
  Set2 = j; disp("Iteration Number for Mq values: "), disp(Set2)
  Mq(j) = M_q(j);

% Defining the system Matrixes for Pitch
Ap = [0, 1, 0; (-M*g*l_theta*cos(Theta_rest))/I_yy, -Mq(j)/I_yy, (K_tau*l_boom)/I_yy;
0 0 -6];
Bp = [0; 0; 540];
Cp = [1, 0, 0; 0, 0, 0; 0, 0, 0];
Dp = 0;


% Checking the eigen values for pitch motion for Stability
lambda_p = eig(Ap);
 if lambda_p < 0
   disp('For the Mq value: '), disp(Mq(j)); disp("The system is stable"),
disp(lambda_p)
 elseif lamdap == 0
   disp('For the Mq value: '), disp(Mq(j)); disp("The system is marginally stable"),
disp(lambda_p)
 else
   disp('For the Mq value:'), disp(Mq(j)); disp("The system is not stable"),
disp(lambda_p)
 end


% Checking if the system is controllable for Pitch
cont = ctrb(Ap, Bp);
Mc = rank(cont);
```

```
if Mc == 3
    disp('For the Mq value: '), disp(Mq(j))
    disp("The system is controllable")
else
    disp('For the Mq value: '), disp(Mq(j))
    disp('The system is uncontrollable'),
end


% Checking if the system is Observable check for Pitch
obs = obsv(Ap, Cp);
Mo = rank(obs);
if Mo == 3
    disp('For the Mq value: '), disp(Mq(j))
    disp("The system is observable")
else
    disp('For the Mq value'), disp(Mq(j))
    disp("The system is not observable")
end
end

% Mq and Lp values selected in stable, controllable, stabilizable, observable, and
detectable range
Lpn = 0.1; % Lp value selected
Mqn = 0.440; % Mq value selected
```

Appendix C – Developing LQR Controller for Roll

```
clear;clc;close all;

%% Assigning Physical Parameters of 3-DOF unit

m = 1.15;                             % Mass of rotor assembly (kg)
M = 3.57;                             % Mass of the whole setup (kg)
l_boom = 0.66;                        % Length from pivot point to heli body (m)
l_phi = 0.004;                        % Length of the pendulum for roll axis (m)
l_theta = 0.014;                      % Length of the pendulum for pitch axis (m)
l_h = 0.177;                          % Length from pivot point to the rotor (m)
I_xx = 0.036;                         % Moment of inertia about x-axis (Nms^2)
I_yy = 0.93;                          % Moment of inertia about y-axis (Nms^2)
I_z = 0.93;                           % Moment of inertia about z-axis (Nms^2)
K_tau = 0.00425;                      % Coefficient of thrust (Ns)
Theta_rest = -25;                     % Theta rest value (degrees)
g = 9.81;                             % Acceleration due to gravity (m/s^2)
L_p = 0.02:0.02:0.2;                  % Roll damping coefficient (Nms), predefined
range
M_q = 0.1:0.085:0.9;                  % Pitch damping coefficient (Nms), prede-
fined range
K_D = 0;                              % Coefficient of drag
w_coll0 = (M*g*l_theta*sin...         % Trim state angular acceleration
(Theta_rest))/(K_tau*l_boom);
K_v = 0.0125*w_coll0*K_tau*l_boom;    % Coefficient of v
n = length(L_p);                      % Array index for Lp for loop
o = length(M_q);                      % Array index for Lp for loop


% Defining the system Matrixes for Roll
Lp = 0.1;
Ar = [0, 1, 0; (-m*g*l_phi)/I_xx, -Lp/I_xx, (K_tau*l_h)/I_xx; 0 0 -6];
Br = [0; 0; 780];
Cr = [1 0 0];
Dr = 0;

%% Developing LQR controller for Roll
sys1 = ss(Ar, Br, Cr,Dr);                                  % Open loop state space
model
Q = Cr'*Cr;                                                % Weight Q from the cost
function
for rho = 1:0.1:2
                                                           % Checking for a suita-
ble rho value
    R = rho*eye(1);                                        % Weight R used from the
cost function
    [K] = lqr(Ar,Br,Q,R);                                  % Determining LQR gain
(K)
    sys2 = ss(Ar-Br*K,Br,Cr,Dr);                           % Finding closed loop
state-space system
    [y2, t2, x21] = step(sys2*20*pi()/180, 20);            % Closed loop step re-
sponse
```

```
    figure(1)                                          % Plotting step response
of the system
    plot(t2,(20*ones(size(t2))-y2*180/pi())); hold on
    title('Response from 20 deg Step Input for Roll (LQR)')
    grid on
    xlabel('Time (s)')
    ylabel('Roll Angle (deg)')


    % Zero Steady State error is included in Vcyc
    Vcyc = 6/780*(x21(:,3))';                          % Plotting Voltage vs
Time for Roll

    figure(3)
    plot(t2,Vcyc); hold on
    title('Vcyc vs Time for Roll (LQR)')
    grid on
    xlabel('Time (s)')
    ylabel('Vcyc (V)')
end

% "After analyzing the Step response graph rho = 1
% is found to be optimal for the system as it is stabiliing the system first" %

Rho = 1.3;                                               % Rho for loop found
rho = 1 is the optimum value
R = Rho*eye(1); % New R
[Krn,P,CLPlam] = lqr(Ar,Br,Q,R);                       % Determining LQR gain
(K) for Rho = 1
disp('Solution P of the associated algebraic Riccati equation '); disp(P)
disp('LQR gain'); disp(K)
disp('Closed loop eigen value'); disp(CLPlam)                    % Determining
Nbar
sys3 = ss(Ar-Br*Krn,Br,Cr,Dr);                     % Finding closed loop state
space system
[y2, t2, x22] = step(sys3*20*pi()/180, 20);            % Closed loop system
step response



figure(2)                                              % Plotting step response
of the system for Rho = 1
plot(t2, (20*ones(size(t2))-y2*180/pi())); hold on
title('Response for 20 deg Step Input for Roll (LQR)')
grid on
xlabel('Time (s)')
ylabel('Roll Angle (deg)')

% Zero Steady State error is included in Vcyc
Vcycn = 6/780*(x22(:,3))';

figure(4)                                              % Plotting Voltage vs
Time for Roll (Rho = 1)
plot(t2,Vcycn)
title('Vcyc vs Time or Roll (LQR)')
```

```
grid on
xlabel('Time (s)')
ylabel('Vcyc (V)')
```

Appendix D – Developing LQ Servo Controller for Pitch

```
clear;clc;close all;
%% Assigning Physical Parameters of 3-DOF unit

m = 1.15;                               % Mass of rotor assembly (kg)
M = 3.57;                               % Mass of the whole setup (kg)
l_boom = 0.66;                          % Length from pivot point to heli body (m)
l_phi = 0.004;                          % Length of the pendulum for roll axis (m)
l_theta = 0.014;                        % Length of the pendulum for pitch axis (m)
l_h = 0.177;                            % Length from pivot point to the rotor (m)
I_xx = 0.036;                           % Moment of inertia about x-axis (Nms^2)
I_yy = 0.93;                            % Moment of inertia about y-axis (Nms^2)
I_z = 0.93;                             % Moment of inertia about z-axis (Nms^2)
K_tau = 0.00425;                        % Coefficient of thrust (Ns)
Theta_rest = -25;                       % Theta rest value (degrees)
g = 9.81;                               % Acceleration due to gravity (m/s^2)
L_p = 0.02:0.02:0.2;                    % Roll damping coefficient (Nms), predefined
range
M_q = 0.1:0.085:0.9;                    % Pitch damping coefficient (Nms), prede-
fined range
K_D = 0;                                % Coefficient of drag
w_coll0 = (M*g*l_theta*sin...           % Trim state angular acceleration
(Theta_rest))/(K_tau*l_boom);
K_v = 0.0125*w_coll0*K_tau*l_boom;      % Coefficient of v
n = length(L_p);                        % Array index for Lp for loop
o = length(M_q);                        % Array index for Lp for loop


%% Developing LQ Servo controller for pitch

% Defining the system Matrixes for Pitch
Mqn=  0.440;
Ap = [0, 1, 0; (-M*g*l_theta*cos(Theta_rest))/I_yy, -Mqn/I_yy, (K_tau*l_boom)/I_yy; 0
0 -6];
Bp = [0; 0; 540];
Cp = [1, 0, 0];
Dp = 0;

Abar = [Ap zeros(3,1); -Cp 0];
Bbar = [Bp;0];

Q = Cp'*Cp;

% for loop for the selecting of Integration penalty error
for i = 10:10:600
E(i) = i;
Qbar = [Q zeros(3,1); zeros(1,3) E(i)];
R = 1;
```

```
[Kbar, Pbar, Lamd_bar] = lqr(Abar, Bbar, Qbar, R);                            %
Determining Kbar with lqr function
disp('The value of kbar is: '); disp(Kbar)
disp('solution to P for the associated algebraic Riccati equation '); disp(Pbar)
disp(Lamd_bar)                                                                %
Displaying the stability of the new pitch closed loop system

sys3 = ss(Abar-Bbar*Kbar,[0; 0; 0;1],[Cp 0],0);
[y31, t31, x31] = step(20*pi()/180*sys3,20);

figure(5)                                                                     %
Plotting step response of the system
plot(t31,(20*ones(size(t31))-y31*180/pi())); hold on
title('Response of Pitch for 20 deg Step Input (LQ SERVO)')
grid on
xlabel('Time (s)')
ylabel('Pitch Angle (deg)')

figure (7)                                                                    %
Plotting Voltage vs Time for Pitch
Vcoll = 6/540*(x31(:,3));
plot (t31, Vcoll); hold on
grid on
title('Vcoll vs Time for Pitch (LQ SERVO)')
xlabel('Time (s)')
ylabel('Vcoll(V)')
end

E = 550;                                                                     %
Selected integration error penalty
Qbar = [Q zeros(3,1); zeros(1,3) E];                                         %
Definition of Qbar
R = 1;

[Kbar, Pbar, Lamd_bar] =lqr(Abar, Bbar, Qbar, R);                            %
Determining Kbar with lqr function
disp('The value of kbar is: '); disp(Kbar)
disp('Solution to P for the associated algebraic Riccati equation '); disp(Pbar)
disp(Lamd_bar)                                                               %
Displaying the stability of the new pitch closed loop system

sys3 = ss(Abar-Bbar*Kbar,[0; 0; 0;1],[Cp 0],0);
[y32, t32, x32] = step(20*pi()/180*sys3,20);

figure(6)                                                                    %
Plotting step response of the system for E = 550
plot(t32,(20*ones(size(t32))-y32*180/pi())); hold on
title('Response of Pitch for 20 deg Step Input (LQ SERVO)')
grid on
xlabel('Time (s)')
ylabel('Pitch Angle (deg)')

figure (8)                                                                   %
Plotting Voltage vs Time for Pitch (E = 550)
Vcoll = 6/540*(x32(:,3));
```

```
plot (t32, Vcoll)
grid on
title('Vcoll Vs Time for Pitch (LQ SERVO)')
xlabel('Time (s)')
ylabel('Vcoll(V)')
```

Appendix E – Creating LQE for Roll and Pitch

```
clc;clear;close all
m = 1.15;                                                               % Mass of
rotor assembly (kg)
M = 3.57;                                                               % Mass of
the whole setup (kg)
l_boom = 0.66;                                                          % Length
from pivot point to heli body (m)
l_phi = 0.004;                                                          % Length of
the pendulum for roll axis (m)
l_theta = 0.014;                                                        % Length of
the pendulum for pitch axis (m)
l_h = 0.177;                                                            % Length
from pivot point to the rotor (m)
I_xx = 0.036;                                                           % Moment of
inertia about x-axis (Nms^2)
I_yy = 0.93;                                                            % Moment of
inertia about y-axis (Nms^2)
I_z = 0.93;                                                             % Moment of
inertia about z-axis (Nms^2)
K_tau = 0.00425;                                                        % Coeffi-
cient of thrust (Ns)
Theta_rest = -25;                                                       % Theta
rest value (degrees)
g = 9.81;                                                               % Accelera-
tion due to gravity (m/s^2)
L_p = 0.02:0.02:0.2;                                                    % Roll
damping coefficient (Nms), predefined range
M_q = 0.1:0.085:0.9;                                                    % Pitch
damping coefficient (Nms), predefined range
K_D = 0;                                                                % Coeffi-
cient of drag
w_coll0 = (M*g*l_theta*sin(Theta_rest))/(K_tau*l_boom);                 % Trim
state angular acceleration
K_v = 0.0125*w_coll0*K_tau*l_boom;                                      % Coeffi-
cient of v
n = length(L_p);                                                        % Array in-
dex for Lp for loop
o = length(M_q);                                                        % Array in-
dex for Lp for loop

%% 4a) LQE for the Roll dynamics

% Defining system matrix for Roll
Lpm = median(L_p);
Ar = [0, 1, 0; (-m*g*l_phi)/I_xx, -Lpm/I_xx, (K_tau*l_h)/I_xx; 0 0 -6];
Br = [0; 0; 780];
Cr = [1 0 0];
```

```
Dr  = 0;

% Using Optimum rhoe value otherwise it overshoots
rhoe = 1;
R_wwr = (covar(ss(Ar,Br,Cr,Dr),range(L_p)));
R_vvr = 0.3;
syms q_11 q_12 q_13 q_22 q_33 q_23  lambdar
Qr = [q_11 q_12 q_13 ; q_12 q_22 q_23; q_13 q_23 q_33];

fr = Ar*Qr+Qr*Ar'+Br*R_wwr*Br'-Qr*Cr'*R_vvr^(-1)*Cr*Qr;
sol = solve(fr);
double(sol.q_11);
double(sol.q_12);
double(sol.q_13);
double(sol.q_22);
double(sol.q_23);
double(sol.q_33);

for n=1:8
 set = n;
 q11=double(sol.q_11(n));
 q12=double(sol.q_12(n));
 q13=double(sol.q_13(n));
 q22=double(sol.q_22(n));
 q23=double(sol.q_23(n));
 q33=double(sol.q_33(n));

 Qrr=[q11 q12 q13; q12 q22 q23; q13 q23 q33];
 L=Qrr*Cr'*R_vvr^(-1);
lambda_cl = double(solve(det(lambdar*eye(3)- Ar+L*Cr)));
if lambda_cl < 0
    disp('the set #: '), disp(set);
  disp('Lamda value for Roll: ') ,disp(lambda_cl);
  disp('Lgain value for Roll:'), disp(L)
else
    disp('not stable')
end
end
% After close evaluation it was determined that set 4 of the for loop gave
% negative lambda for the closed-loop.

Lr = (lqr(Ar',Cr',R_wwr, R_vvr))';

Br_aug = [Br eye(3) 0*Br];
Dr_aug = [0 0 0 0 1];
sys4 = ss(Ar-Lr*Cr,Br, Cr,0); % Closed-loop estimator for roll
sysC = ss(Ar, Br_aug, Cr,Dr_aug);
dt = .01;
t = 0:dt:50;
ur = deg2rad(10*ones(size(t)));
[y4,t10,x4] = lsim(sys4,ur,t); % State estimate

figure(9)
plot(t10, y4*180/(pi()));
hold on;
```

```
title 'Roll Estimator'
xlabel('Time (s)')
ylabel('Roll Angle (deg)')
grid on;

%% Phase 4b) - Creating the LQE for the pitch dynamics
clc;clear;close all

m = 1.15;                                                                    % Mass of
rotor assembly (kg)
M = 3.57;                                                                    % Mass of
the whole setup (kg)
l_boom = 0.66;                                                               % Length
from pivot point to heli body (m)
l_phi = 0.004;                                                               % Length of
the pendulum for roll axis (m)
l_theta = 0.014;                                                             % Length of
the pendulum for pitch axis (m)
l_h = 0.177;                                                                 % Length
from pivot point to the rotor (m)
I_xx = 0.036;                                                                % Moment of
inertia about x-axis (Nms^2)
I_yy = 0.93;                                                                 % Moment of
inertia about y-axis (Nms^2)
I_z = 0.93;                                                                  % Moment of
inertia about z-axis (Nms^2)
K_tau = 0.00425;                                                             % Coeffi-
cient of thrust (Ns)
Theta_rest = -25;                                                            % Theta
rest value (degrees)
g = 9.81;                                                                    % Accelera-
tion due to gravity (m/s^2)
L_p = 0.02:0.02:0.2;                                                         % Roll
damping coefficient (Nms), predefined range
M_q = 0.1:0.085:0.9;                                                         % Pitch
damping coefficient (Nms), predefined range
K_D = 0;                                                                     % Coeffi-
cient of drag
w_coll0 = (M*g*l_theta*sin(Theta_rest))/(K_tau*l_boom);                      % Trim
state angular acceleration
K_v = 0.0125*w_coll0*K_tau*l_boom;                                           % Coeffi-
cient of v
n = length(L_p);                                                             % Array in-
dex for Lp for loop
o = length(M_q);                                                             % Array in-
dex for Lp for loop

% Defining Pitch system matrix for median M_q value
Mqm = median(M_q);
Ap = [0, 1, 0; (-M*g*l_theta*cos(Theta_rest))/I_yy, -Mqm/I_yy, (K_tau*l_boom)/I_yy; 0
0 -6];
Bp = [0; 0; 540];
Cp = [1, 0, 0];
Dp = 0;
```

```
% Using Optimum rhoe value otherwise it overshoots
rhoe = 1;
R_wwr = covar(ss(Ap,Bp,Cp,Dp),range(M_q));
R_vvr = 1;
syms q_11 q_12 q_13 q_22 q_33 q_23  lambdar
Qr = [q_11 q_12 q_13 ; q_12 q_22 q_23; q_13 q_23 q_33];

fr = Ap*Qr+Qr*Ap'+Bp*R_wwr*Bp'-Qr*Cp'*R_vvr^(-1)*Cp*Qr;
sol = solve(fr);
double(sol.q_11);
double(sol.q_12);
double(sol.q_13);
double(sol.q_22);
double(sol.q_23);
double(sol.q_33);
for n=1:8
 set = n;
 q11=double(sol.q_11(n));
 q12=double(sol.q_12(n));
 q13=double(sol.q_13(n));
 q22=double(sol.q_22(n));
 q23=double(sol.q_23(n));
 q33=double(sol.q_33(n));

 Qrr=[q11 q12 q13; q12 q22 q23; q13 q23 q33];
 L=Qrr*Cp'*R_vvr^(-1);
lambda_cl = double(solve(det(lambdar*eye(3)- Ap+L*Cp)));
if lambda_cl < 0
    disp('the set #: '), disp(set);
  disp('Lamda value for Roll: ') ,disp(lambda_cl);
  disp('Lgain value for Roll:'), disp(L)
else
    disp('not stable')
end
end


% After close evaluation it was determined that set 4 of the for loop gave
% negative lambda for the closed-loop.

Lr = (lqr(Ap',Cp',R_wwr, R_vvr))';

Br_aug = [Bp eye(3) 0*Bp];
Dr_aug = [0 0 0 0 1];
sys4 = ss(Ap-Lr*Cp,Bp, Cp,0);                                                % Closed-
loop estimator for roll
sysC = ss(Ap, Br_aug, Cp,Dr_aug);
dt = .01;
t = 0:dt:50;
ur = deg2rad(10*ones(size(t)));
[y4,t10,x4] = lsim(sys4,ur,t);                                               % State es-
timating

figure(9)
plot(t10, y4*180/(pi()));
```

```
hold on;
title 'Pitch Estimator'
xlabel('Time (s)')
ylabel('Pitch Angle (deg)')
grid on;
```

Appendix F – Question DOFB for Roll and Pitch

```
clear, clc, close all;
m = 1.15;                                                                  % Mass of
rotor assembly (kg)
M = 3.57;                                                                  % Mass of
the whole setup (kg)
l_boom = 0.66;                                                             % Length
from pivot point to heli body (m)
l_phi = 0.004;                                                             % Length of
the pendulum for roll axis (m)
l_theta = 0.014;                                                           % Length of
the pendulum for pitch axis (m)
l_h = 0.177;                                                               % Length
from pivot point to the rotor (m)
I_xx = 0.036;                                                              % Moment of
inertia about x-axis (Nms^2)
I_yy = 0.93;                                                               % Moment of
inertia about y-axis (Nms^2)
I_z = 0.93;                                                                % Moment of
inertia about z-axis (Nms^2)
K_tau = 0.00425;                                                           % Coeffi-
cient of thrust (Ns)
Theta_rest = -25;                                                          % Theta
rest value (degrees)
g = 9.81;                                                                  % Accelera-
tion due to gravity (m/s^2)
L_p = 0.02:0.02:0.2;                                                      % Roll damp-
ing coefficient (Nms), predefined range
M_q = 0.1:0.085:0.9;                                                      % Pitch damp-
ing coefficient (Nms), predefined range
K_D = 0;                                                                   % Coeffi-
cient of drag
w_coll0 = (M*g*l_theta*sin(Theta_rest))/(K_tau*l_boom);                    % Trim
state angular acceleration
K_v = 0.0125*w_coll0*K_tau*l_boom;                                         % Coeffi-
cient of v
n = length(L_p);                                                           % Array in-
dex for Lp for loop
o = length(M_q);                                                           % Array in-
dex for Lp for loop

%% define system Matrix
Lp=median(L_p);
A1 = [0, 1, 0; (-m*g*l_phi)/I_xx, -Lp/I_xx, (K_tau*l_h)/I_xx; 0 0 -6];
B2 = [0; 0; 780];
C3 = [1 0 0];
D4  = 0;
```

```
%% Define LQ Servo
Abar = [A1 zeros(3,1); -C3 zeros(1)];
Bbar = [B2; 0];
Cbar = [C3, 0];


%% Form statistical Matrix from LQ servo leccture
rho = 2; Q = C3'*C3; R = rho*eye(1);
E = 550; Qbar = [Q zeros(3,1); zeros(1,3) E];
alpha = 0.2; Ra = -alpha*C3';
Rww = (covar(ss(A1,B2,C3,D4),range(L_p)));  Rvv = 0.3;

%% Calculate LQ servo control plane
Kbar = lqr(Abar, Bbar, Qbar, R);
L = lqr(A1',C3',B2*Rww*B2',Rvv)';
K = Kbar(1:3);
KI = Kbar(4:4);

%% form closed loop matrix with servo and sevrvo a
Abarcl = Abar - Bbar*Kbar;
Bbarcl = [zeros(3,1); eye(1)];
BbarclR = [-B2*K*Ra; eye(1)];

%% form closed loop matrix for DOFB integrator
Acheckcl = [A1 -B2*Kbar;[L*C3; -C3] [A1-L*C3-B2*K -B2*KI; 0 0 0 0]];
Bcheckcl = [-B2*K*Ra; [-B2*K*Ra; 1]];
Ccheckcl = [C3 C3*0 0];

%% from cl matrix contorller + estimator with Nbar
Anbarcl = [A1 -B2*K; L*C3 A1-B2*K-L*C3];
Bnbarcl = [B2;B2];
Cnbarcl = [C3 C3*0];
Nbar = -inv(Cnbarcl*inv(Anbarcl)*Bnbarcl);

%% from closed loop system
S1cl = ss(Abarcl,Bbarcl,Cbar,0);
S1clR = ss(Abarcl,BbarclR,Cbar,0);
S1clDOFB = ss(Acheckcl,Bcheckcl,Ccheckcl,0);
S1clN = ss(Anbarcl,Nbar*Bnbarcl,Cnbarcl,0);

%% simulate the closed loop models
[y1, t1, x1] = step(S1cl,10);
[y2, t2, x2] = step(S1clR,10);
[y3, t3, x3] = step(S1clDOFB,10);
[y4, t4, x4] = step(S1clN,10);

%% plot results
yline(0,'k-.','DisplayName','Setpoints','Linewidth',2);
hold on
plot(t1, ones(size(t1))-y1, 'r','DisplayName','LQR (no $\bar{R}$)', 'LineWidth',2);
plot(t2, ones(size(t2))-y2, 'b','DisplayName','LQR (with $\bar{R}$)', 'LineWidth',2);
plot(t3, ones(size(t3))-y3, 'c','DisplayName','DOFB (Check $\bar{R}$)', 'Lin-
eWidth',2);
```

```
plot(t4, ones(size(t4))-y4, 'g','DisplayName','Controller + Estimator (with
$\bar{N}$)', 'LineWidth',2);

legend("location",'best','Interpreter','latex');

xlabel("Time (s)");
ylabel('Roll Angle (deg)');

grid on
set(gca, 'GridLineStyle', '--');
set(gca, 'Fontsize', 16, 'FontName', 'Times', 'ycolor', 'k');
set(gca, 'LooseInset', get(gca, 'TightInset'));
%% plot results
figure(2)
yline(0.46,'k-.','DisplayName','Setpoints','Linewidth',2);
hold on
plot(t1, 6/780*(x1(:,3)), 'r','DisplayName','LQR (no $\bar{R}$)', 'LineWidth',2);
plot(t2, 6/780*(x2(:,3)), 'b','DisplayName','LQR (with $\bar{R}$)', 'LineWidth',2);
plot(t3, 6/780*(x3(:,3)), 'c','DisplayName','DOFB (Check $\bar{R}$)', 'LineWidth',2);
plot(t4, 6/780*(x4(:,3)), 'g','DisplayName','Controller + Estimator (with
$\bar{N}$)', 'LineWidth',2);

legend("location",'best','Interpreter','latex');

xlabel("Time (s)");
ylabel('Voltage (V)');

grid on
set(gca, 'GridLineStyle', '--');
set(gca, 'Fontsize', 16, 'FontName', 'Times', 'ycolor', 'k');
set(gca, 'LooseInset', get(gca, 'TightInset'));



clear, clc, close all;
m = 1.15;                                                                    % Mass of
rotor assembly (kg)
M = 3.57;                                                                    % Mass of
the whole setup (kg)
l_boom = 0.66;                                                               % Length
from pivot point to heli body (m)
l_phi = 0.004;                                                               % Length of
the pendulum for roll axis (m)
l_theta = 0.014;                                                             % Length of
the pendulum for pitch axis (m)
l_h = 0.177;                                                                 % Length
from pivot point to the rotor (m)
I_xx = 0.036;                                                                % Moment of
inertia about x-axis (Nms^2)
I_yy = 0.93;                                                                 % Moment of
inertia about y-axis (Nms^2)
I_z = 0.93;                                                                  % Moment of
inertia about z-axis (Nms^2)
K_tau = 0.00425;                                                             % Coeffi-
cient of thrust (Ns)
```

```
Theta_rest = -25;                                                       % Theta
rest value (degrees)
g = 9.81;                                                               % Accelera-
tion due to gravity (m/s^2)
L_p = 0.02:0.02:0.2;                                                    % Roll
damping coefficient (Nms), predefined range
M_q = 0.1:0.085:0.9;                                                    % Pitch
damping coefficient (Nms), predefined range
K_D = 0;                                                                % Coeffi-
cient of drag
w_coll0 = (M*g*l_theta*sin(Theta_rest))/(K_tau*l_boom);                 % Trim
state angular acceleration
K_v = 0.0125*w_coll0*K_tau*l_boom;                                      % Coeffi-
cient of v
n = length(L_p);                                                        % Array in-
dex for Lp for loop
o = length(M_q);                                                        % Array in-
dex for Lp for loop

%% define system Matrix for Pitch
A = [0, 1, 0; (-M*g*l_theta*cos(Theta_rest))/I_yy, -0.9/I_yy, (K_tau*l_boom)/I_yy; 0
0 -6];
B = [0; 0; 540];
C = [1, 0, 0];
D = 0;
%% Define LQ Servo
Abar = [A zeros(3,1); -C zeros(1)];
Bbar = [B; 0];
Cbar = [C, 0];


%% Form statistical Matrix from LQ servo leccture
rho = 100; Q = C'*C; R = rho;
E = 550; Qbar = [Q zeros(3,1); zeros(1,3) E];
alpha = 0.02; Ra = -alpha*C';
Rww = covar(ss(A,B,C,D),range(M_q)); Rvv = 1;

%% Calculate LQ servo control plane
Kbar = lqr(Abar, Bbar, Qbar, R);
L = lqr(A',C',B*Rww*B',Rvv)';
K = Kbar(1:3);
KI = Kbar(4:4);

%% form closed loop matrix with servo and sevrvo a
Abarcl = Abar - Bbar*Kbar;
Bbarcl = [zeros(3,1); eye(1)];
BbarclR = [-B*K*Ra; eye(1)];

%% form closed loop matrix for DOFB integrator
Acheckcl = [A -B*Kbar;[L*C; -C] [A-L*C-B*K -B*KI; 0 0 0 0]];
Bcheckcl = [-B*K*Ra; [-B*K*Ra; 1]];
Ccheckcl = [C C*0 0];

%% from cl matrix contorller + estimator with Nbar
Anbarcl = [A -B*K; L*C A-B*K-L*C];
```

```
Bnbarcl = [B;B];
Cnbarcl = [C C*0];
Nbar = -inv(Cnbarcl*inv(Anbarcl)*Bnbarcl);

%% from closed loop system
S1cl = ss(Abarcl,Bbarcl,Cbar,0);
S1clR = ss(Abarcl,BbarclR,Cbar,0);
S1clDOFB = ss(Acheckcl,Bcheckcl,Ccheckcl,0);
S1clN = ss(Anbarcl,Nbar*Bnbarcl,Cnbarcl,0);

%% simulate the closed loop models
[y1, t1, x1] = step(S1cl,10);
[y2, t2, x2] = step(S1clR,10);
[y3, t3, x3] = step(S1clDOFB,10);
[y4, t4, x4] = step(S1clN,10);

%% plot results
figure(1)
yline(0,'k-.','DisplayName','Setpoints','Linewidth',2);
hold on
plot(t1, ones(size(t1))-y1, 'r','DisplayName','LQR (no $\bar{R}$)', 'LineWidth',2);
plot(t2, ones(size(t2))-y2, 'b','DisplayName','LQR (with $\bar{R}$)', 'LineWidth',2);
plot(t3, ones(size(t3))-y3, 'c','DisplayName','DOFB (Check $\bar{R}$)', 'Lin-
eWidth',2);
plot(t4, ones(size(t4))-y4, 'g','DisplayName','Controller + Estimator (with
$\bar{N}$)', 'LineWidth',2);

legend("location",'best','Interpreter','latex');

xlabel("Time (s)");
ylabel('Pitch Angle (deg)');

grid on
set(gca, 'GridLineStyle', '--');
set(gca, 'Fontsize', 16, 'FontName', 'Times', 'ycolor', 'k');
set(gca, 'LooseInset', get(gca, 'TightInset'));

%% plot results
figure(2)
yline(1.9,'k-.','DisplayName','Setpoints','Linewidth',2);
hold on
plot(t1, 6/540*(x1(:,3)), 'r','DisplayName','LQR (no $\bar{R}$)', 'LineWidth',2);
plot(t2, 6/540*(x2(:,3)), 'b','DisplayName','LQR (with $\bar{R}$)', 'LineWidth',2);
plot(t3, 6/540*(x3(:,3)), 'c','DisplayName','DOFB (Check $\bar{R}$)', 'LineWidth',2);
plot(t4, 6/540*(x4(:,3)), 'g','DisplayName','Controller + Estimator (with
$\bar{N}$)', 'LineWidth',2);

legend("location",'best','Interpreter','latex');

xlabel("Time (s)");
ylabel('Voltage (V)');

grid on
set(gca, 'GridLineStyle', '--');
set(gca, 'Fontsize', 16, 'FontName', 'Times', 'ycolor', 'k');
```

```
set(gca, 'LooseInset', get(gca, 'TightInset'));
```

Appendix G – DOFB into Simulink

```
% Item 7 simlink model
clc,clear
m=1.15;
M=3.57;
lboom=0.66;
lroll=0.004;
lpitch=0.014;
lh=0.177;
Ixx=0.036;
Iyy=0.93;
Kthrust=4.25e-3;
thetarest=-25;
g=9.81;
Lp=[0.02:.02:0.2];
Mq=[0.1:0.1:0.9];
Lpm= median(Lp);
Mqm = median(Mq);
Iz=0.93;
Kdrag=0;
wcoll0=M*g*lpitch*sin(thetarest)/(Kthrust*lboom);
Kv=0.0125*wcoll0*Kthrust*lboom;


%% Matrix of the whole system
A=[0,0,0,1,0,0,0,0;
    0,0,0,0,1,0,0,0;
    0,0,0,0,0,1,0,0;
    -m*g*lroll/Ixx,0,0,-Lpm/Ixx,0,-Kv*lh/Ixx,Kthrust*lh/Ixx,0;
    0,-M*g*lpitch*cos(thetarest)/Iyy,0,0,-Mqm/Iyy,-Kv*lboom/Iyy,0,Kthrust*lboom/Iyy;
    lboom*Kthrust*wcoll0/Iz,0,0,0,0,-Kdrag*lboom/Iz,0,0;
    0,0,0,0,0,0,-6,0;
    0,0,0,0,0,0,0,-6];


B=[0,0;
    0,0;
    0,0;
    0,0;
    0,0;
    0,0;
    780,0;
    0,540];


C=[1,0,0,0,0,0,0,0;
    0,1,0,0,0,0,0,0;
    0,0,1,0,0,0,0,0];
```

```
%%  from item 5 roll
Aroll=[ 0    1.0000         0;
   -1.2535   -3.0556    0.0209;
         0         0   -6.0000];


Broll=[0;
     0;
   780];
Croll=[  1     0     0];
Kroll= [4.4291,1.0872,0.0031];
KIroll=-4.4721;
Rroll= 1;
Lroll=  [0.3901;
    0.0761;
   12.1286];


%% from item pitch
Apitch=[ 0    1.0000         0;
   -0.5226   -0.5376    0.0030;
         0         0   -6.0000];
Bpitch=[0;
     0;
   540];
Cpitch=[  1     0     0];
Kpitch= [24.6417   13.5462    0.0055];
KIpitch=-22.3607;
Rpitch= 1;
Lpitch=  [0.0131;
    0.0001;
    0.1905];
```

Appendix H – Response for 20 deg Step Input for Roll (LQR)

Figure 9

Response for 20 deg Step Input for Roll (LQR)

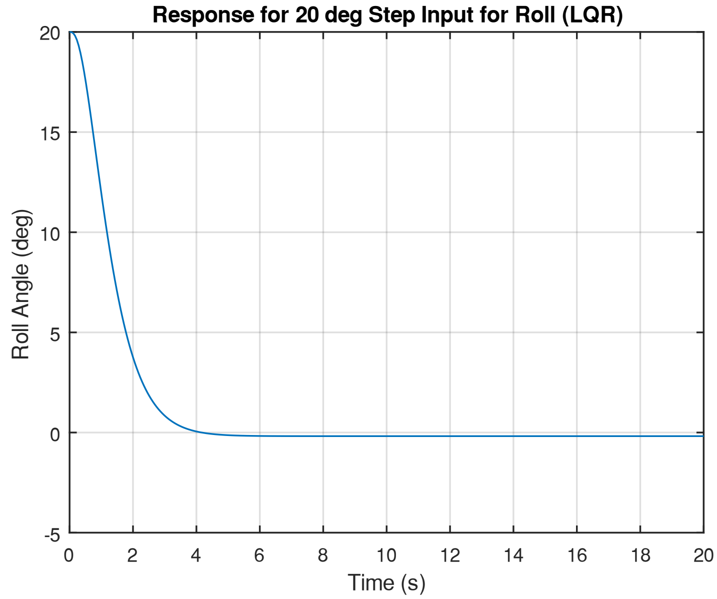


Fig. (9) stimulates the step response for a 20-degree step response at rho 1.3.

Figure 10

Response for Pitch for 20 deg Step Input

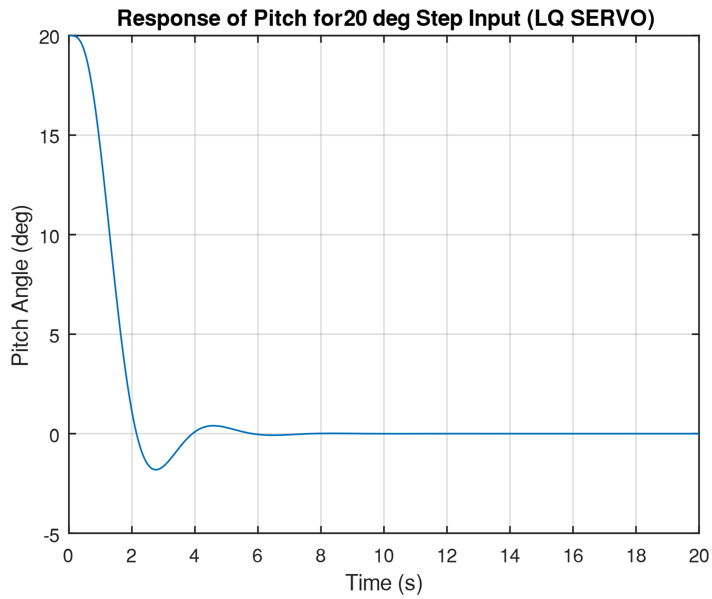


Fig. (11) shows the stimulated response for 20-degree step input for Pitch at E = 550 (LQ Servo)

Appendix I – Vcyc and Vcoll Vs Time for Roll (LQR)

Figure 11

Vcyc Vs Time for Roll (LQR)

V_{cyc} vs Time for Roll

at rho = 1.3

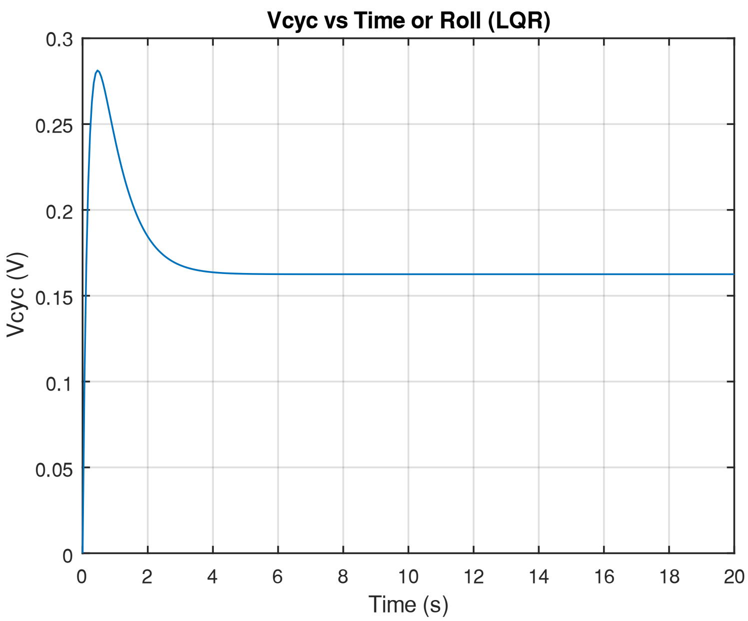


Figure 12

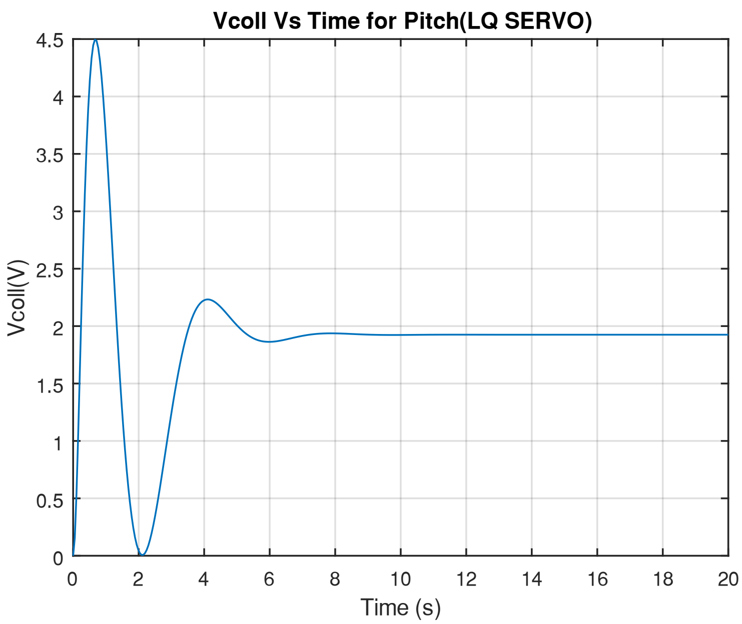
Voltage (V_{coll}) Vs Time Graph

V_{coll} vs Time for Roll

at

Integration

error E = 550



Appendix J –Voltage(V) Vs Time(s) DOFB Roll and Pitch

Figure 13

Voltage Vs Time Graph (DOFB) Roll

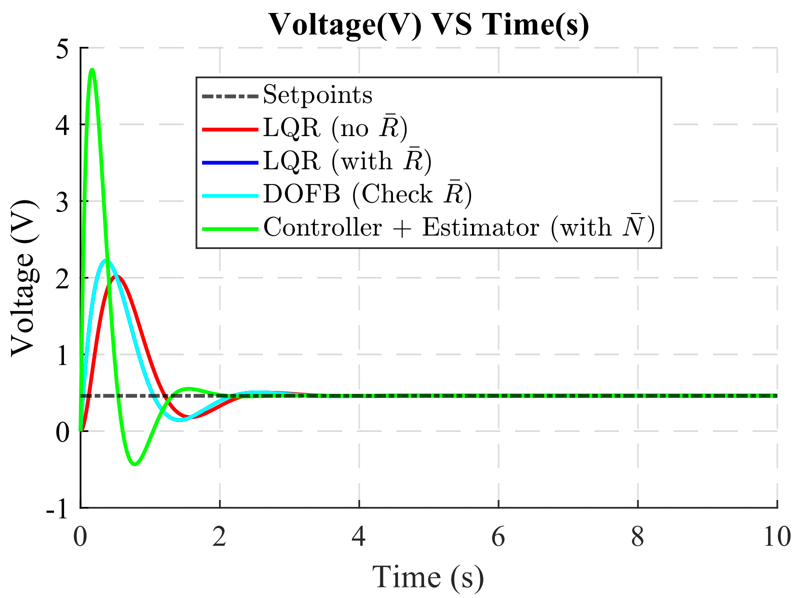


Figure (8) shows the voltage vs time graph for DOFB that is approximately reaching to steady state after 2.5 secs

Figure 14

Voltage Vs Time Graph DOFB Pitch

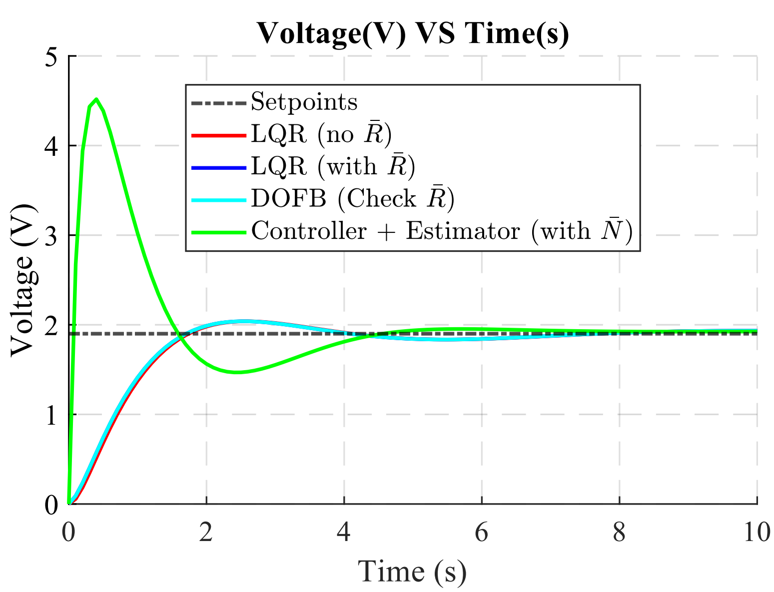


Figure (10) shows the voltage vs time graph for DOFB that is approximately reaching to steady state after 5 secs

Appendix K – Pitch and Roll Angle(deg) Vs Time(s) Simulink Graph

Figure 15

Pitch Angle(deg) vs Time for Simulink Graph

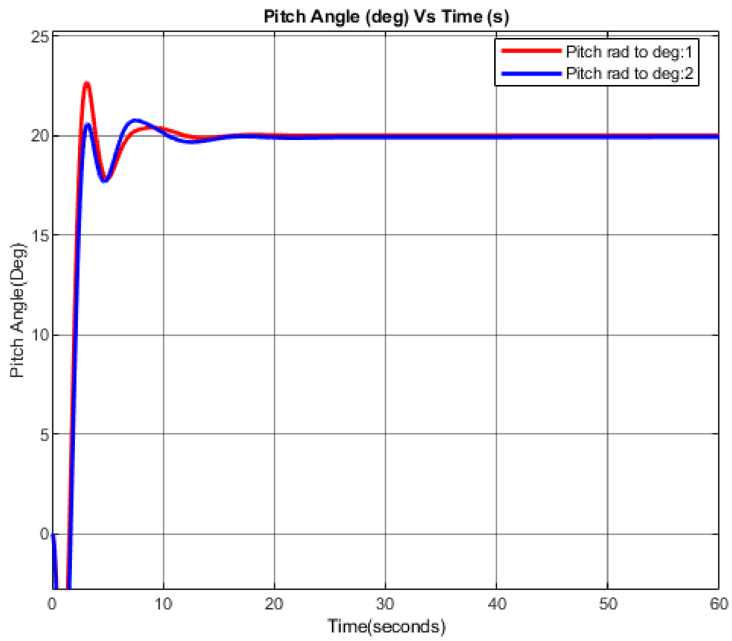
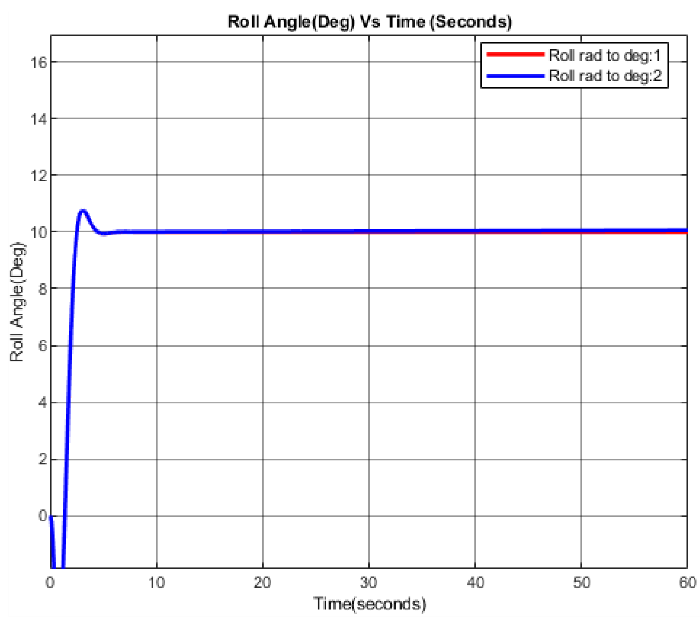


Figure 16

Roll Angle(deg) vs Time for Simulink Graph



Appendix L – V_{coll}(V) and V_{cyc}(V) Vs Time(s) Simulink Graph

Figure 17

V_{coll}(V) Vs Time(s) Simulink Graph

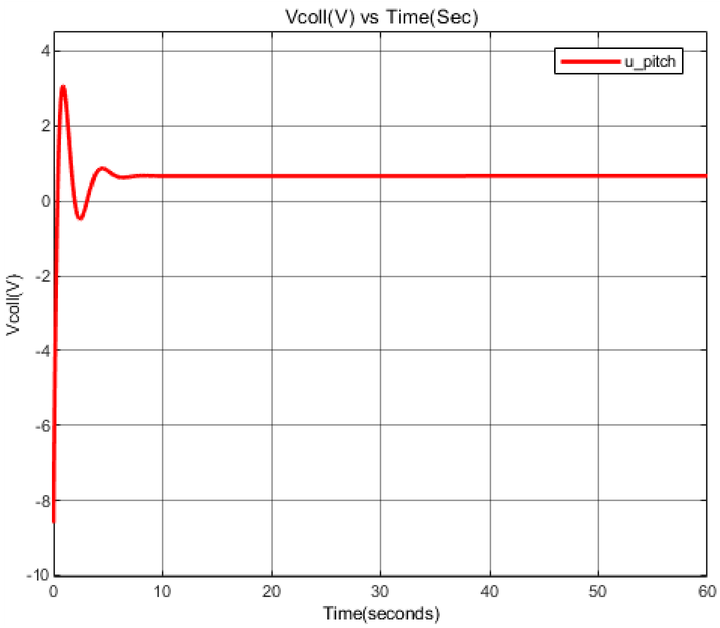
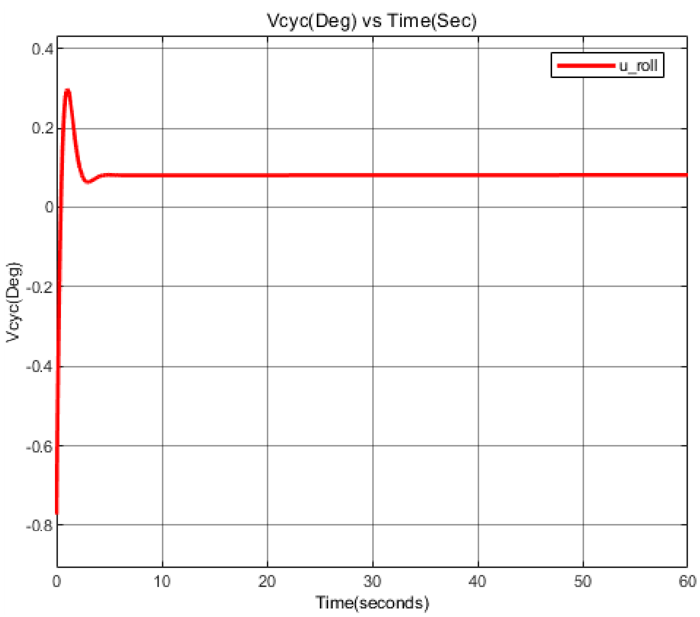


Figure 18

V_{cyc}(V) Vs Time(s) Simulink Graph



Appendix M – Roll and Pitch Controller