DESIGINING A CONTROL SYSTEM FOR THE AUTONOMOUS SMALL SCALE HELICOPTER

A Thesis

Submitted to

the Temple University Graduate Board

In Partial Fulfillment

Of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

By Satish Sajja

August 2010

Dr. John J. Helferty

Thesis Advisor

Dr. Dennis Silage

Committee Member

Dr. Li Bai

Committee Member

# COPYRIGHT NOTIFICATION

# ABSTRACT

Designing a control system for the autonomous small scale helicopter mainly consists of two parts. First, Identify the system model (Helicopter model) and the second part is designing the compensator. Modeling the helicopter can be achieved by either first principles modeling or black box modeling. Based on the Model obtained, different controlling techniques can be used to design a compensator to achieve stability.

This thesis presents an easy and accurate method to design the control system for the autonomous small scale helicopter. The helicopter system is identified using experimental data (black box modeling) with the help of MATLAB system identification tool box. And then the compensator is designed by using the MATLAB control system tool box. Various controlling methods like PID (Proportional Integral and Derivative), LQG (Linear Quadratic Gaussian) and IMC (Internal Model Controller) are compared while tuning the system to achieve high stability. The stability of the system is measured by its step response. Finally the system is simulated and its functionality is verified in MATLAB simulink.

Since this thesis is focusing on the stable hovering state of the helicopter, we developed a simulated control system for the swash plate of the helicopter. Swash plate is mainly responsible for the helicopter's dynamics in rotation about X body axis (Roll) and rotation about Y body axis (Pitch). So the control system is developed and simulated in order to control the roll and pitch angles of the helicopter. The results show that LQG method gives the better step response than IMC, PID methods.

# ACKNOWLEDGEMENTS

The author would like to thank the following people who have each made this thesis possible:

My thesis supervisor, Dr. John J Helferty: for guidance and support from inception to completion, and for giving me the academic freedom to express my opinion in research findings or differences in approaches.

My thesis committee members, Dr. Dennis Silage and Dr. Li Bai: for their excellent support, encouragement and guidance.

Senior Design team Rafael Ortiz, Jeremiah Stewart, Marcos Hanna, Hasan Albahrani, who made the helicopter platform and Testing (JIG) platform ready.

Temple University, College of Engineering faculty: for their support and guidance throughout my master's degree program.

Parents and Friends: for their love and support.

Thank you all!!!

# TABLE OF CONTENTS PG NO

# LIST OF FIGURES

**PG NO**

# LIST OF TABLES

**PG NO**

# CHAPTER 1

# BACKGROUND

This chapter provides the background details of autonomous helicopter and proposed autonomous helicopter system and its functionality.

## 1.1 Autonomous helicopter

An autonomous helicopter is a self operated system, making decisions and finishing tasks without human input.

Research in the field of small scale autonomous helicopter has become popular because of its applications in various fields. For example, autonomous helicopters could be used for agricultural field observation and chemical distribution. They could be useful for search and rescue operations, where they can enter remote and difficult places where the people could not reach. The autonomous helicopters could then start the search operation to locate victims of natural disasters and accidents, and then send information to the ground station for rescue. These mini helicopters can be used as fireguards by patrolling over difficult areas like forest. Other applications include surveillance, traffic observation in cities, law enforcement (where vision-guided autonomous helicopters can fly overhead to aid the police in dangerous high-speed chases or criminal search operations), cartography (where small scale helicopters can be used for automatic landscape photographing making cost effective compared with traditional aerial snap shooting).

These applications require a high level of autonomy to avoid causing loss or harm to people. Because of the limited level of autonomy and high cost, the contemporary small scale helicopters are not being extensively used to their potential. These limitations can be reduced if accurate and low cost control systems can be designed.

For autonomous helicopters, the autopilot is the control system which handles the flying and decision making. In order to make the decisions, the autopilot receives information about helicopter's state and orientation. This information is provided by various sensors like an Ultrasonic level sensor, Inertial Measurement Unit (IMU) and Global Positioning system (GPS) etc. To process this information a processing platform like Field Programmable Gate Array (FPGA) [1] or Microprocessors are used.

Factors like mechanical design, integrating various sensors, software architecture, processing platform, cost and size of the autopilot makes the autopilot design a challenging task for researchers. This is a multidisciplinary project. It covers many disciplines such as hardware design; digital signal and image signal processing (to decode the information sent by the helicopter) [2, 3]; mathematical modeling and system identification [4, 5] to control the helicopter; networking and communication for telemetry; incorporating artificial intelligence techniques [6] to enhance the level of autonomy. This is a very interesting and challenging area for researchers from various fields. Since this is a relatively fresh field, lots of researchers are working towards better improvements and new approaches.

## 1.2 Proposed system

The proposed total system architecture is shown in the figure 1. It consists of four sections; they are Sensor data parsing, Autopilot, Flight control, and PWM generation. The four main sensors provide the information about the helicopter in all states. Ultrasonic level sensor provides the data about the height of the helicopter from the ground, IMU (Inertial Measurement Unit) provides the orientation data, and GPS (Global Positioning system) provides the data about helicopter's position.

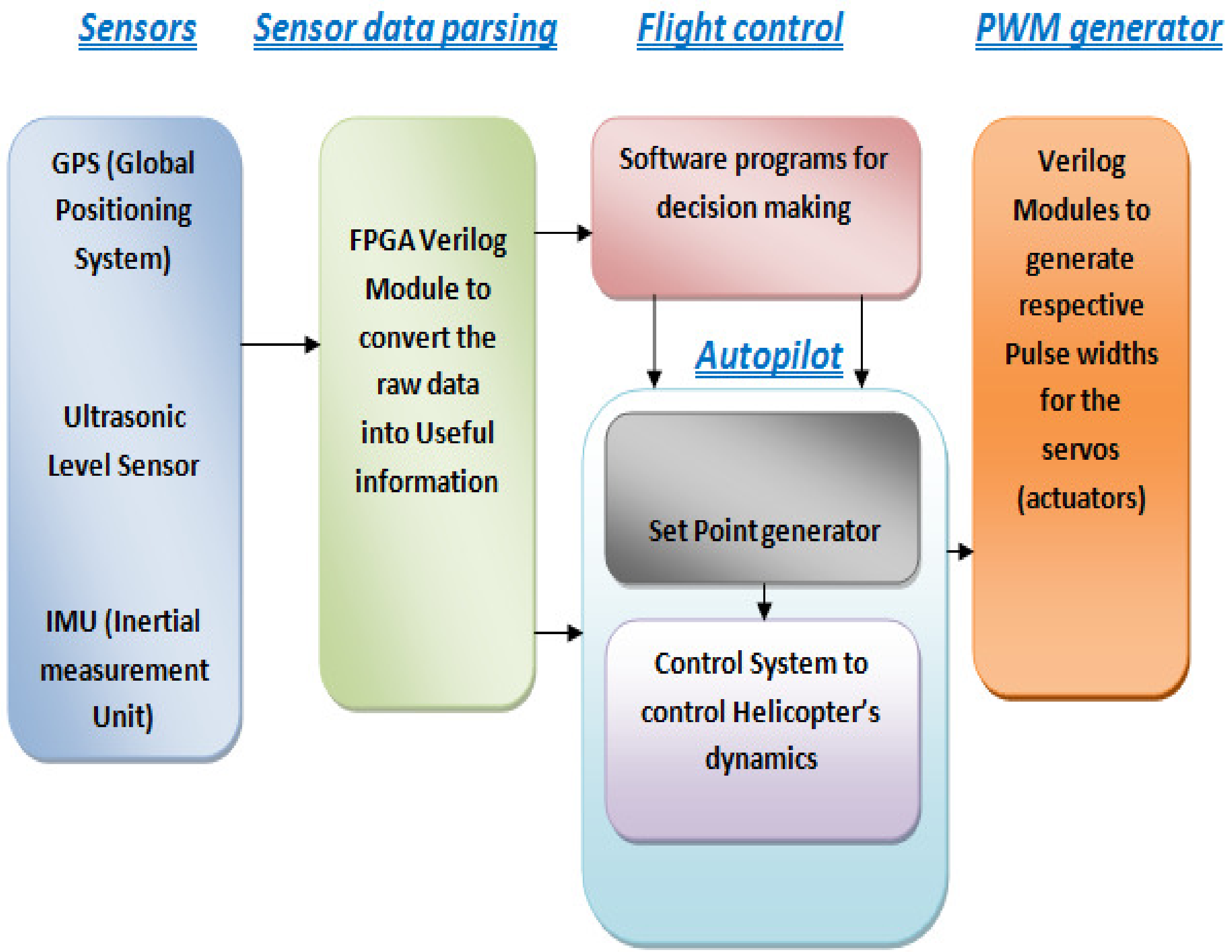


Figure 1: Proposed autonomous helicopter system

The each section of the figure 1 is briefly explained as follows.

**Sensor data parsing**

The raw data acquired from the input sensors ultrasonic level sensor, IMU and GPS will be converted into the useful units according to conversion factors. These conversion factors can be found in the sensor output data specifications.

**Flight control**

The flight control system sends the control signals to the autopilot system. Basically the flight control system tells the autopilot system about the state (lift off, hovering, forwarding, and landing) of the helicopter system. The execution commands are stored in this system according the task. These commands direct the autopilot system what to do and which state to get according to the navigation information given by GPS. These execution commands normally stored before initializing the task and can be changed according to the requirements during the task using wireless Bluetooth technology.

**Autopilot system**

The autopilot system is the heart of the helicopter control system which drives the helicopter like human pilot. It has mainly two parts, one is set point generator and other one is control system. The set point generator generates the set points in each state according the state information given by the flight control system. The control system compares the input sensor data (IMU, ultrasonic sensor data) with the set points to generate the four input setting values for throttle, roll, pitch, and yaw. This autopilot control system consists of controllers to control each value of throttle, roll, pitch, yaw settings. These four values will be directed to the PWM generator to activate the servos.

**PWM (Pulse Width Modulation) generator**

The PWM generator generates required pulse width helicopter control signals based on the four input values for throttle, roll, pitch, yaw settings

## 1.3 Current State

The current state of the project is explained section wise as follows.

Helicopter and Testing platform:

- The helicopter system has been assembled and Jig platform (platform which simulates safe flight) is ready for testing the helicopter [see Appendix-B for details].

Sensors & Processing Platform:

- Inertial Measurement Unit (Orientation sensor) and Ultrasonic level sensors have been acquired [see Appendix-B for details].
- Spartan-3E FPGA board has been acquired for testing processing platform [see Appendix-B for details].

Autopilot and Control system:

- A custom based circuit board which integrates and supplies required power to all the sensors is designed to receive the data from the IMU while testing the helicopter.
- Bluetooth modules are acquired to establish a wireless link between the helicopter and the host computer.
- Verilog modules have been designed to transmit the commands to the IMU and to receive the respective data.
- The Verilog codes to convert the acquired data into useful information (sensor data parsing) are designed. Matlab code is written to receive the information at the host end [see Appendix-A for details].
- The helicopter has been tested to acquire the experimental data for the roll, pitch angles and respective servo pulse widths. This experimental data is used to identify the helicopter's model and to design the control system.
- The Helicopter's model is identified using the system identification tool box of MATLAB to control roll and pitch angles of the helicopter.
- The controller transfer function is derived for the identified models using control system tool box of MATLAB and simulated the control system in simulink to verify the step response.

## 1.4 Outline

This thesis document is organized as follows. ***Chapter 1*** describes the background details of this project. ***Chapter 2*** discusses the problem statement and briefly explains the approach to solve the problem. This chapter also provides the objective and related work of this thesis. ***Chapter 3*** gives the introduction to system identification and various controlling techniques. ***Chapter 4*** explains the implementation details of the proposed control system. ***Chapter 5*** provides the results of this project followed by ***conclusion future work, reference and appendices.***

# CHAPTER 2

# PROBLEM STATEMENT

The problems involved in designing the accurate controller are explained in this chapter. Our approach to design the controller is explained briefly in this chapter and the complete details are provided in the chapter 4. The objective and related work is also provided in this chapter.

## 2.1 Statement of the problem

The control system for an autonomous small scale helicopter has been designed using various methods. During the period of 1990s the classical control systems like proportional integral derivative (PID) feedback controllers have been used. The parameters of these control systems were usually tuned empirically. This trial and error approach to design an acceptable control system however is not agreeable with complex control systems like helicopter. This trial and error approach is very complex, time consuming and expensive. The helicopter needs to be continually tested and controller has to be designed again and again until high stability is achieved. These flight tests may turn out to be dangerous when controllers become unstable consequently expensive sensors can be damaged.

The other drawbacks of these techniques are the need for extensive knowledge of helicopter flight mechanics to develop a mathematical model of the helicopter system and the lack of sufficient accuracy of the extracted linear models needed for controller design. These models require difficult and inefficient manual tuning of many physical parameters. The models developed from first principles method suffer from low accuracy [7, 8]. Without accurate modeling for a helicopter system, design, evaluation, and implementation of advanced control systems for complex autonomous flight operations is impractical.

In order to overcome these problems an accurate model of the helicopter should be derived. According to the derived helicopter model, an inexpensive and sophisticated controller has to be designed. Our approach is focused on eliminating these drawbacks by using advanced techniques to promise accurate and inexpensive control system.

## 2.2 Objective

The goal of this thesis research is to develop a control system to control the dynamics of the helicopter that are created by the swash plate. Swash plate of the helicopter mainly creates two kinds of helicopter dynamics. First one is rotation about X body axis (roll) and second one is rotation about Y body axis (pitch). In a nutshell our goal is to develop an accurate control system to control the roll and pitch angles of the Helicopter.

## 2.3 Approach

Since the first principles method and the classical controlling, manual tuning methods are not promising accurate models and high stability; we followed advanced techniques to identify the model of the helicopter and to tune the helicopter's system to achieve stability. This approach is very simple and inexpensive. The only thing this approach requires is the knowledge about system identification tool box and graphical tuning technique provided by control system tool box of Mathworks Company. This approach mainly consists of four steps and briefly explained as follows.

**Experimental data gathering**

Once the inputs and outputs of the system are known, the helicopter will be tested to acquire the inputs and outputs data.

**System identification**

Using the experimental data values, the helicopter's system model will be identified. This method is called as black box modeling. Mathworks system identification tool box is used to serve the purpose. This technique is fast and accurate compared to first principles modeling. The output of this step is the transfer function of the helicopter system.

**Controller design**

Once the transfer function of the helicopter system is derived, the controller will be designed to compensate the error.  Advanced controlling techniques like LQG (Linear Quadratic Gaussian), IMC (Internal Model Controller) are used along with old techniques like PID (Proportional Integral Derivative) to design the controller. Advanced tuning methods like graphical tuning are used to tune the system to achieve stability.  Mathworks control system tool box facilitates these controller design and graphical tuning techniques. This graphical tuning method overcomes the problems of manual tuning and saves a lot of time. The output of this step is the transfer function of the controller.

**Simulation**

The whole control system is simulated using simulink to verify the response of the system to step input.

## 2.4 Related work

Because of its wide research scope, many people are involved in the research field of small scale autonomous aerial vehicles.

A more suitable parameter identification approach [9] is used which combines both system identification and first principle modeling techniques to obtain a high fidelity linear model for hover and near hover flight conditions. A flight control law, which is based on the linear quadratic Gaussian (LQG) method and dynamic inversion approach, has been designed for the realization of automatic control.

Various methods have been used to implement accurate control systems for small scale autonomous helicopters. Some of the methods are S-CDM (Squared Coefficient Diagram Method) and optimal tracking controller method. The S-CDM approach is used for hovering control of small-scale helicopter. A practical tracking controller is implemented using Optimal tracking controller method [10, 11, and 12].

A novel modeling technique that integrates first-principles and system identification techniques through the use of global optimization methods in the frequency domain has been developed. This research suggests that the models obtained with the integrated technique are adequate for the design of high-performance control systems and the accurate simulation of complex autonomous missions [13].

A Mathematical model of a small rotorcraft has been developed using Matlab symbolic tool box and developed a methodology for state estimation using Extended Kalman Filter. Control algorithms based on PI (proportional Integral), LQG (Linear Quadratic Gaussian) and SDRE (State-Dependent Riccati Equation [14]) approaches are used to implement the control system [15].

Diversity in unmanned vehicle designs and control as well as diversity in existing autopilot systems [16, 17, 18, 19, 20] lead to major compatibility issues among different platforms. A new approach [21] is used to design of a new modular, flexible, easy to integrate autopilot, suitable for multiple platforms. This design is fully integrated with MATLAB/Simulink allowing for both simplified high level programming and hardware in the loop capabilities and provides a flexible

FPGA based platform that maximizes parallel processing and allows for analog signal conditioning that can be modified through digital communication from FPGA processor.

# CHAPTER 3

# INTRODUCTION

A brief introduction to system identification method and the different controlling methods are presented in this chapter.

## 3.1 System Identification

The system's model of a dynamic system can be identified mainly by two ways. They are white box or first principles modeling and black box modeling.

White box models are the results of diligent and extensive physical modeling from first principles. These first principles are developed from known mathematical equations. This method consists of writing down all known relationships between relevant variables. This method is very complex if the model consists of complex dynamics and relationships among the variables. Vast knowledge of system's dynamics is required to develop an accurate model for the complex systems like helicopters.

Without writing down mathematical equations and without any knowledge about the relationships among the variables, the system's model can be identified using experimental data that is input- output data of the system. This technique is commonly known as black box modeling. The introduction of advanced software methods (MATLAB system identification) made this technique simple even for systems having complex dynamics.

The Mathworks Company provides software to construct mathematical models of dynamic systems from measured input-output data. This data-driven approach helps to describe systems that are not easily modeled from first principles. Models obtained with system identification toolbox are well suited for simulation, prediction, and control system design using products such as simulink, control system toolbox. The key features of this tool box are

- Identifies linear and nonlinear models from time- and frequency-domain data
- Simplifies identification of first-, second-, and third-order continuous-time models
- Analyzes measured data and advises on data quality, required preprocessing, and presence of feedback or nonlinearities
- Provides data processing tools for filtering, and reconstructing missing data
- Provides Simulink blocks for simulating identified models and transferring data to and from the MATLAB workspace
- Provides an interface for using linear models in Control System Toolbox.

## 3.2 Controlling methods

Designing the controller for the system's model can be done using the control system tool box of the Matlab. The control system tool box basically provides three controlling methods. They are PID (proportional, Integral, Derivative), IMC (Internal model controller) and LQG (Linear quadratic Gaussian). The graphical tuning technique of the control system tool box facilitates easy tuning operation to obtain desired step response. The details of these three methods are explained as follows.

**PID (proportional Integral derivative) controller**

The transfer function of the PID controller looks like the following:

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

Where Kp = Proportional gain, KI = Integral gain and Kd = Derivative gain

A proportional controller (Kp) will have the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error. An integral control (Ki) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (Kd) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response.

**IMC (Internal model controller)**

The Internal Model Controller works on internal model principle. This principle states that "control can be achieved only if the control system encapsulates either implicitly or explicitly, some representation of the process to be controlled. In other words, perfect control is theoretically possible if the control scheme has been developed based on the model of the process.

The general structure of Internal Model Controller (IMC) is shown in Figure 2.

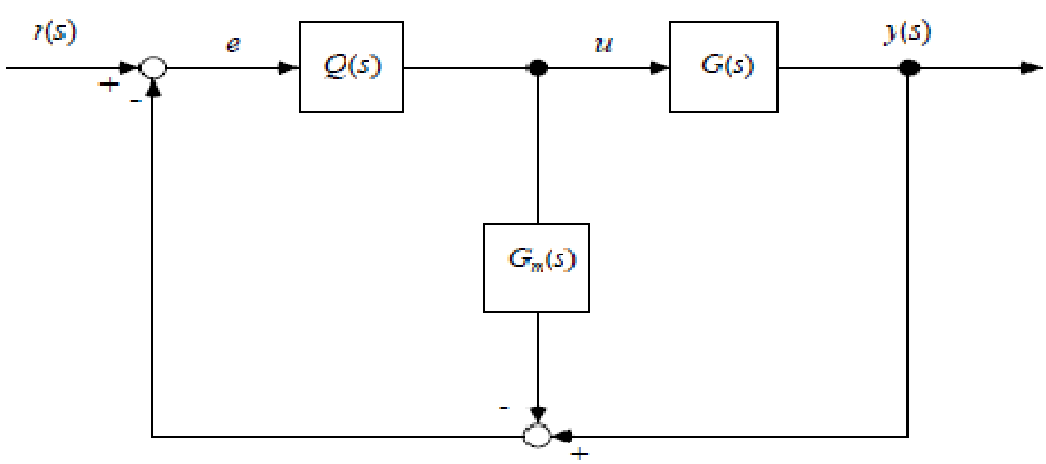


Figure 2: Structure of Internal Model Controller

Here G(s) is the single-input/single-output plant, Gm(s) is the nominal plant, and Q is the controller, r is the reference input, y is the output and u is the control input.

**Linear-Quadratic-Gaussian (LQG) controller**

Linear-Quadratic-Gaussian (LQG) control is a type of optimal control technique. It concerns uncertain linear systems disturbed by additive white Gaussian noise, incomplete state information (i.e. not all the state variables are measured and available for feedback) also disturbed by additive white Gaussian noise and quadratic costs.

The LQG controller is simply the combination of a Kalman filter i.e. a Linear-Quadratic Estimator (LQE) with a Linear-Quadratic Regulator (LQR). The separation principle guarantees that these can be designed and computed independently. LQG control applies to both linear time-invariant systems as well as linear time-varying systems. The application to linear time-invariant systems is well known. The application to linear time-varying systems enables the design of linear feedback controllers for non-linear uncertain systems.

# CHAPTER 4

# IMPLEMENTATION

The four steps involved in our approach to design and simulate the control system are elaborated in this chapter.

The implementation details of the system are explained in four steps as follows.

1. Experimental data gathering
2. System identification
3. Controller design
4. Simulating the control system in simulink

## 4.1 Experimental data gathering

Before gathering the experimental data some preliminary steps should be taken. They are

- Ensure that all sensors are working and transmitting the data to the onboard processor.
- Ensure that all servos are working and transmitting the data to the onboard processor.
- Ensure that wireless link has been established between the helicopter and Host computer.

Once all the checking process is done, the helicopter can be tested to acquire the time domain experimental data. Here the experimental data includes input and output data. Inputs are nothing but the pulse widths (in seconds) of the servos and outputs are roll and pitch angle (in degrees) of the helicopter. There are three servos which are responsible for roll and pitch angles; they are elevator (S1), aileron (S2), auxiliary servo (S3). The output data that is roll and pitch angle data is gathered from the inertial measurement unit (orientation sensor). Microstrain

3DM-GX1 orientation sensor [22] has been used to serve this purpose. The set up for the data gathering process is shown in the figure 3.

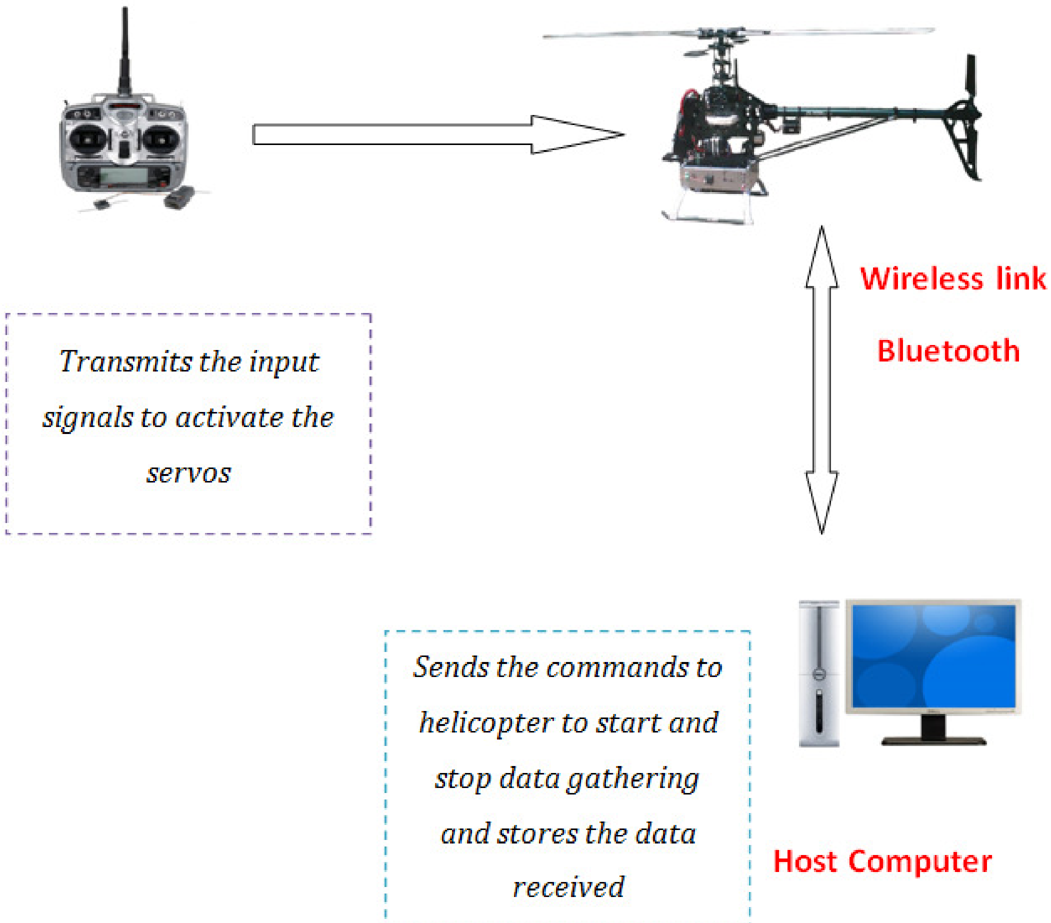


Figure 3: Set up for the experimental data gathering process

## 4.2 System Identification

The helicopter system model will be identified using the time domain experimental data with the help of MATLAB system identification tool box. The figure 4 depicts the process of system identification for the helicopter.

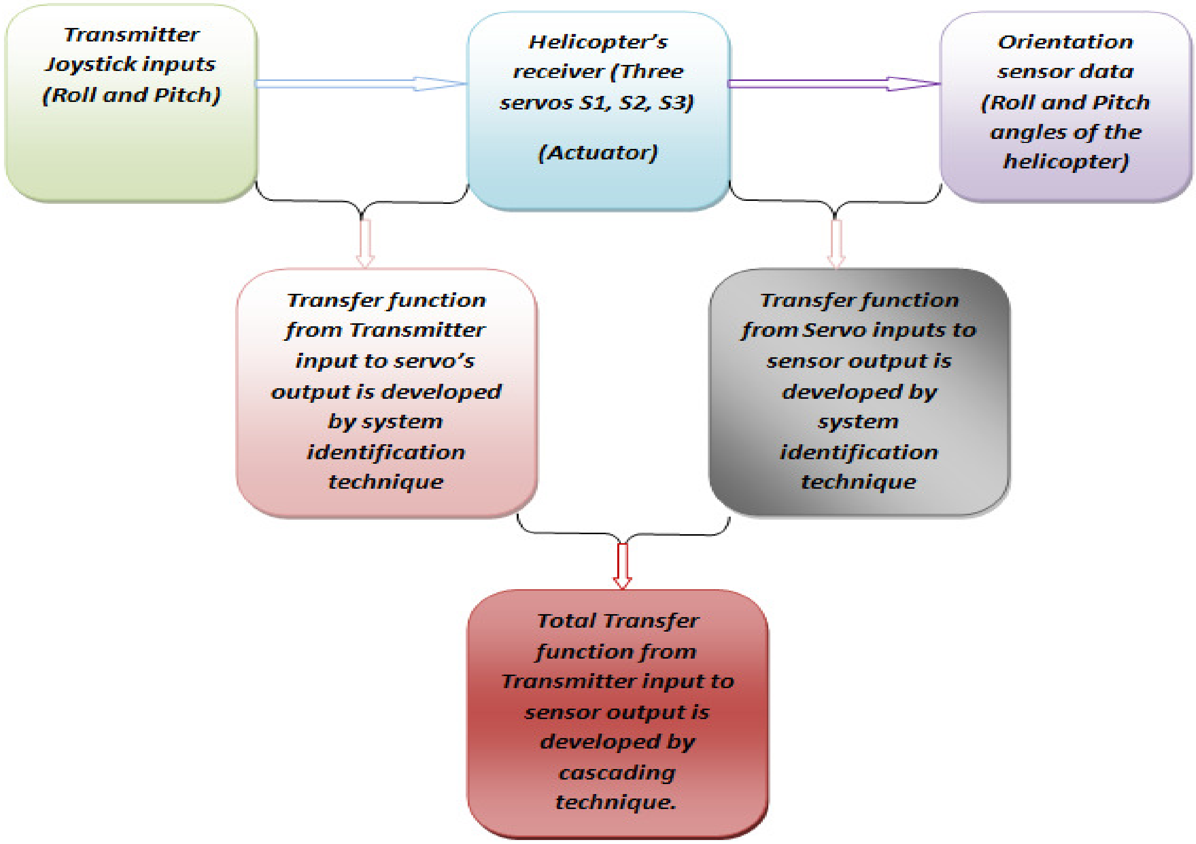


Figure 4: System identification process to develop helicopter system model

As shown in the figure 4 the system model is developed using system identification technique. First the transfer function is developed from transmitter input to the servo pulse width outputs. After that transfer function from servo pulse width inputs to the orientation sensor data (roll, pitch angle in degrees) is developed. The total system model is developed by cascading the two transfer function blocks.

**Identifying helicopter system model to control the roll angle**

In order to identify any system model, the information about the system and what the inputs and outputs of the system should be known. The total system to control the roll angle can be viewed as

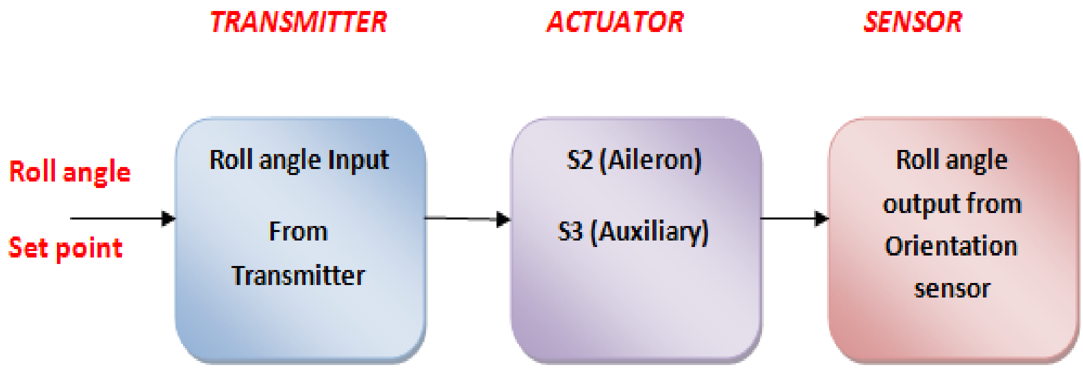


Figure 5: System identification process to develop helicopter system model to control the roll angle

From the figure 5 we can say that the total system consists of two sub systems whose model should be identified. The first subsystem is about finds transfer function between the Transmitter input and the actuator. The second subsystem is finds the transfer function between the actuator and sensor output. Once the two transfer functions in S-domain are obtained, the total system's transfer function is obtained by cascading the two subsystem's transfer functions.

**Identifying helicopter system model to control the pitch angle**

System identification procedure to control the pitch angle is same as we followed for the roll.

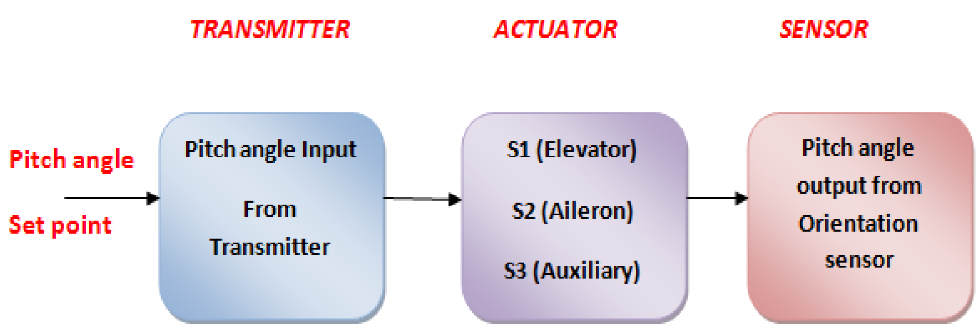


Figure 6: System identification process to develop helicopter system model to control the pitch angle

The only difference here is the inputs and outputs. The input is pitch angle data and the ervos that are responsible for the pitch angle are elevator (S1), aileron (S2), and auxiliary (S3).

The figure 7 shows the GUI (graphical user interface) of system identification tool box. This GUI enables to import the input, output data to identify the model. The type of the model selected here is state space model. This state space model will be converted to transfer function using Matlab code.

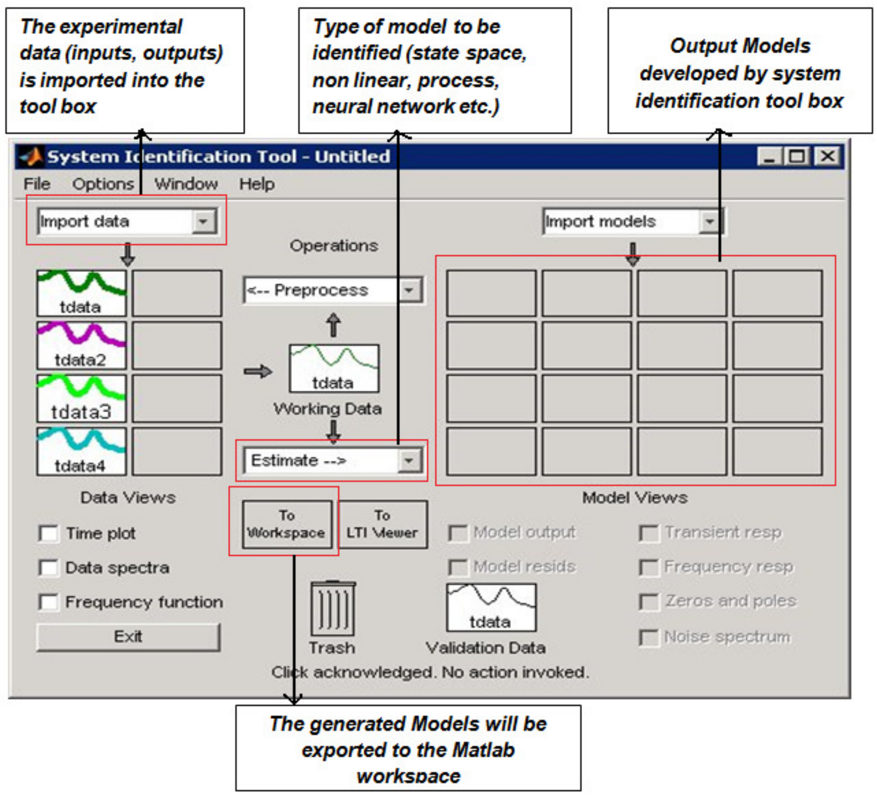


Figure 7: The GUI (graphical user interface) of system identification tool box

## 4.3 Controller design

Once the accurate system model is developed the controller will be designed using control system tool box of the MATALAB. This tool box provides various controlling techniques like

Proportional Integral Derivative (PID), Linear Quadratic Gaussian (LQG), Internal Model controller (IMC) to design the sophisticated controller to achieve high stability. Desired step response will be achieved by graphical tuning technique of control system tool box. The control system structure of the helicopter system is shown in figure 8.

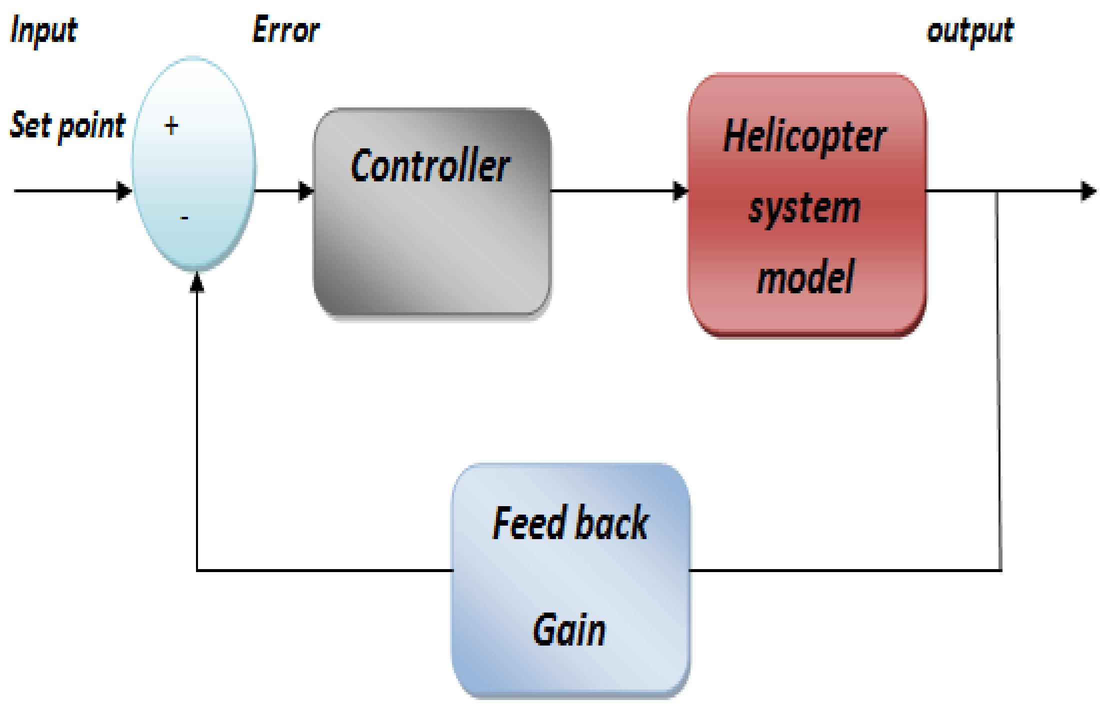


Figure 8: Control system structure of the helicopter

Figure 8 depicts the elements of the total control system. Here the helicopter system model is the transfer function developed by system identification process. Feedback gain element is the value of the gain added to the sensor output in order to eliminate the error. The controller element is the transfer function developed by tuning the system graphically. Root locus plot and bode plot are used to tune the system graphically until desired step response is achieved. Methods like pole placement and pole splitting have been used while tuning the system using root locus plot. This graphical tuning method is shown in the figure 9.

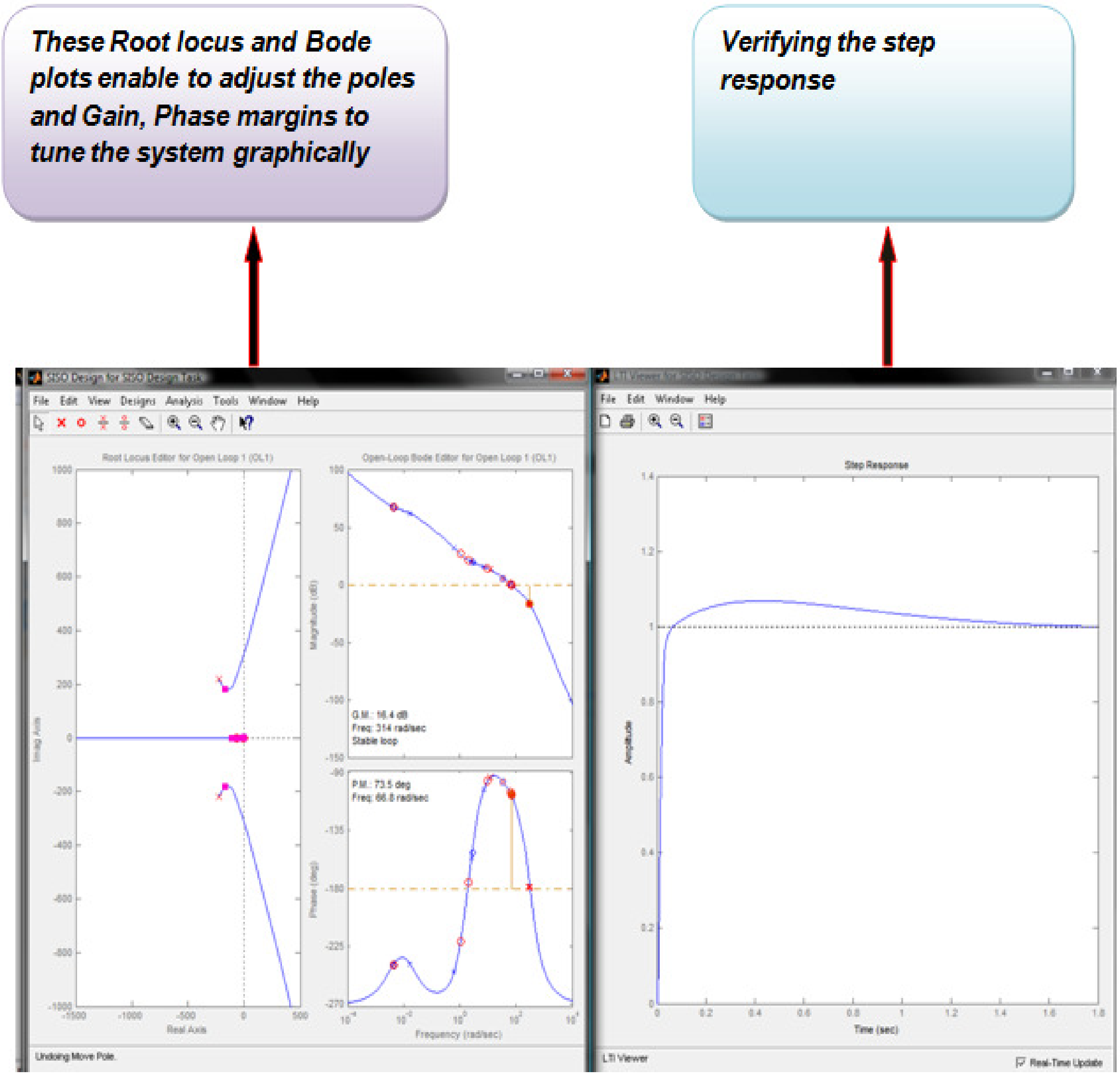


Figure 9: Graphical tuning method

## 4.4 Simulation

The transfer functions of helicopter system model and controller that are developed by above procedures will be used to simulate the control system in the simulink. The step input will be applied to verify the response of the system.

The figure 10 shows a sample simulated total control system in simulink. The “C” block represents the transfer function of the controller and “G” block represents the transfer function of the helicopter system model.

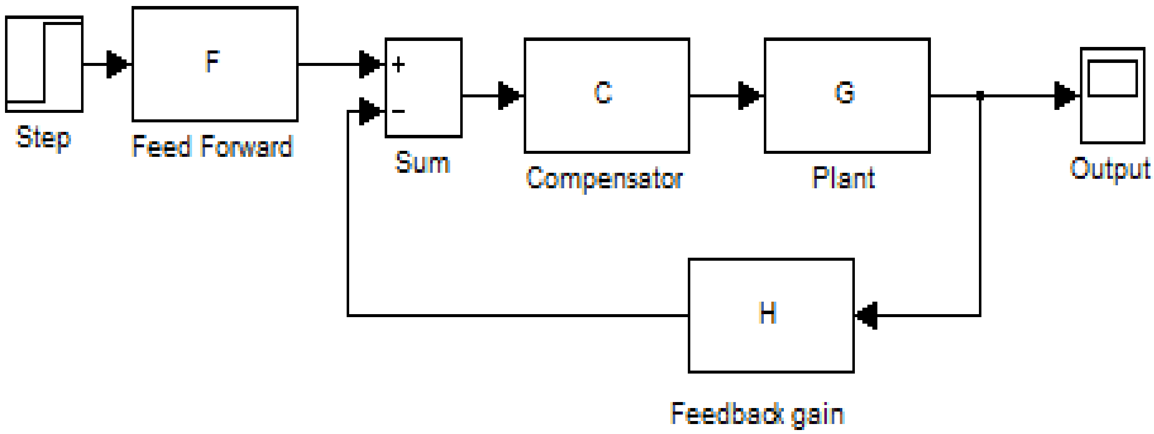


Figure 10: control system simulated in simulink.

The “H” and “F” blocks represent the feedback and feed forward gains respectively. The input is applied through input block and the output is verified in the output block.

# CHAPTER 5

# RESULTS

This chapter provides the results of this thesis.

## 5.1 Control system for roll angle

### System identification for servo S2 and roll angle

The servos responsible for the roll angle are aileron (S2) and auxiliary (S3).So the data values of these servo pulse widths in seconds are obtained for the respective roll angle in degrees. The figure 11 shows the servo pulse widths values of aileron (S2) in seconds and the respective roll angle in degrees.

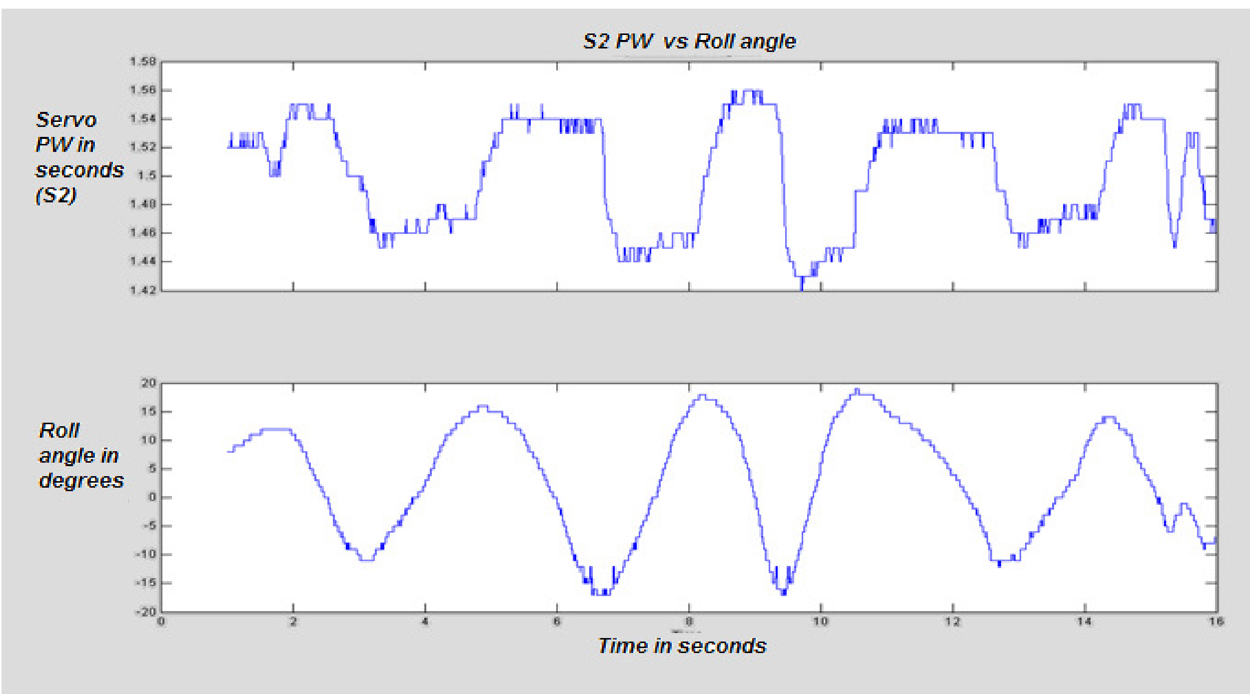


Figure 11: Servo Pulse width in seconds (S2 aileron) and corresponding roll angle in degrees

The system model is identified from the input roll angle to servo pulse width (S2) using system identification tool box. The figure 12 shows the original and predicted output values. The prediction rate is 87.9.

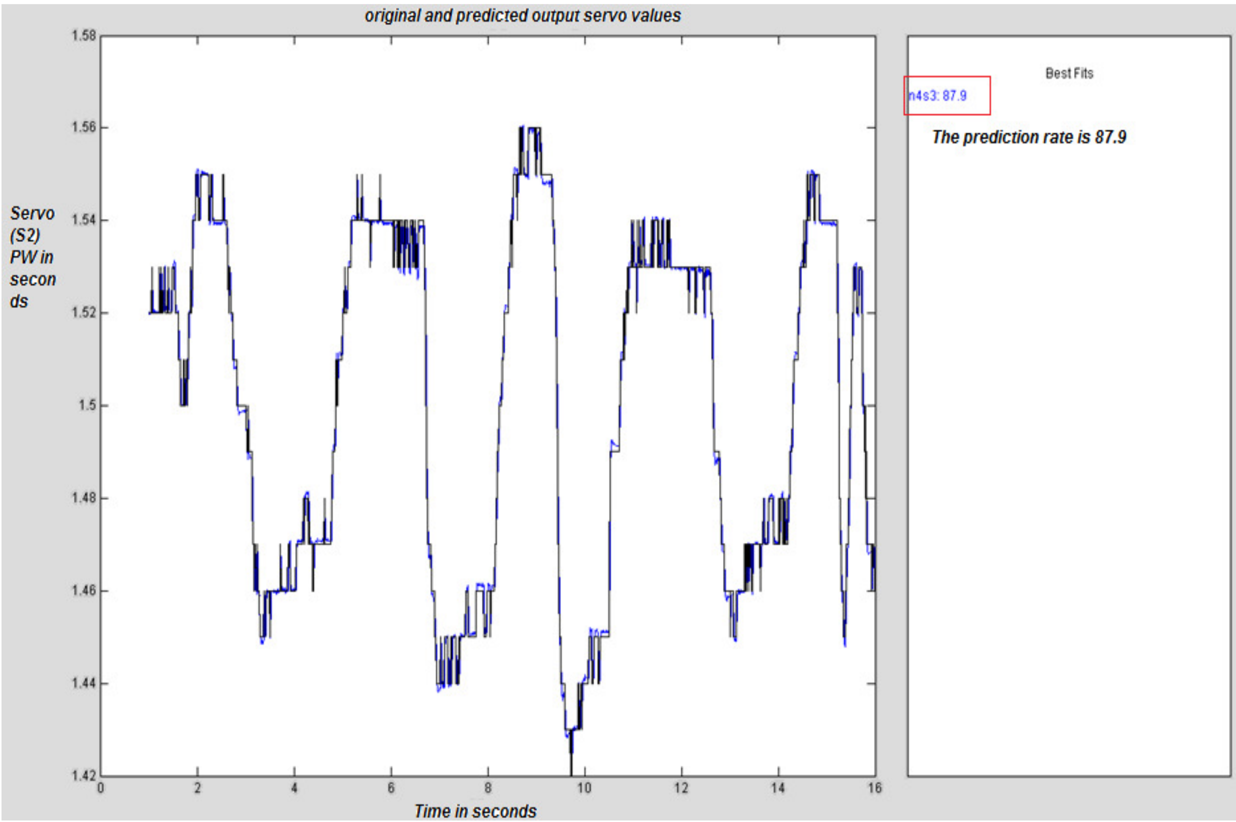


Figure 12: Real and predicted output servo pulse width data

*The model identified is in state space model*

$$x(t+Ts) = A\ x(t) + B\ u(t) + K\ e(t)$$

$$y(t) = C\ x(t) + D\ u(t) + e(t)$$

where

*A =*

| | *x1* | *x2* | *x3* |
|---|---|---|---|
| *x1* | *0.99982* | *0.0019985* | *0.001149* |
| *x2* | *0.0022047* | *0.5779* | *-0.79714* |

*x3    0.0022885     -0.49401     -0.18295*

*B =*

*roll angle i*

| | |
|---|---|
| *x1* | *8.1162e-006* |
| *x2* | *-0.0022473* |
| *x3* | *-0.0029943* |

*C =*

| | *x1* | *x2* | *x3* |
|---|---|---|---|
| *Servo2 Output* | *5544.5* | *-1.4663* | *-2.0372* |

*D =*

| | *roll angle i* |
|---|---|
| *Servo2 Output* | *0* |

*K =*

| | *Servo2 Output* |
|---|---|
| *x1* | *0.00014719* |
| *x2* | *0.0028111* |
| *x3* | *-0.0023308* |

*x(0) =*

| | |
|---|---|
| *x1* | *0.027405* |
| *x2* | *-0.024958* |
| *x3* | *-0.0070595* |

*Loss function 0.209884*

Similarly the model is identified from servo pulse width to sensor roll angle

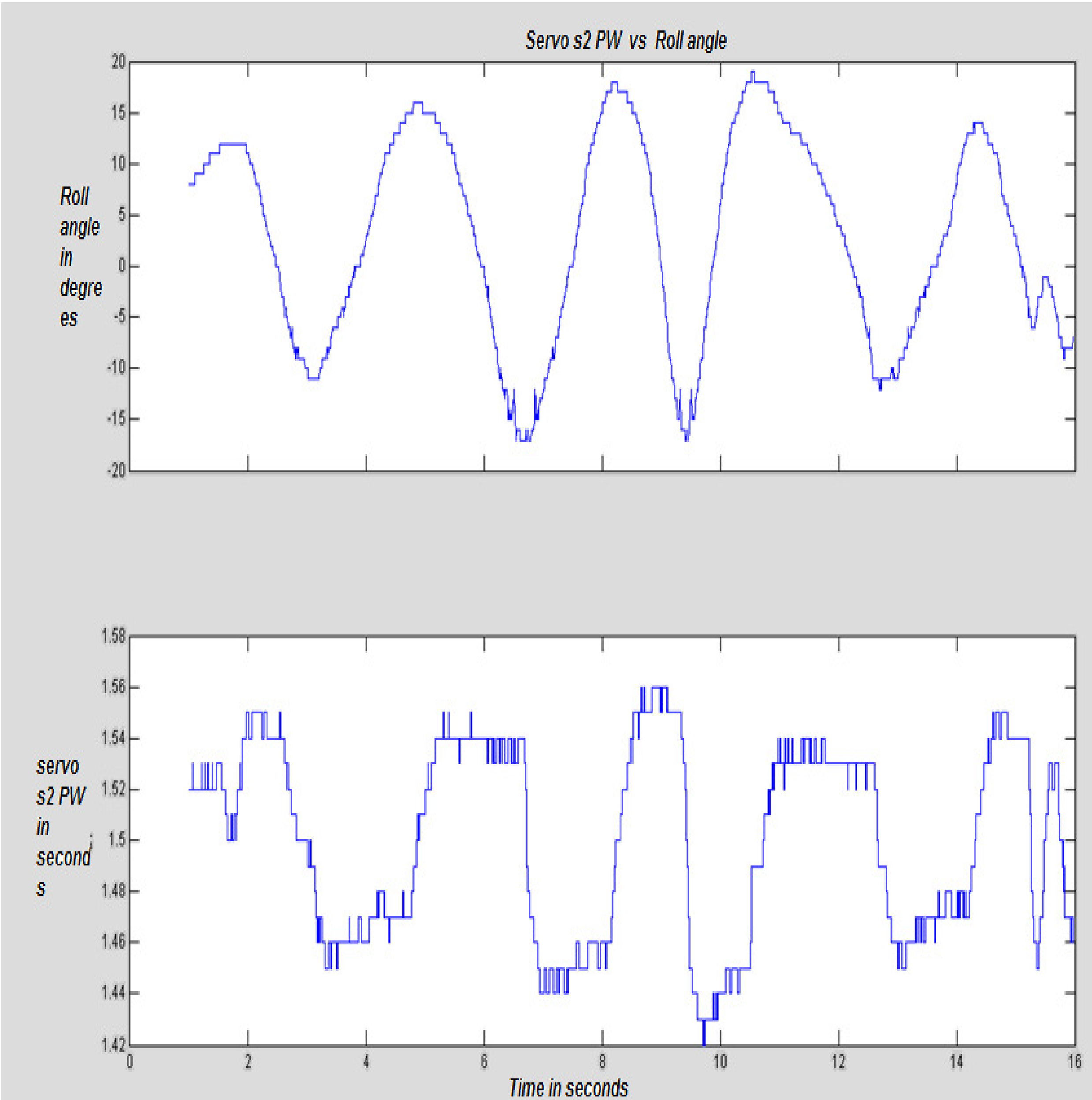


Figure 13: Roll angle in degrees and corresponding servo Pulse width in seconds (S2 aileron)

The system model is identified from the input servo pulse width (S2) to output sensor roll angle using system identification tool box. The figure 14 shows the real and predicted output values. The prediction rate is 94.95.

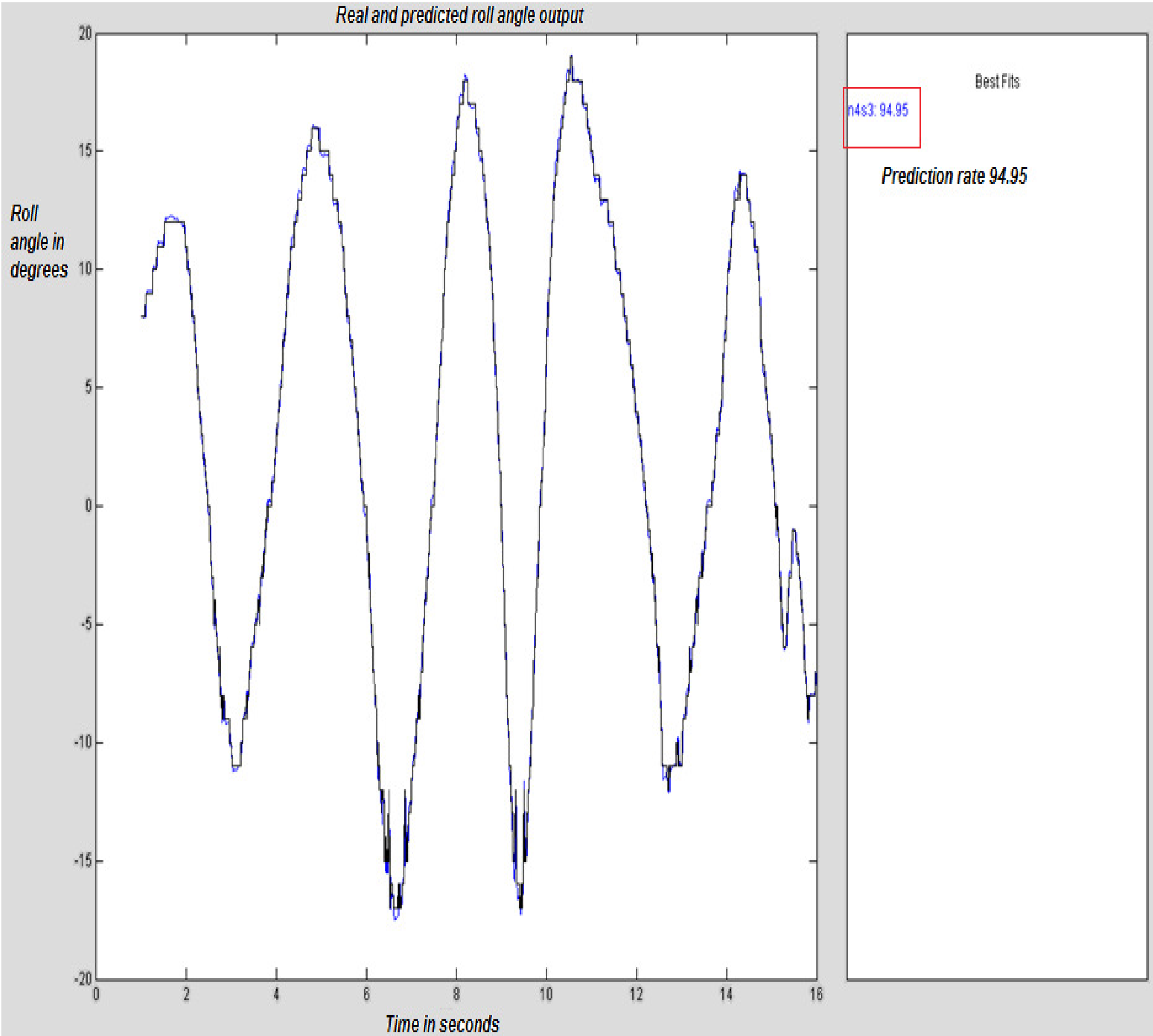


**Figure 14: Real and predicted output values of roll angle**

*The model identified is in state space model form*

$$x(t+Ts) = A\,x(t) + B\,u(t) + K\,e(t)$$

$$y(t) = C\,x(t) + D\,u(t) + e(t)$$

*Where*

*A =*

| | x1 | x2 | x3 |
|---|---|---|---|
| x1 | 0.9981 | -0.028818 | 0.026893 |

| | | | |
|---|---|---|---|
| *x2* | *-0.0036891* | *1.0251* | *0.48799* |
| *x3* | *-0.0016887* | *-0.0080357* | *-0.49671* |

*B =*

| | *servo2 PW* |
|---|---|
| *x1* | *-0.00019898* |
| *x2* | *0.00011138* |
| *x3* | *0.0001283* |

*C =*

| | *x1* | *x2* | *x3* |
|---|---|---|---|
| *roll angle* | *361.3* | *3.6028* | *-1.6449* |

*D =*

| | *servo2 PW* |
|---|---|
| *roll angle* | *0* |

*K =*

| | *roll angle* |
|---|---|
| *x1* | *0.002476* |
| *x2* | *-0.0042106* |
| *x3* | *-0.0037935* |

*x(0) =*

| | |
|---|---|
| *x1* | *0.032543* |
| *x2* | *-1.036* |
| *x3* | *0.017891* |

Loss function 0.256869

These state space models are converted into transfer functions and the total system transfer function is derived.

***-1.137e-013 s^5 - 22.76 s^4 - 2453 s^3 - 6.256e004 s^2 - 1.568e005 s - 667.4***

***-----------------------------------------------------------------------------***

***s^6 + 137.9 s^5 + 5055 s^4 + 2.034e004 s^3 - 7.154e004 s^2 - 5.273e004 s - 860.6***

**System identification for servo S3 and roll angle**

Similarly the system identification process is carried out for the roll angle output due to servo (S3).Transfer function from input roll angle to output servo (S3) pulse width:

***-0.04854 s^2 + 12.3 s + 30.84***

***--------------------------------------------***

***s^3 + 137.2 s^2 + 3999 s - 38.65***

Transfer function from input servo pulse width (S3) to output sensor roll angle:

***21.52 s^2 - 9.681e004 s - 862.7***

***------------------------------------------***

***s^3 + 138 s^2 - 70.2 s + 445.8***

Total system Transfer function:

***-2.274e-013 s^5 - 1.044 s^4 + 4964 s^3 - 1.19e006 s^2 - 2.996e006 s - 2.661e004***

***----------------------------------------------------------------------------------------------------------------***

***s^6 + 275.2 s^5 + 2.286e004 s^4 + 5.426e005 s^3 - 2.249e005 s^2 + 1.785e006 s - 1.723e004***

**Designing the controller for servo S2**

The controller transfer function will be derived using Matlab control system tool box and graphical tuning method. Three controlling methods PID, IMC and LQG are used to design the controller. The method which ensures the good step response is selected to design the controller.

PID (Proportional, Integral and Derivative) method

Root locus and bode plots

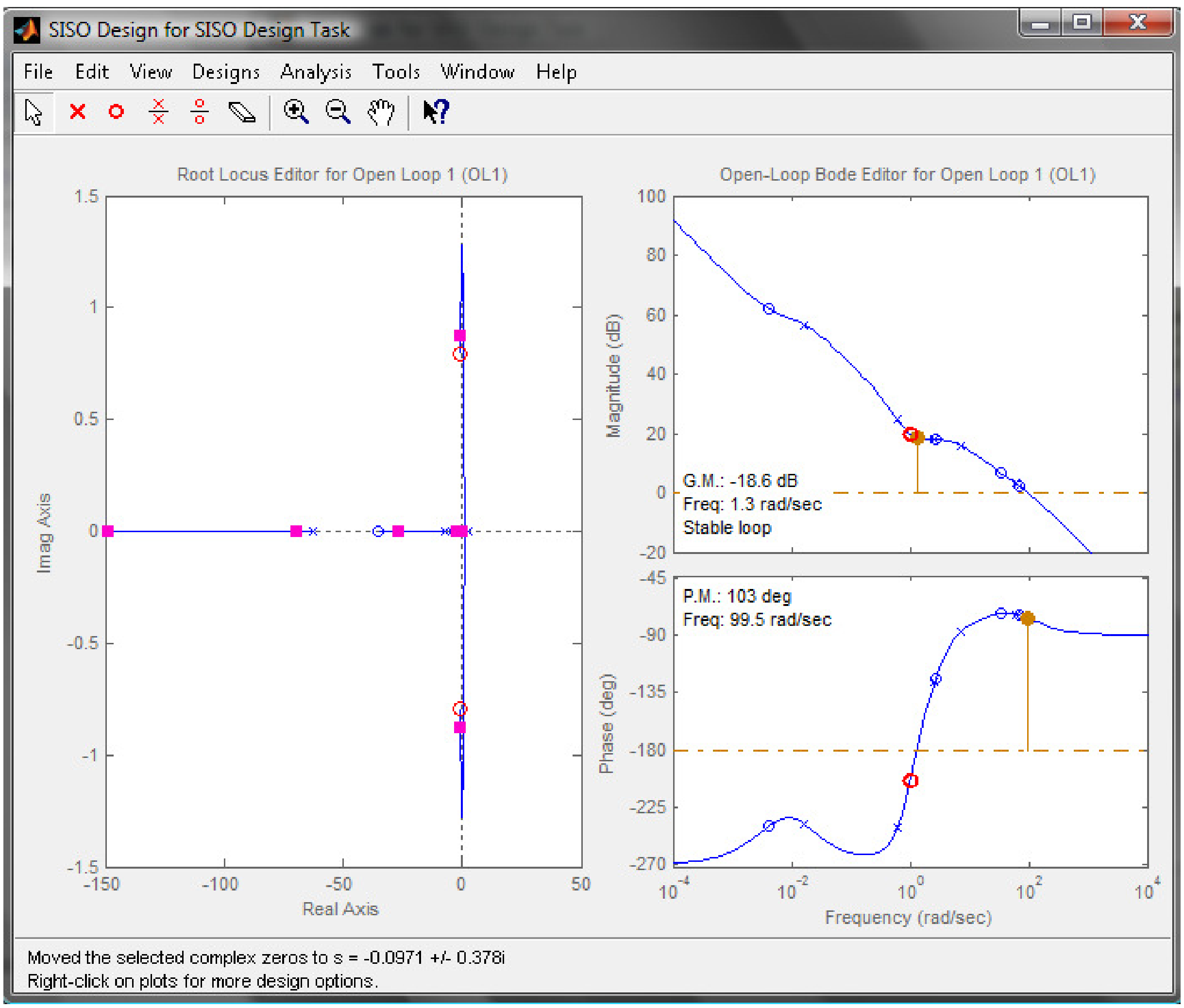


Figure 15: Root locus and bode plots while tuning the system to design the controller for S2 servo using PID method for roll angle control

Step response

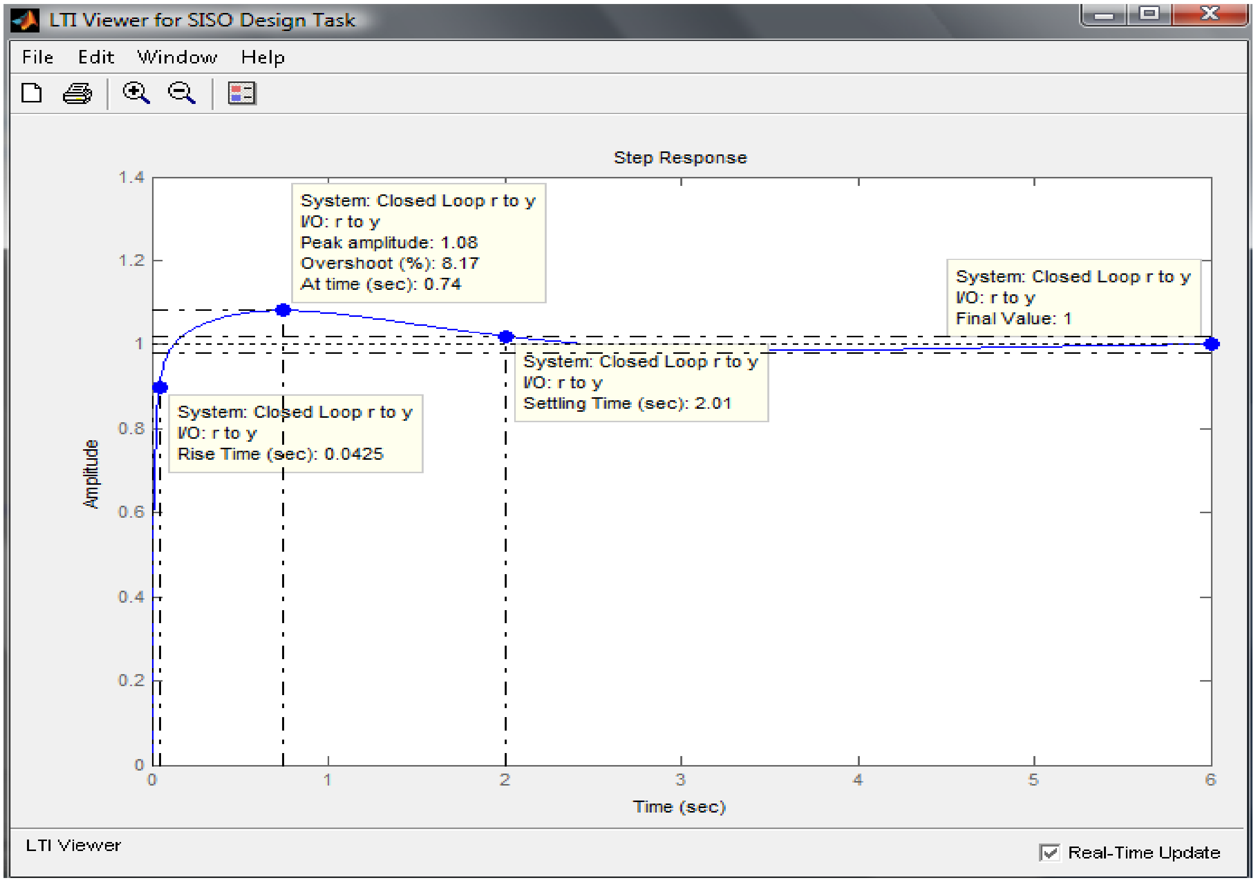


Figure 16: Step response obtained while tuning the system to design the controller for S2 servo using PID method for roll angle control

| | |
|---|---|
| **Rise time** | 0.0425 sec |
| **overshoot** | 8.17 % |
| **Peak amplitude** | 1.08 |
| **Settling time** | 2.01 sec |
| **Final value** | 1 |

Table 1: step response values obtained while tuning the system to design the controller for S2 servo using PID method for roll angle control

IMC (Internal Model Controller) method

Root locus and bode plots

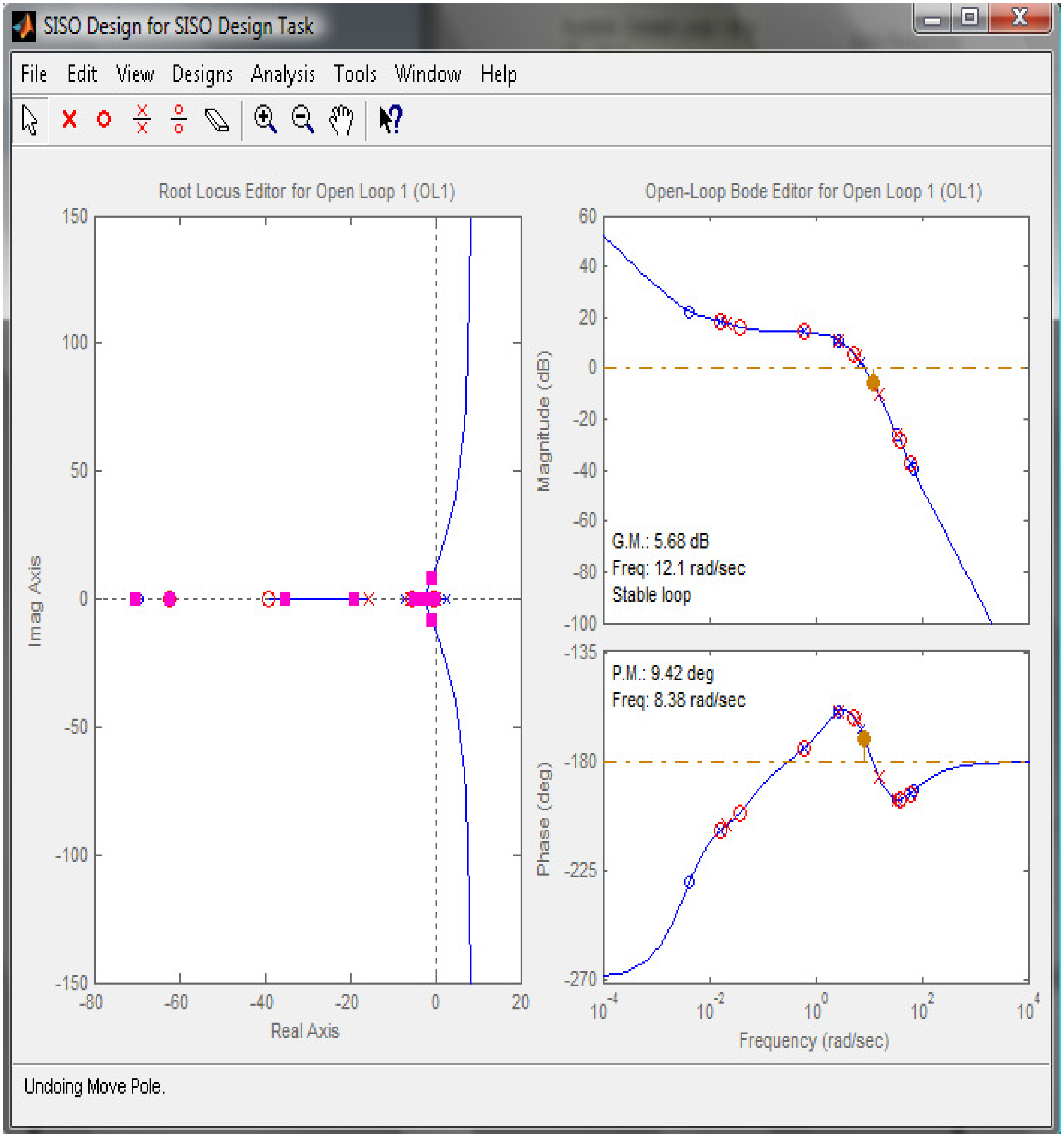


Figure 17: Root locus and bode plots while tuning the system to design the controller for S2 servo using IMC method for roll angle control

Step response

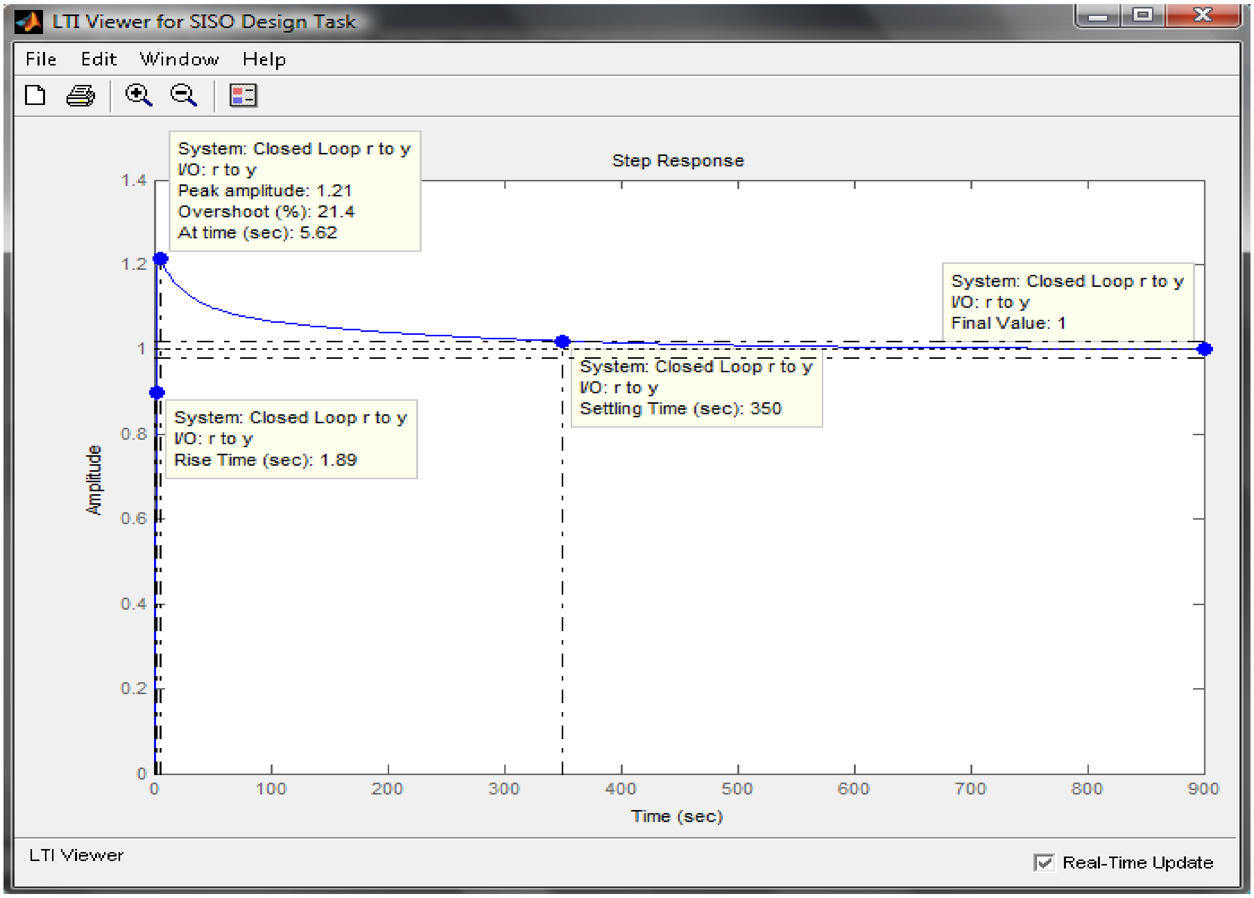


Figure 18: Step response obtained while tuning the system to design the controller for S2 servo using IMC method for roll angle control

| | |
|---|---|
| **Rise time** | 1.89 sec |
| **overshoot** | 21.4 % |
| **Peak amplitude** | 1.21 |
| **Settling time** | 350sec |
| **Final value** | 1 |

Table 2: Step response values obtained while tuning the system to design the controller for S2 servo using IMC method for roll angle control

LQG (Linear Quadratic Gaussian) method

Root locus and bode plots

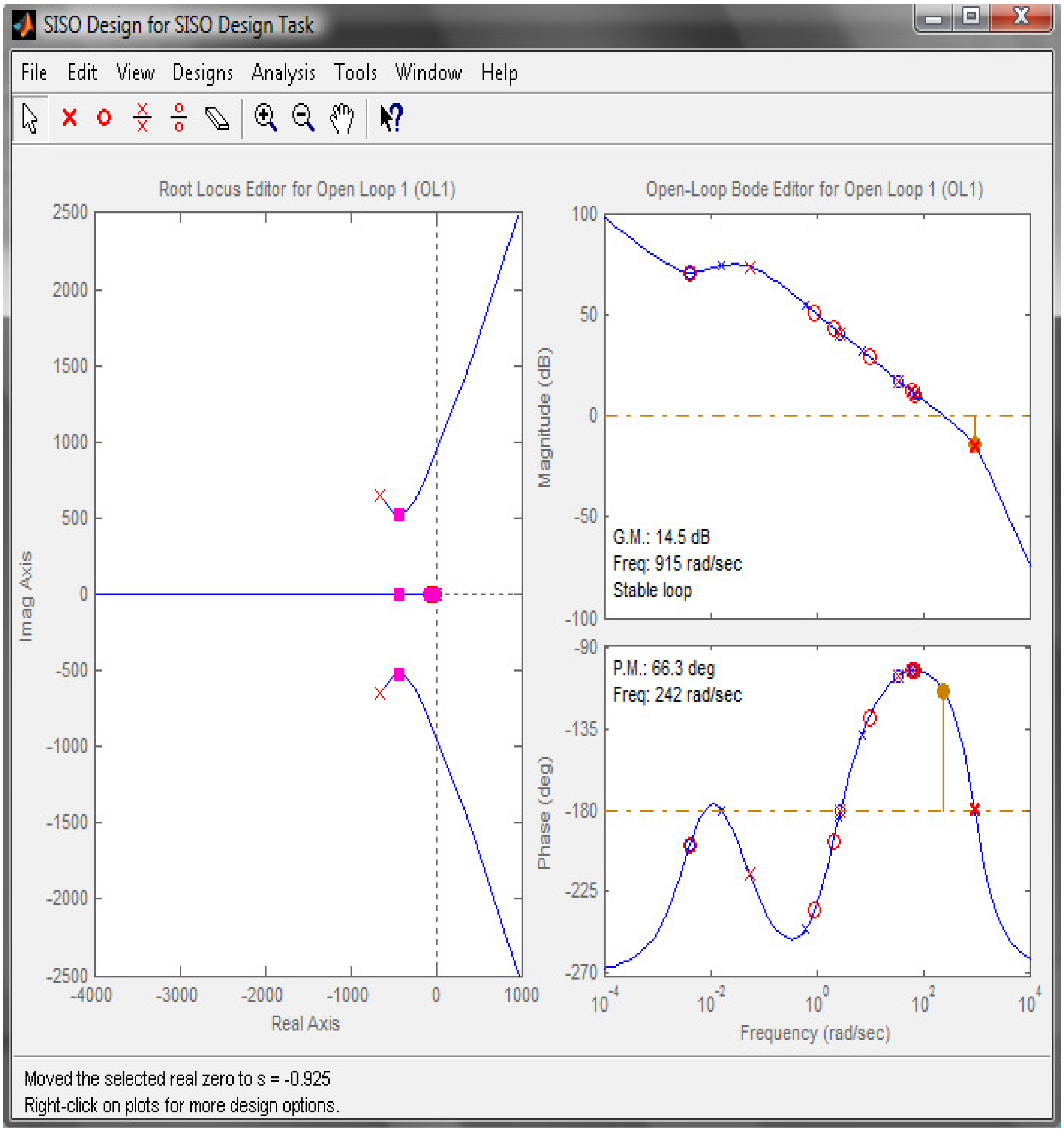


Figure 19: Root locus and bode plots while tuning the system to design the controller for S2 servo using LQG method for roll angle control

Step Response

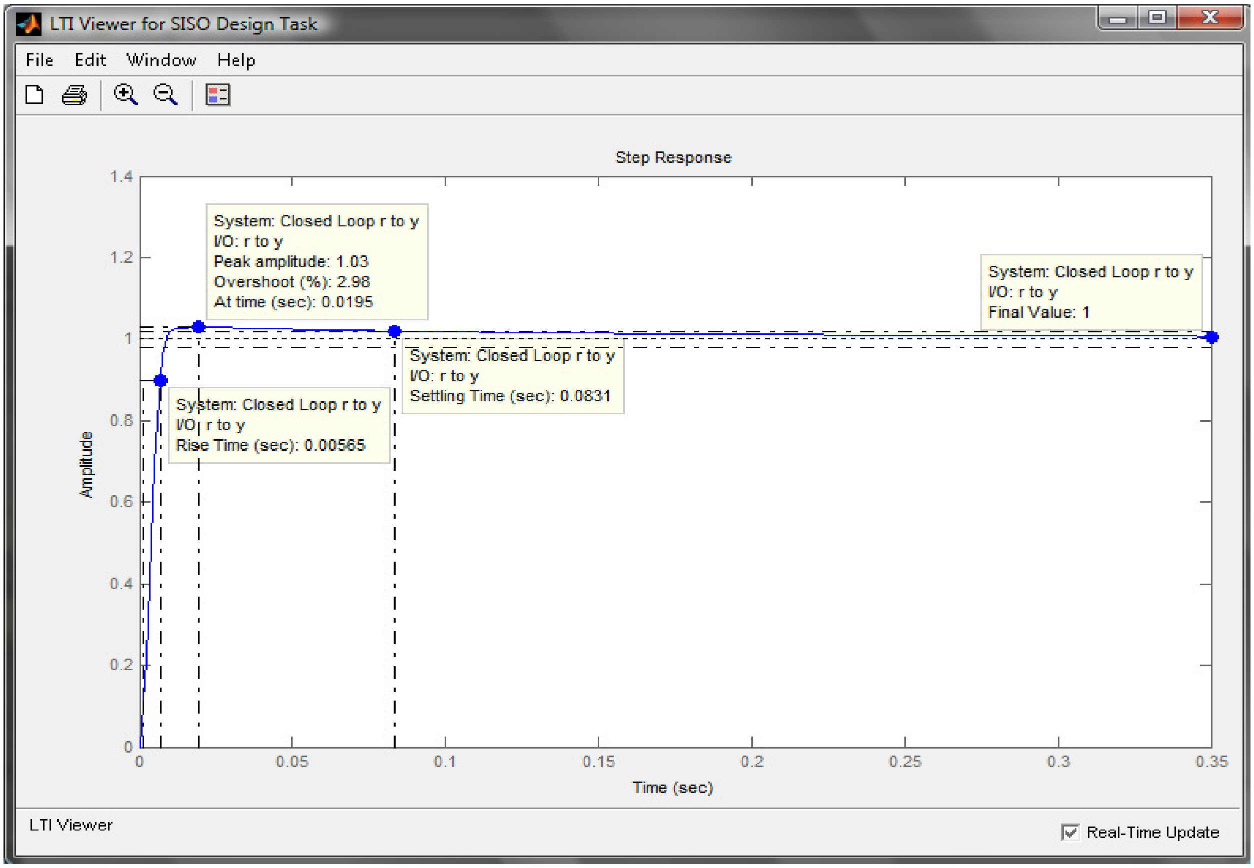


Figure 20: Step response obtained while tuning the system to design the controller for S2 servo using LQG method for roll angle control

| | |
|---|---|
| **Rise time** | 0.00565 sec |
| **overshoot** | 2.98 % |
| **Peak amplitude** | 1.03 |
| **Settling time** | 0.0831 sec |
| **Final value** | 1 |

Table 3: Step response values obtained while tuning the system to design the controller for S2 servo using LQG method for roll angle control

| ***Method*** | ***Rise time in seconds*** | ***Overshoot (%)*** | ***Peak amplitude*** | ***Settling time in seconds*** | ***Final value*** |
|---|---|---|---|---|---|
| **PID** | 0.0425 | 8.17 | 1.08 | 2.01 | 1 |
| **IMC** | 1.89 | 21.4 | 1.21 | 350 | 1 |
| **LQG** | 0.00565 | 2.98 | 1.03 | 0.0831 | 1 |

Table 4: Comparing the step response values of all three methods to select the controller for S2 servo for roll angle control

The table 4 clearly indicates that LQG method has the least settling time, overshoot and rise time. So the transfer function derived my LQG method is selected as the transfer function for the controller. The derived transfer function is

***-1.137e-013 s^5 - 22.76 s^4 - 2453 s^3 - 6.256e004 s^2 - 1.568e005 s - 667.4***

**---------------------------------------------------------------------------------------------------------**

***s^6 + 137.9 s^5 + 5055 s^4 + 2.034e004 s^3 - 7.154e004 s^2 - 5.273e004 s - 860.6***

### Designing the controller for servo S3

Similarly the controller transfer function for servo S3 input will be derived

LQG (Linear Quadratic Gaussian) method

Root locus and bode plots

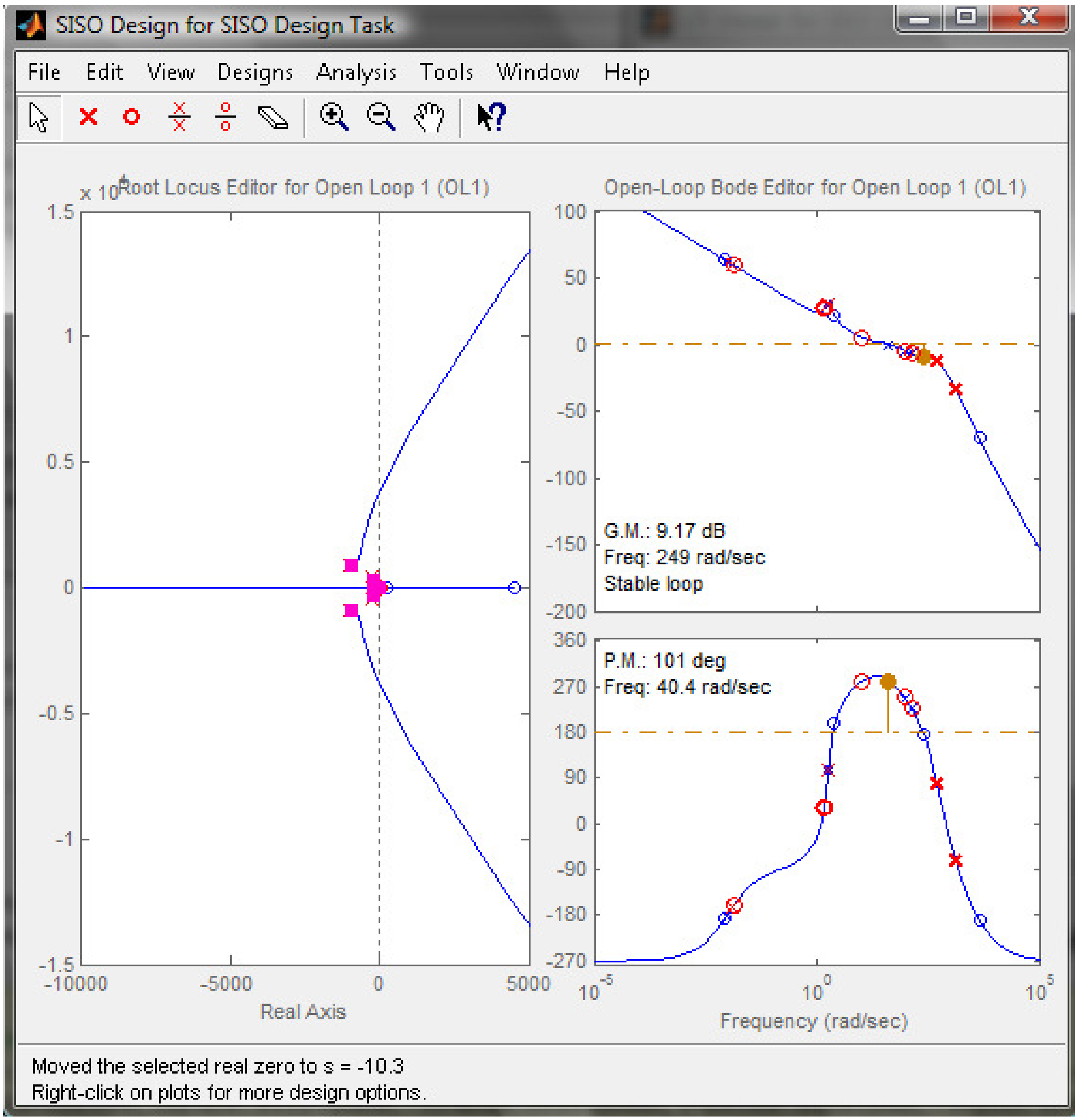


Figure 21: Root locus and bode plots while tuning the system to design the controller for S3 servo using LQG method for roll angle control

Step response

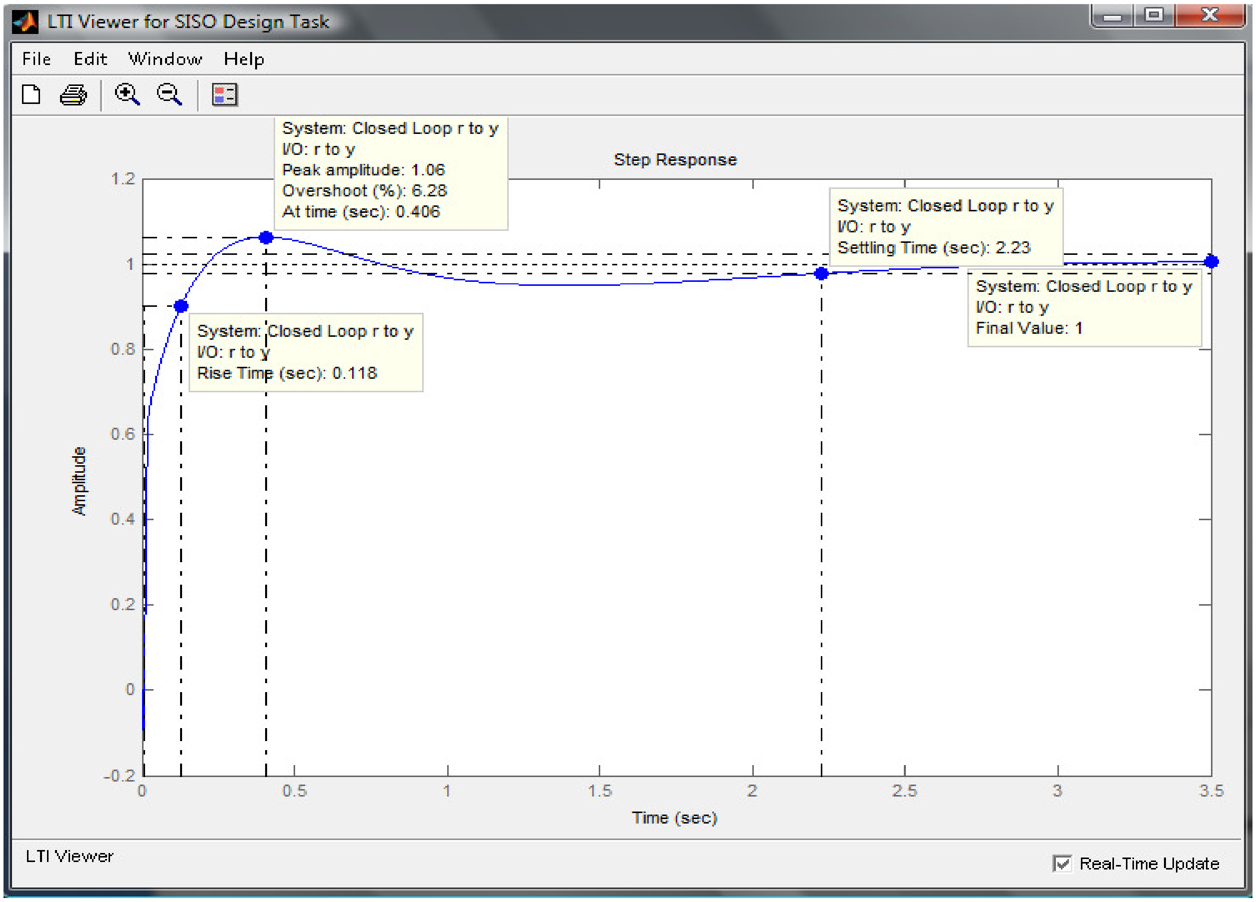


Figure 22: Step response obtained while tuning the system to design the controller for S3 servo using LQG method for roll angle control

| | |
|---|---|
| **Rise time** | 0.118 sec |
| **overshoot** | 6.28% |
| **Peak amplitude** | 1.06 |
| **Settling time** | 2.23 sec |
| **Final value** | 1 |

Table 5: Step response values obtained while tuning the system to design the controller for S3 servo using LQG method for roll angle control

The transfer function obtained is

$$\frac{-2.274e{-}013\ s^5 - 1.044\ s^4 + 4964\ s^3 - 1.19e006\ s^2 - 2.996e006\ s - 2.661e004}{s^6 + 275.2\ s^5 + 2.286e004\ s^4 + 5.426e005\ s^3 - 2.249e005\ s^2 + 1.785e006\ s - 1.723e004}$$

**Simulation**

Using the transfer functions derived for the system model and controller, total system is simulated in simulink. The figure 23 shows the simulated model. The input is unit step signal. The helicopter image is the subsystem. This subsystem is shown in the figure 25. The output is verified in the oscilloscope.

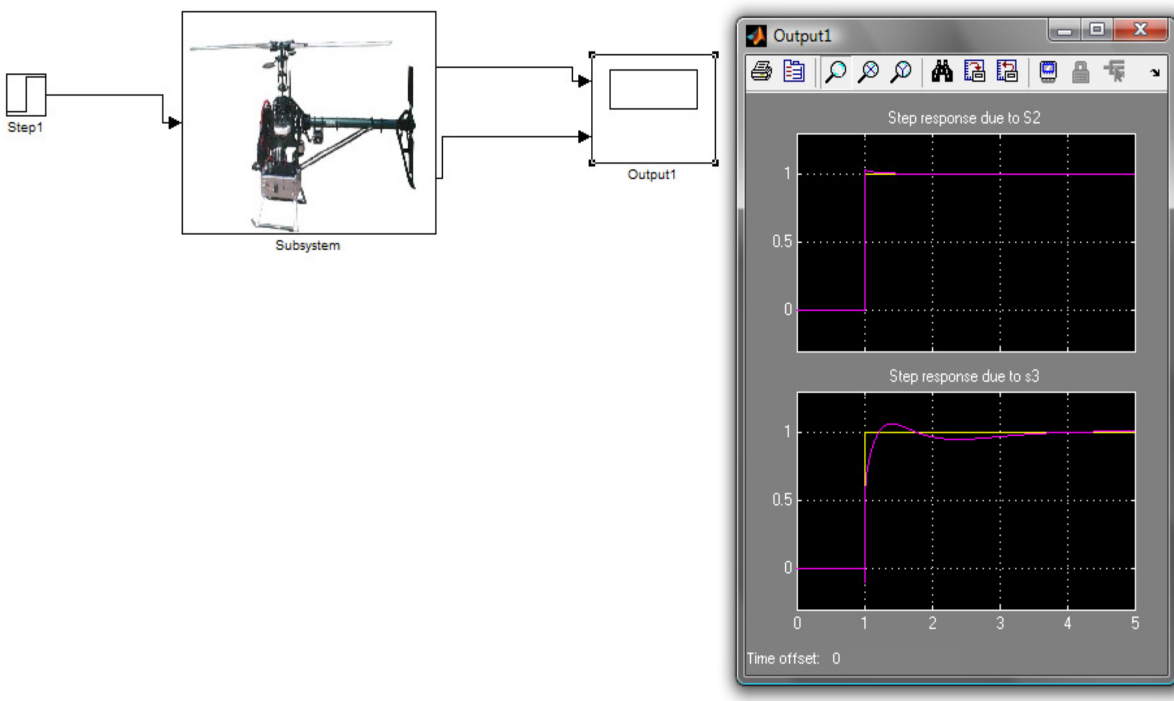


Figure 23: Simulated control system for the roll angle

Scope output

The yellow color graph represents the input unit step signal. The pink color graph represents the helicopter system output for two servos (S2 and S3). The values on X axis represent the time in seconds and the values on the Y axis represent the amplitude.

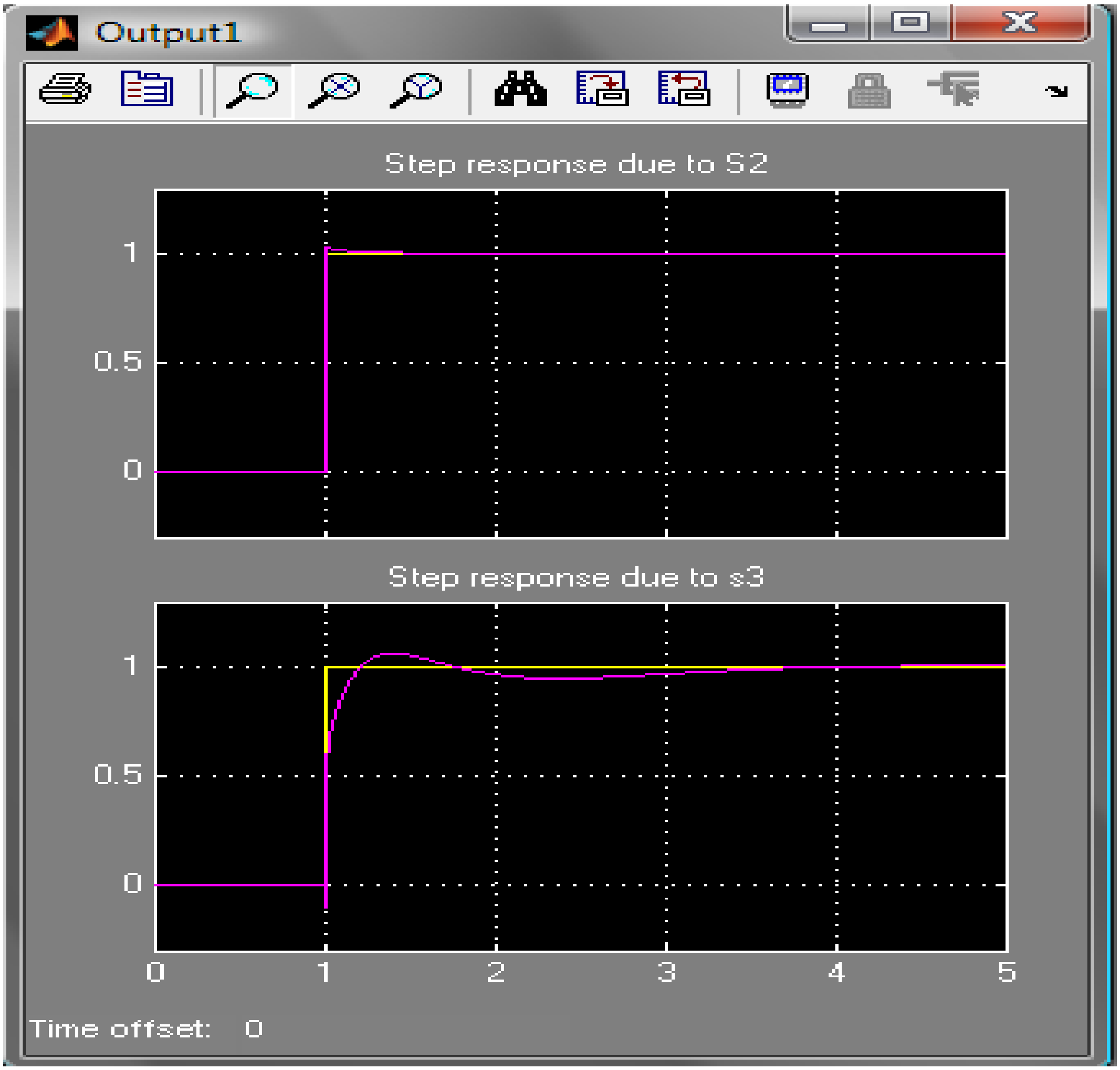


Figure 24: The scope output for the simulated roll angle control system

Subsystem

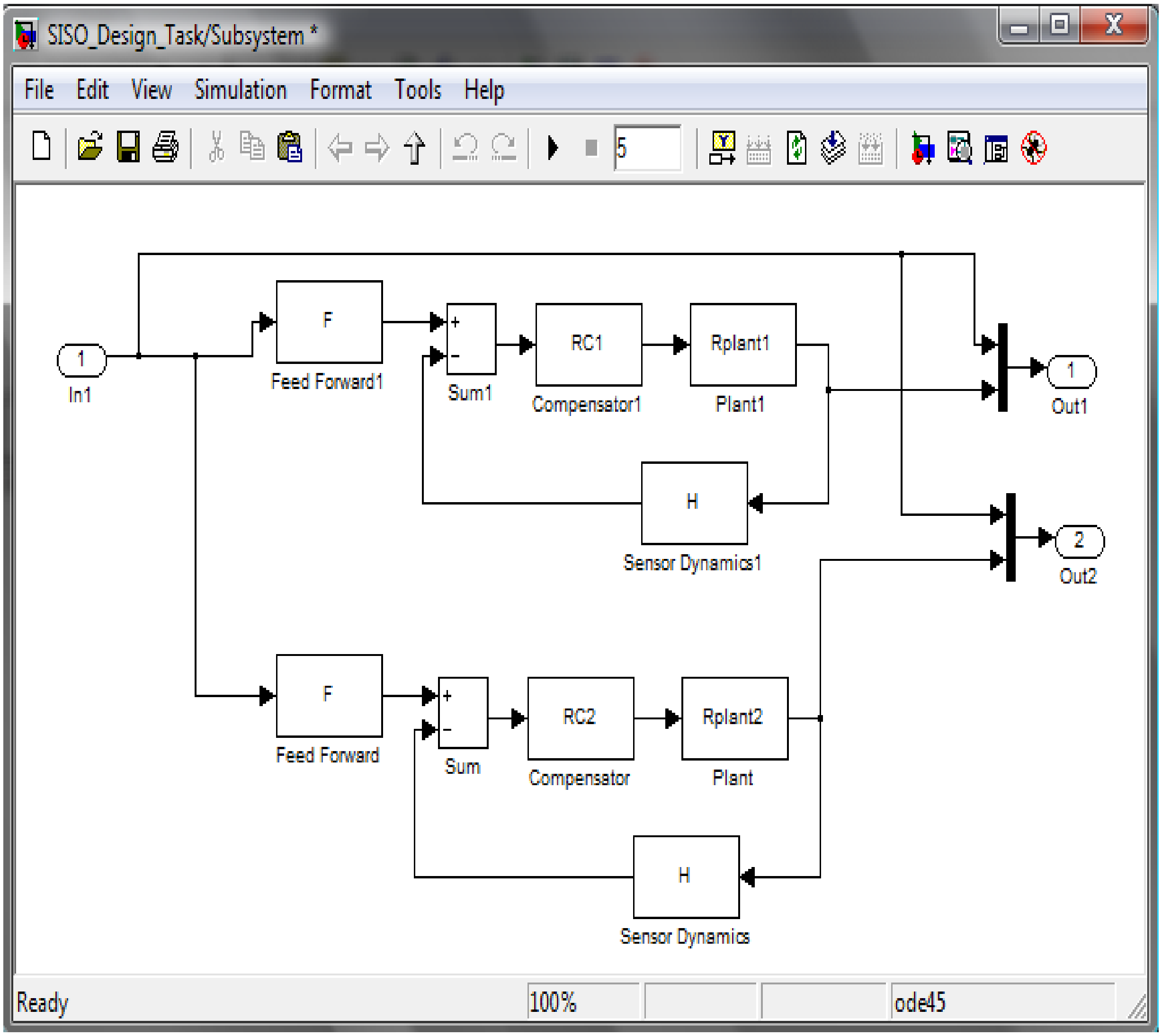


Figure 25: The subsystem for the roll angle control system

## 5.2 Control system for pitch angle

The servos responsible for the pitch angle are elevator (S1), aileron (S2), and auxiliary (S3). The control system transfer functions are derived for these three servo inputs in the similar fashion we designed the control system for the roll angle.

**Designing the controller for servo S1**

Root locus and Bode plots

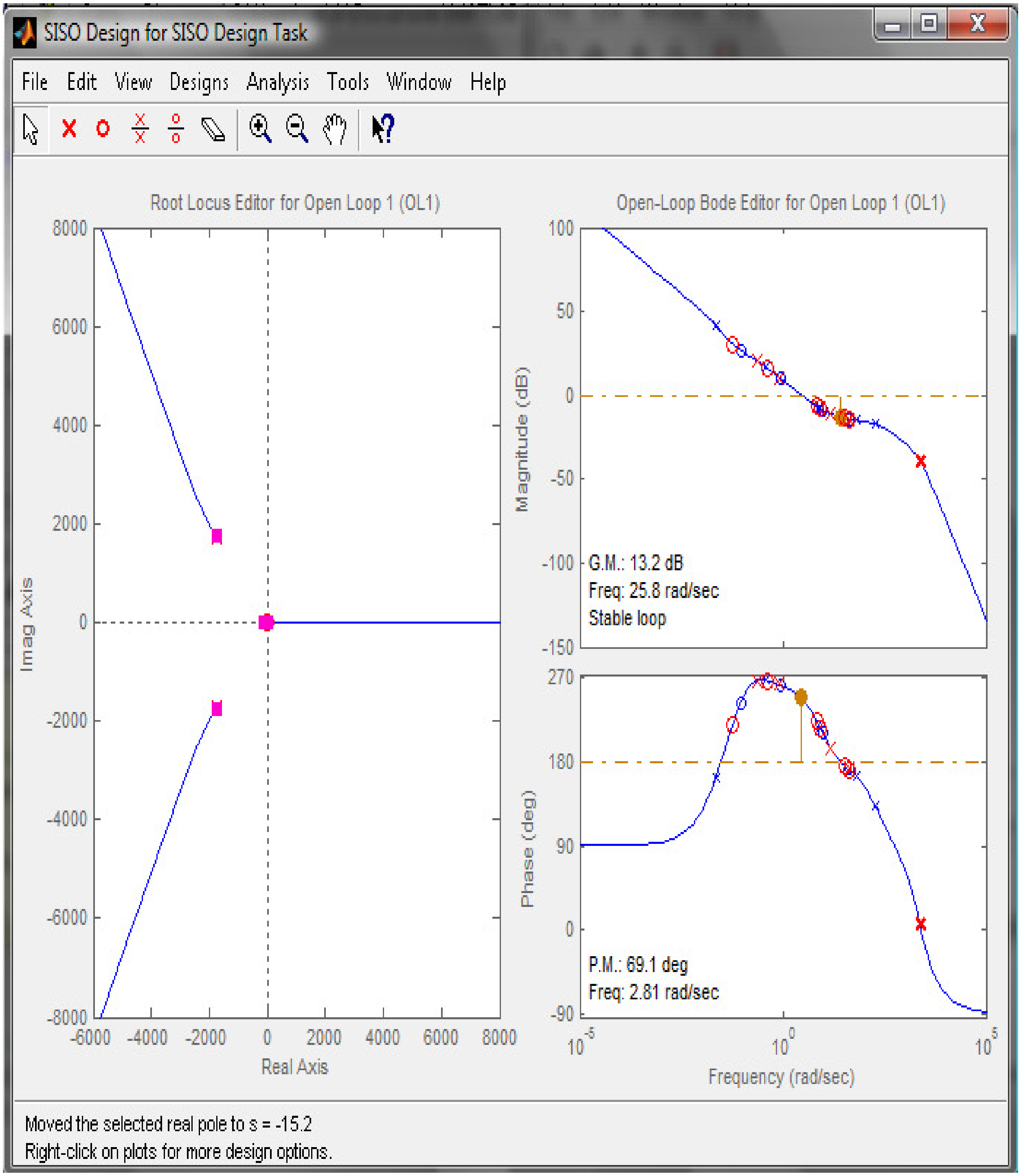


Figure 26: Root locus and bode plots while tuning the system to design the controller for S1 servo using LQG method for pitch angle control

Step response

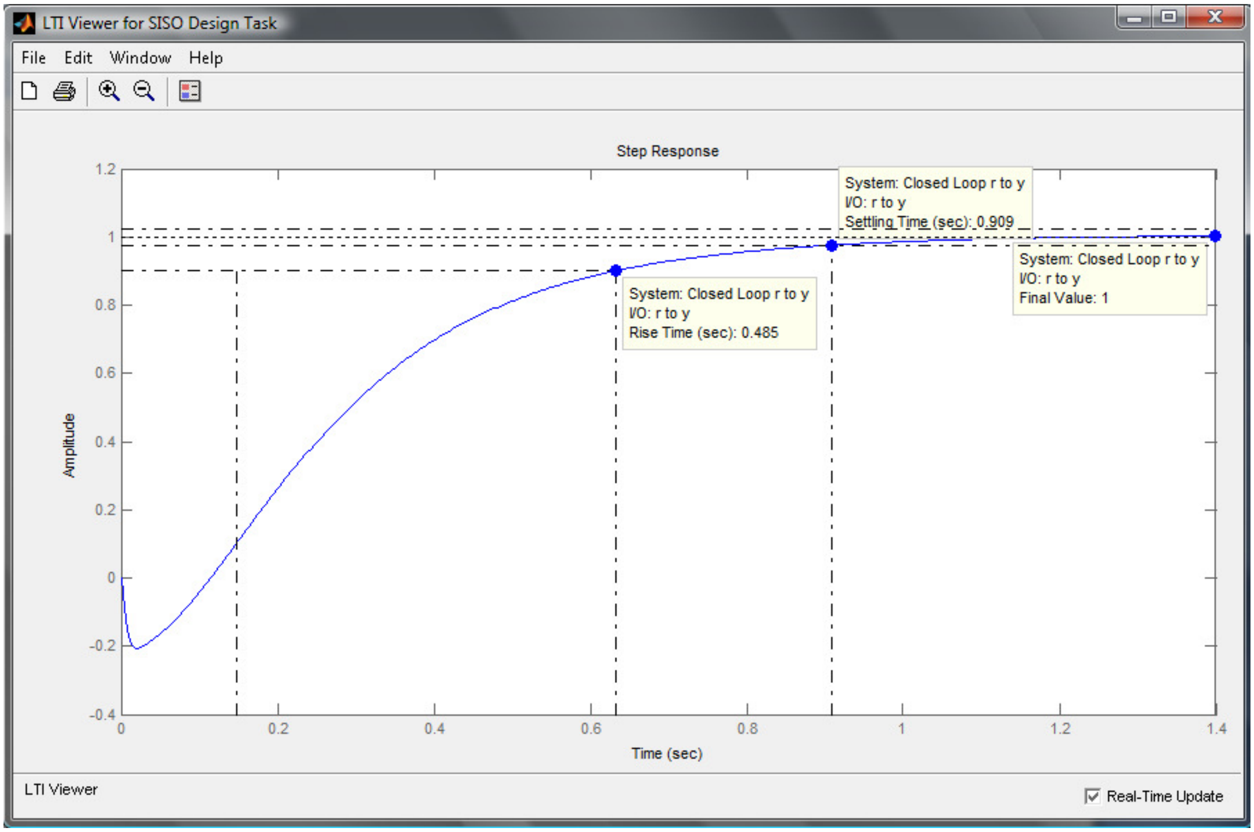


Figure 27: Step response obtained while tuning the system to design the controller for S1 servo using LQG method for pitch angle control

| | |
|---|---|
| **Rise time** | 0.485 sec |
| **Undershoot** | 2 % |
| **Negative Peak amplitude** | -0.2 |
| **Settling time** | 0.909 sec |
| **Final value** | 1 |

Table 6: Step response values obtained while tuning the system to design the controller for S1 servo using LQG method for pitch angle control

The transfer function obtained is

***-2.451e006 s^6 - 2.287e008 s^5 - 6.786e009 s^4 - 7.034e010 s^3 - 2.476e011 s^2 - 1.067e011 s - 5.507e009***

**---------------------------------------------------------------------------------------------------------------**

***s^7 + 3538 s^6 + 6.239e006 s^5 + 3.523e008 s^4 + 4.191e009 s^3 + 4.106e009 s^2 + 7.332e008 s***

**Designing the controller for servo S2**

Root locus and Bode plots

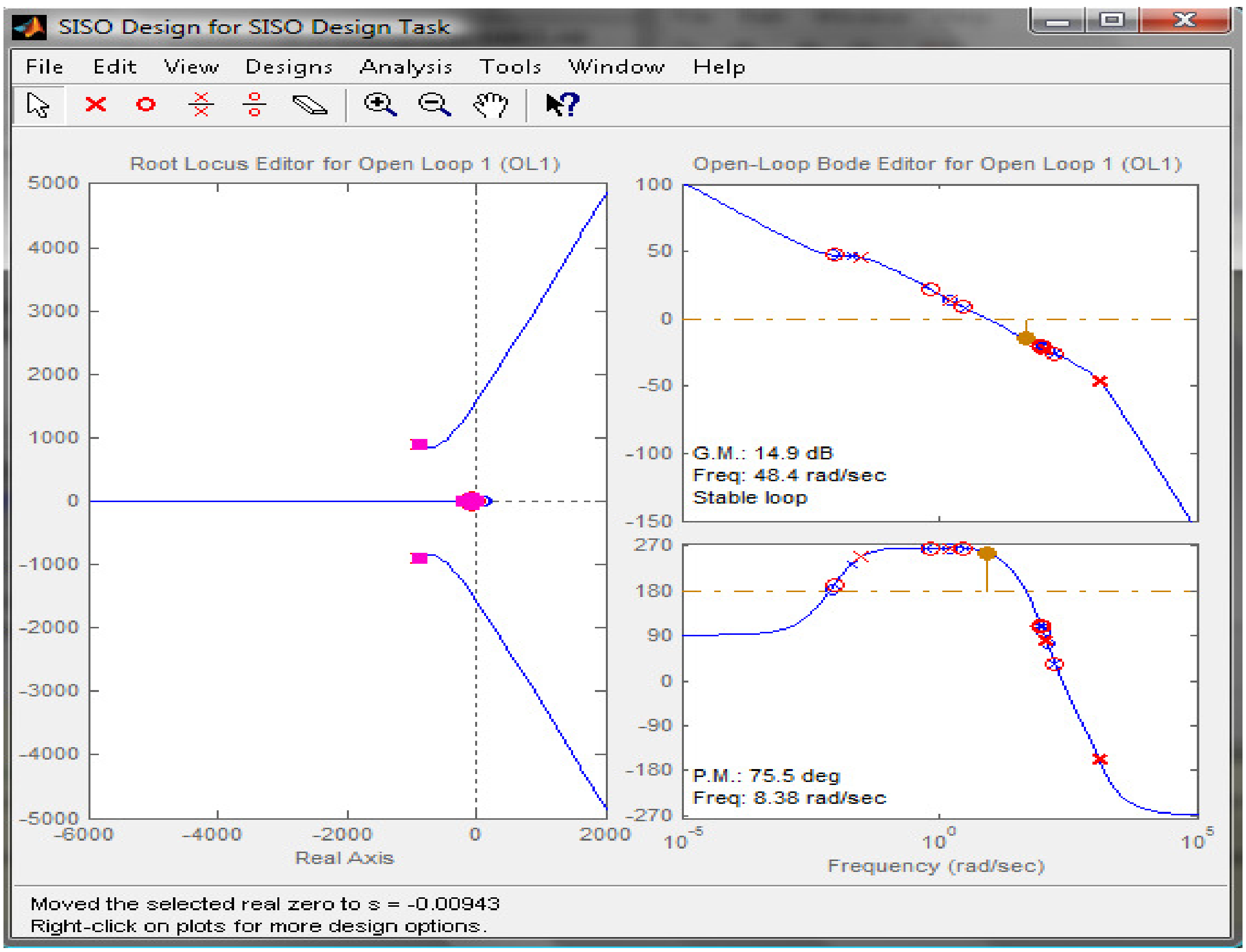


Figure 28: Root locus and bode plots while tuning the system to design the controller for S2 servo using LQG method for pitch angle control

Step response

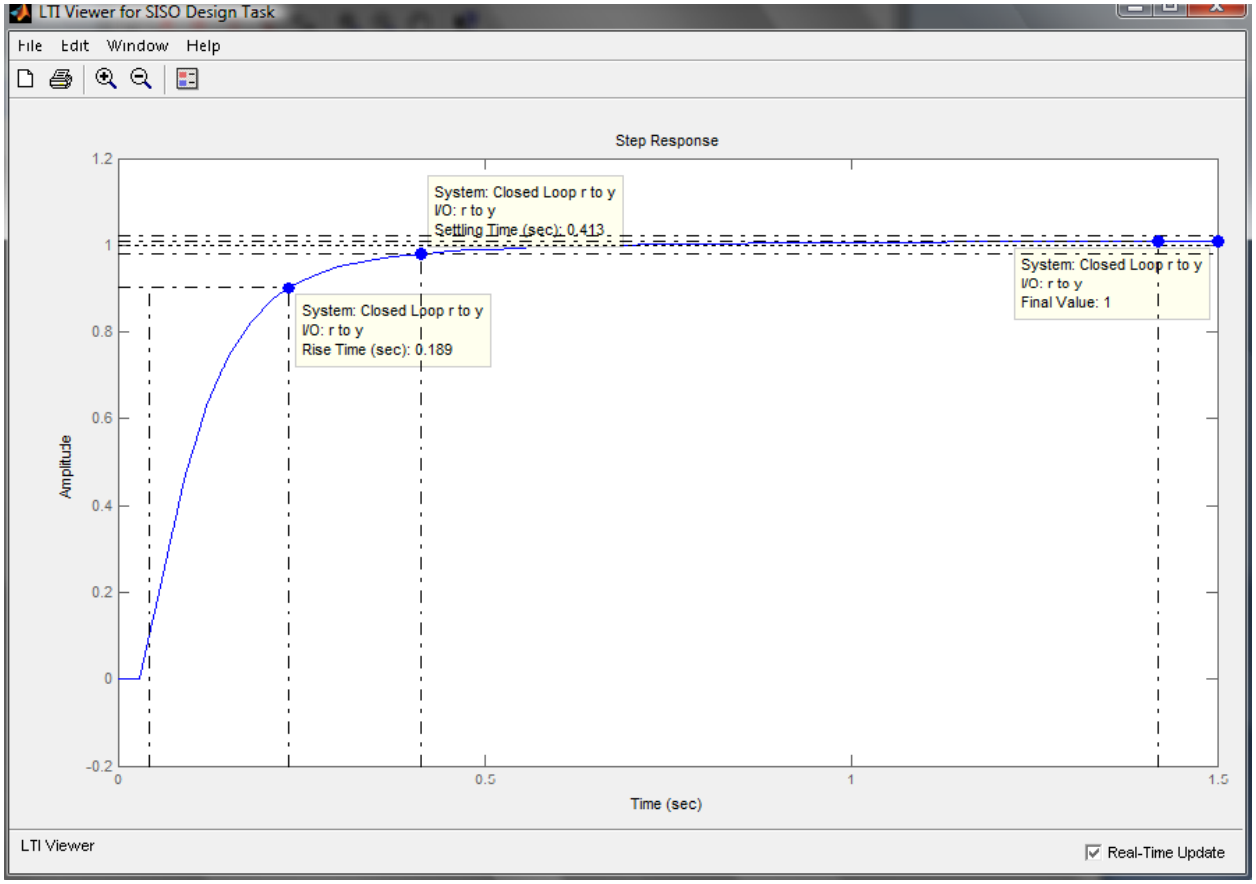


Figure 29: Step response obtained while tuning the system to design the controller for S2 servo using LQG method for pitch angle control

| | |
|---|---|
| **Rise time** | 0.189 sec |
| **overshoot** | 0 |
| **Peak amplitude** | 0 |
| **Settling time** | 0.413 sec |
| **Final value** | 1 |

Table 7: Step response values obtained while tuning the system to design the controller for S2 servo using LQG method for pitch angle control

The transfer function obtained is

***-5.745e005 s^6 - 1.985e008 s^5 - 2.292e010 s^4 - 1.059e012 s^3 - 3.546e012 s^2 - 1.962e012 s - 1.819e010***

**------------------------------------------------------------------------------------------------------------**

***s^7 + 2026 s^6 + 2.033e006 s^5 + 4.049e008 s^4 + 2.421e010 s^3 + 3.833e010 s^2 + 1.138e009 s***

**Designing the controller for servo S3**

Root locus and bode plots

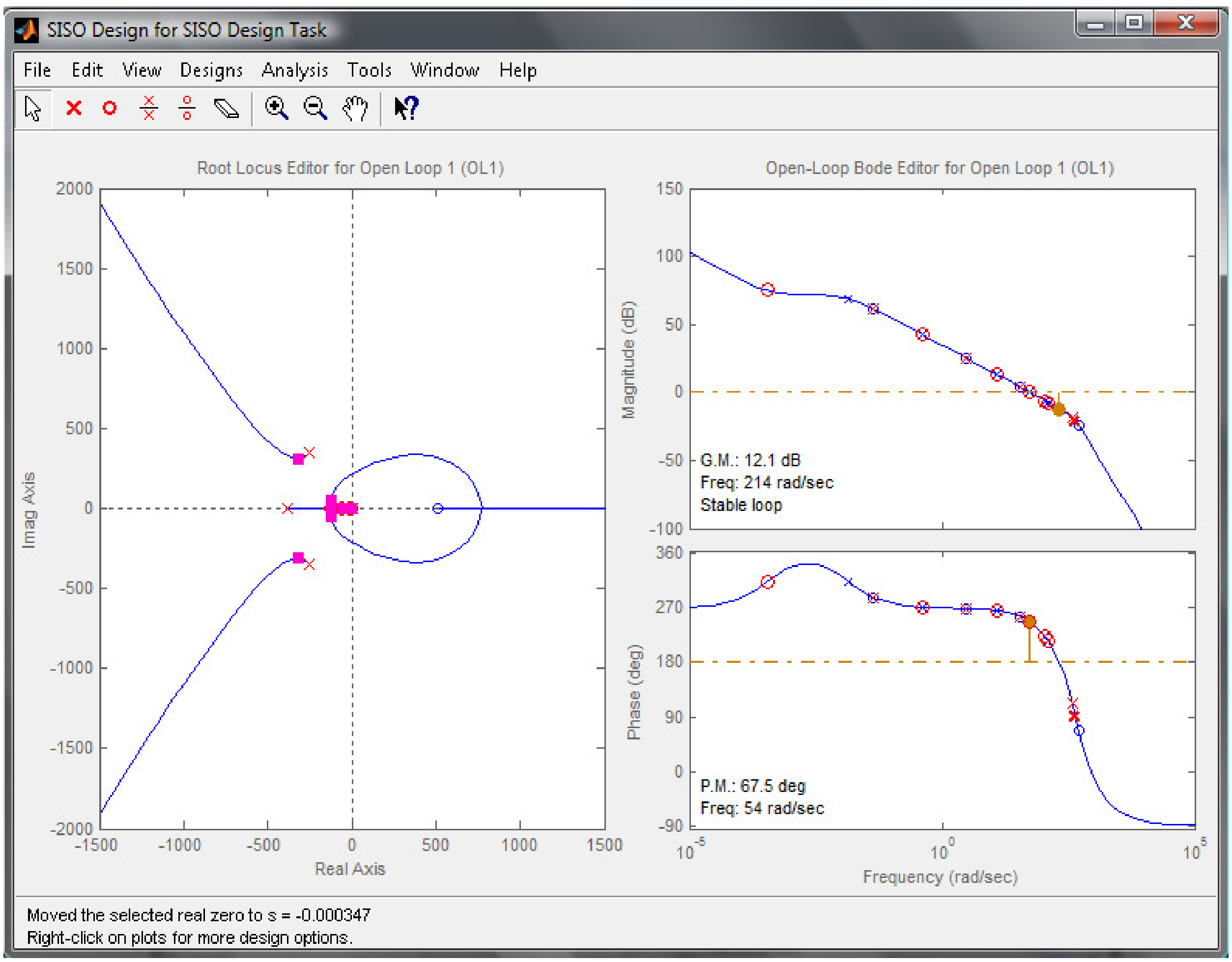


Figure 30: Root locus and bode plots while tuning the system to design the controller for S3 servo using LQG method for pitch angle control

Step response

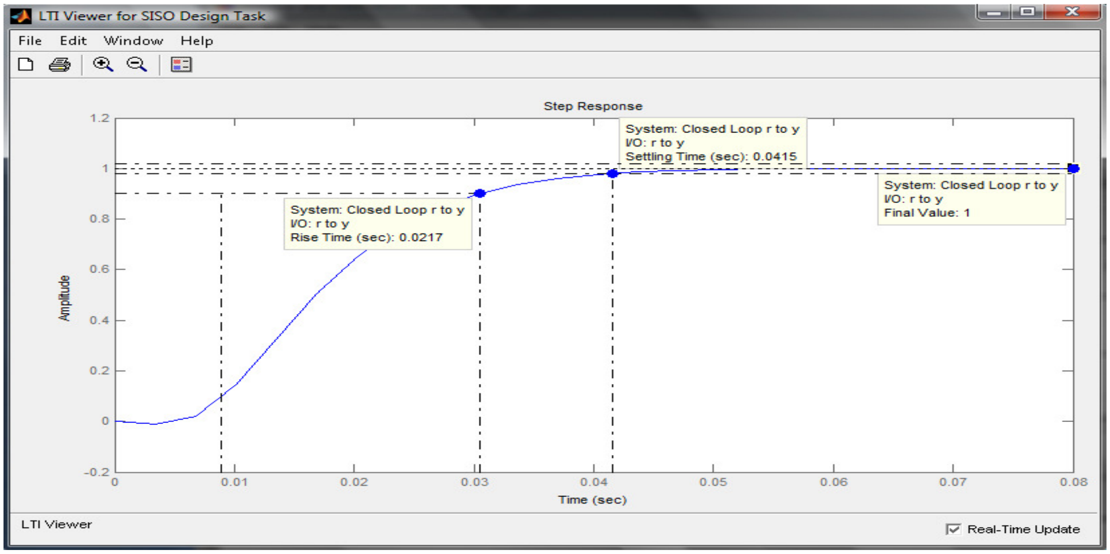


Figure 31: Step response obtained while tuning the system to design the controller for S3 servo using LQG method for pitch angle control

| | |
|---|---|
| **Rise time** | 0.0217 sec |
| **overshoot** | 0 |
| **Peak amplitude** | 0 |
| **Settling time** | 0.0414 sec |
| **Final value** | 1 |

Table 8: Step response values obtained while tuning the system to design the controller for S3 servo using LQG method for pitch angle control

The transfer function obtained is

***3.431e006 s^6 + 1.062e009 s^5 + 1.085e011 s^4 + 3.956e012 s^3 + 3.462e013 s^2 + 1.409e013 s + 4.881e009***

**------------------------------------------------------------------------------------------------------------**

***s^7 + 926.9 s^6 + 4.133e005 s^5 + 8.51e007 s^4 + 2.731e009 s^3 + 7.363e009 s^2 + 3.235e008 s***

**Simulation**

Using the transfer functions derived for the system model and controller, total system is simulated in simulink. The figure 32 shows the simulated model. The input is unit step signal. The helicopter image is the subsystem. This subsystem is shown in the figure 34. The output is verified in the oscilloscope.

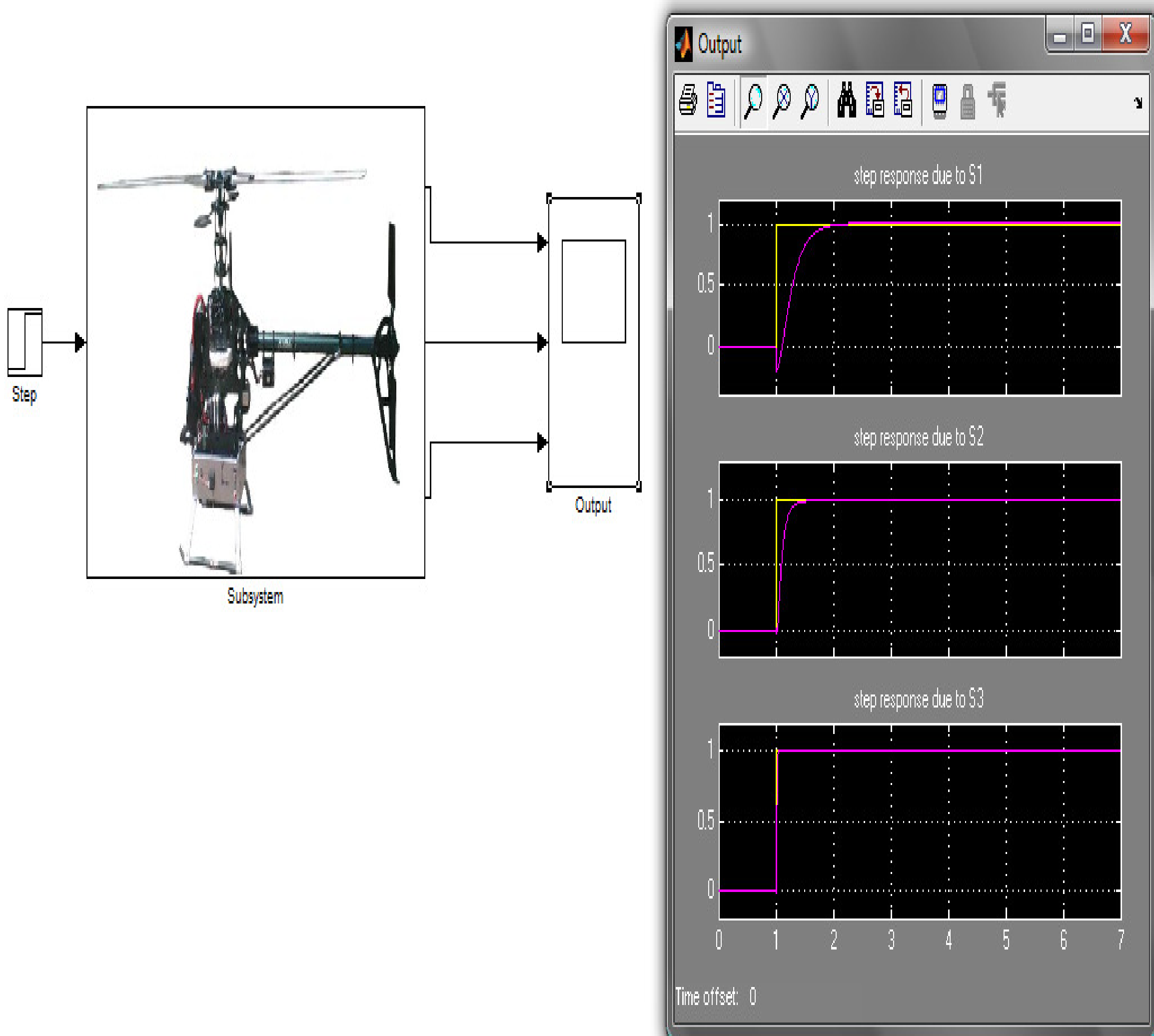


Figure 32: Simulated control system for the pitch angle

Scope output

The yellow color graph represents the input unit step signal. The pink color graph represents the helicopter system output for three servos (S1, S2 and S3). The values on X axis represent the time in seconds and the values on the Y axis represent the amplitude.

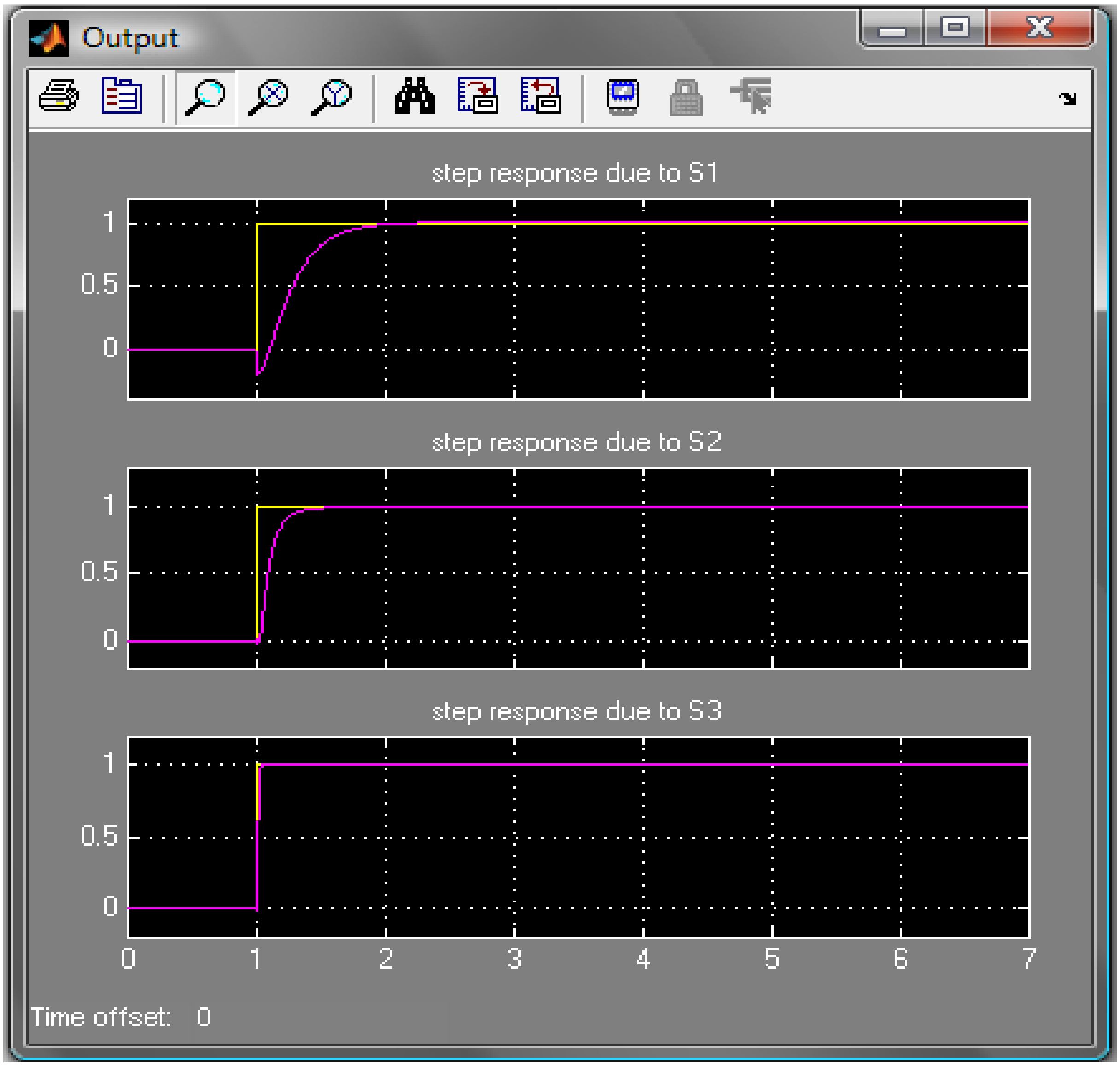


Figure 33: The scope output for the simulated Pitch angle control system

Subsystem

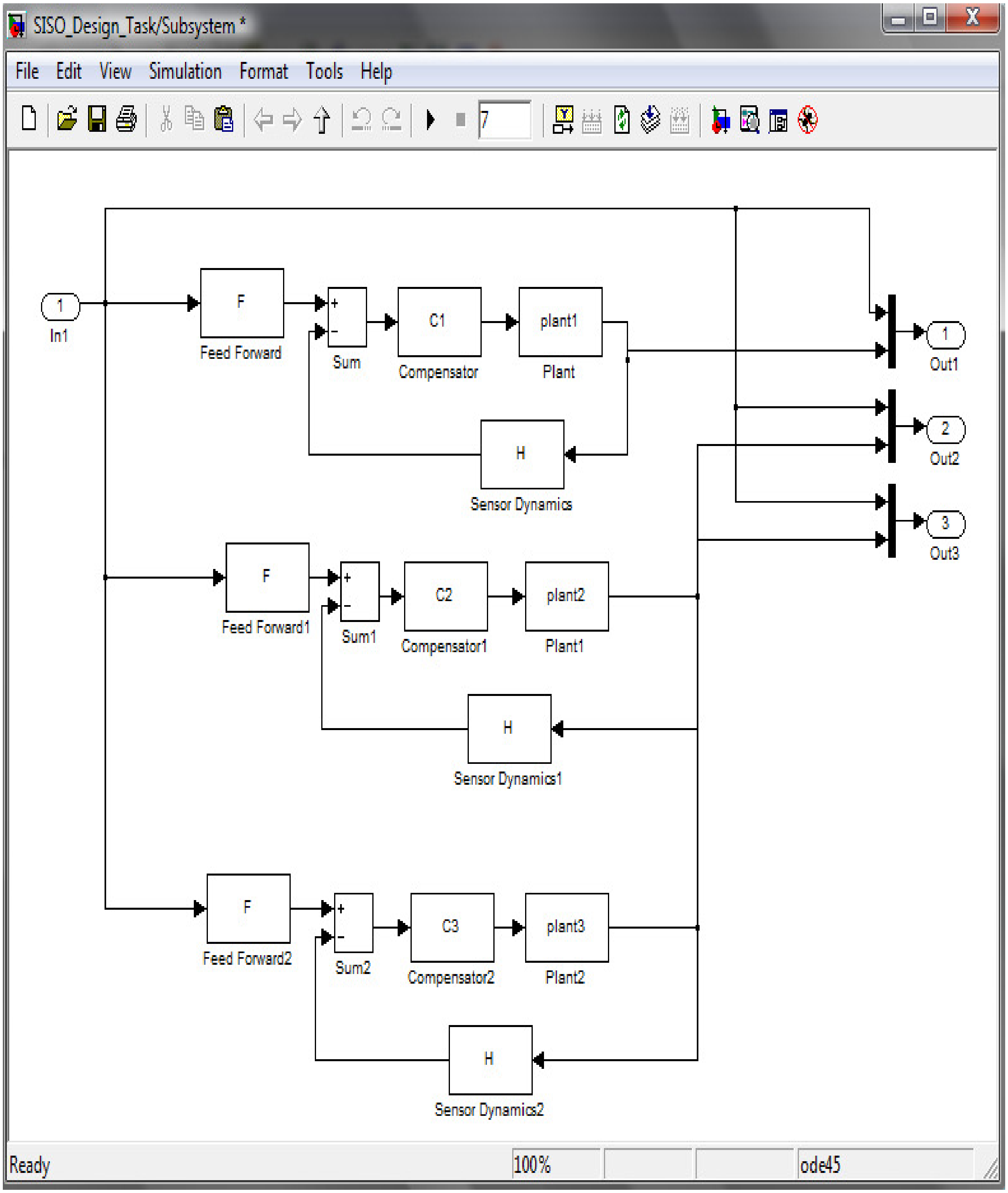


Figure 34: The subsystem for the pitch angle control system

### Total control system

The control systems for the roll and pitch angles is coupled together to design the total control system. The figure shows the total control system and its step response.

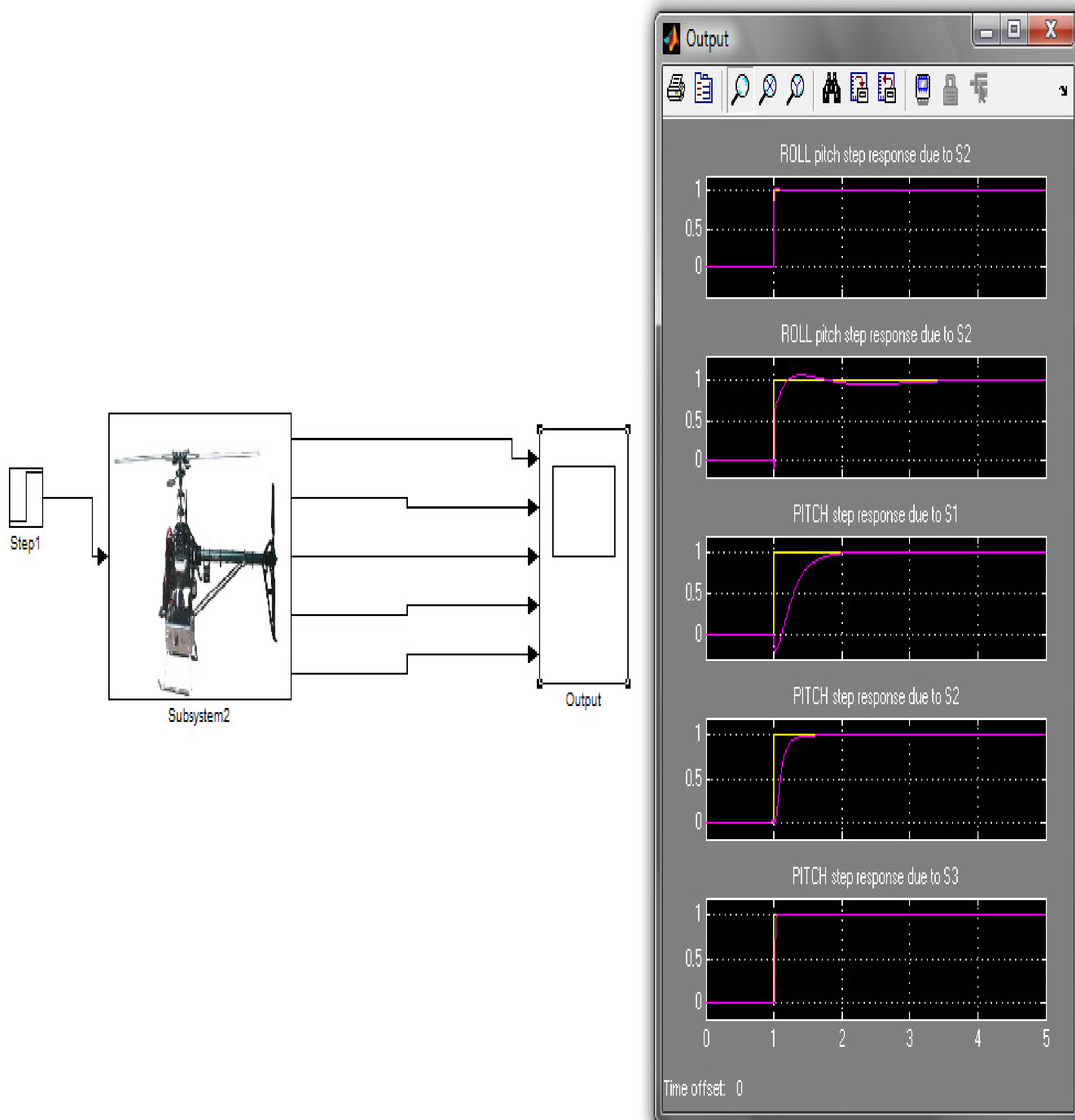


Figure 35: Simulated control system for the roll and pitch angle

Scope output

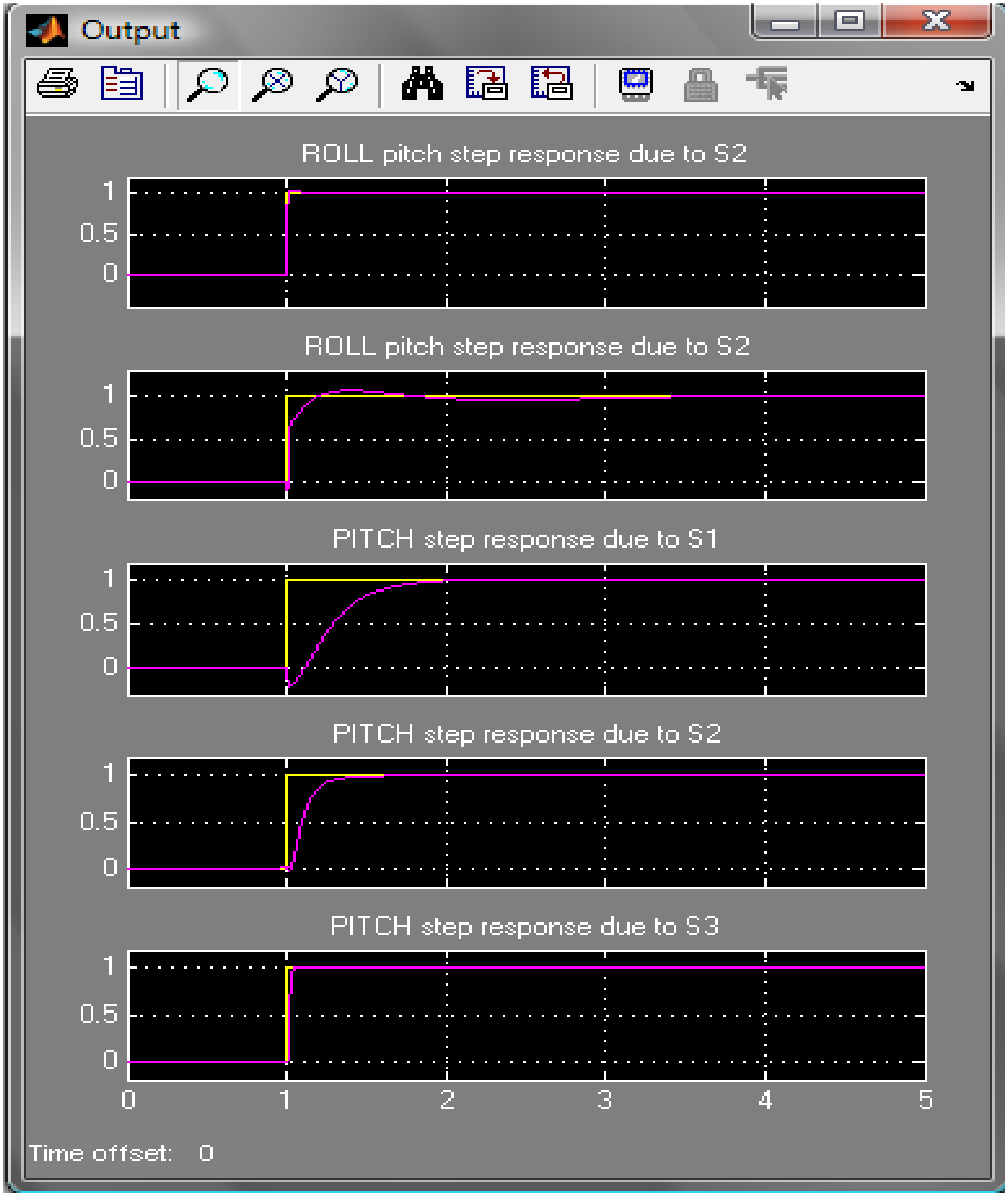


Figure 36: The scope output for the simulated roll and pitch angle control system

# CHAPTER 6

# CONCLUSION

An easy and accurate control system for the autonomous helicopter to control the roll and pitch angles is presented in this document. The problems in developing the accurate model of the helicopter are addressed. The solution to these problems is explained by developing the accurate model using system identification tool box. Advanced controlling methods like LQG (Linear Quadratic Gaussian) and IMC (Internal model controller) have been used to design the controller. Results showed that LQG method gives the better step response than PID and IMC controlling methods. Finally the system has been simulated in simulink to verify the system functionality.

# CHAPTER 7

# FUTURE WORK

- The generated transfer functions for the controller have to be converted into HDL or verilog code to check the helicopter response physically. Simulink HDL coder software can be used to serve this purpose.
- As explained above the controller transfer function has to be designed for other helicopter dynamics. Precautions have to be taken while testing the helicopter in lateral and longitudinal movements to avoid damage. An experienced pilot is required to test the helicopter for these dynamics because helicopter will not be tested on the jig platform.
- Programming has to be done for other modules like set point generator, Flight controller and Pulse width generator according to their functionality as explained in this document.

# REFERENCES


[1] FPGA Based Flexible Autopilot Platform for Unmanned Systems W. Alvis, S. Murthy, K. Valavanis, W. Moreno, M. Fields, S. Katkoori, Mediterranean conference on control and automation July 27-29, 2007, Athens-Greece

[2] Applications of photogrammetric processing using an autonomous model helicopter, Henri Eisenbeiss, Institute of Geodesy and Photogrammetry, 2005, ETH Zurich CH-8093, Zurich, Switzerland.

[3] Autonomous mini helicopter: a powerful platform for mobile mapping, Henri Eisenbeiss, Institute of Geodesy and Photogrammetry, 2005, ETH Zurich CH-8093, Zurich, Switzerland.

[4] System identification modeling of a small scale unmanned rotorcraft by Bernard Mettler, Mark B. Tischler ,Takeo Kanade Journal of American helicopter society, January 2002

[5] A Flight-Dynamic Helicopter Mathematical Model with a Single Flap-Lag-Torsion Main Rotor by M. D. Takahashi, NASATechnicalMemorandum102267, February 1990.

[6] http://www.sciencedaily.com/releases/2008/09/080902171117.htm

[7] Weilenmann M.F. and Geering H.P. A test bench for rotorcraft hover control. Journal of Guidance, Control, and Dynamics, 17:729–736, 1994.

[8] Gavrilets V., Mettler B., and Feron E. Nonlinear model for a small-size acrobatic helicopter. In AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA 2001-4333. August 2001.

[9] Construction, Modeling and Control of a Mini Autonomous UAV Helicopter, Proceedings of the IEEE International Conference on Automation and Logistics
Qingdao, China September 2008.

[10] Budiyono, A, Sudiyanto, T and Lesmana, H. (2007) First principle approach to modeling of small scale helicopter, International Conference on Intelligent Unmanned Systems, ICIUS, Bali, Indonesia

[11] An Algebraic Approach for the MIMO Control of Small Scale Helicopter A. Budiyono* and T. Sudiyanto Center for Unmanned System Studies Institut Teknologi Bandung, Indonesia ICIUS 2007 Oct 24-25, 2007 Bali, Indonesia

[12] Optimal Tracking Controller Design for a Small Scale Helicopter; Agus Budiyono* and Singgih S. Wibowo, ICIUS 2007; Oct 24-25, 2007; Bali, Indonesia

[13] Marco La Civita, William C. Messner, Takeo Kanade Modeling of Small-Scale Helicopters with Integrated First-Principles and System-Identification Techniques , Presented at the American Helicopter Society 58th Annual Forum, Montreal, Canada, June 11-13, 2002.

[14] Bogdanov A., Wan E., Harvey G., SDRE Flight Control For XCell and R-Max Autonomous Helicopters. Proceedings of the AIAA Guidance Navigation and Control Conference, OGI School of Science and Engineering, Austin, USA, 2004

[15] http://rtime.felk.cvut.cz/helicopter

[16] http://www.rotomotion.com

[17] http://www.procerusuav.com

[18] http://www.micropilot.com

[19] http://www.auav.net/ezi-nav.htm

[20] http://www.o-navi.com.

[21] FPGA Based Flexible Autopilot Platform for Unmanned Systems Mediterranean conference on control automation, July 27-29, 2007, Athens, Greece.

[22] http://www.microstrain.com/3dm-gx1.aspx

# BIBLIOGRAPHY

M. D. Takahashi, **"A Flight-Dynamic Helicopter Mathematical Model with a Single Flap-Lag-Torsion Main Rotor",** NASATechnicalMemorandum102267, February 1990.

A. Budiyono and T. Sudiyanto, **"An Algebraic Approach for the MIMO Control of Small Scale Helicopter"**, Center for Unmanned System Studies, Indonesia ICIUS 2007 Oct 24-25, 2007 Bali, Indonesia

Henri Eisenbeiss, **"Applications of photogrammetric processing using an autonomous model helicopter",** Institute of Geodesy and Photogrammetry, ETH Zurich CH-8093, Zurich, Switzerland, 2005.

Henri Eisenbeiss, **"Autonomous mini helicopter: a powerful platform for mobile mapping",** Institute of Geodesy and Photogrammetry, 2005, ETH Zurich CH-8093, Zurich, Switzerland,2005.

Budiyono, A, Sudiyanto, T and Lesmana, **"First principle approach to modeling of small scale helicopter",** International Conference on Intelligent Unmanned Systems, 2007, ICIUS, Bali, Indonesia

Guowei Cai Cai, A.K. Chen, B.M. Lee, T.H, **"Construction, Modeling and Control of a Mini Autonomous UAV Helicopter"**, Proceedings of the IEEE International Conference on Automation and Logistics Qingdao, China September 2008.

W. Alvis, S. Murthy, K. Valavanis, W. Moreno, M. Fields, S. Katkoori , **"FPGA Based Flexible Autopilot Platform for Unmanned Systems",** Mediterranean conference on control and automation July 27-29, 2007, Athens-Greece

Gavrilets V, Mettler B, and Feron E, **"Nonlinear model for a small-size acrobatic helicopter",** AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA 2001-4333. August 2001.

Marco La Civita, William C. Messner, Takeo Kanade**, "Modeling of Small-Scale Helicopters with Integrated First-Principles and System-Identification Techniques",** Presented at the American Helicopter Society 58th Annual Forum, Montreal, Canada, June 11-13, 2002.

Agus Budiyono and Singgih S. Wibowo, **"Optimal Tracking Controller Design for a Small Scale Helicopter",** ICIUS 2007; Oct 24-25, 2007; Bali, Indonesia.

Bogdanov A., Wan E., Harvey G., **"SDRE Flight Control For XCell and R-Max Autonomous Helicopters"**, Proceedings of the AIAA Guidance Navigation and Control Conference, OGI School of Science and Engineering, Austin, USA, 2004

Bernard Mettler, Mark B. Tischler ,Takeo Kanade, **"System identification modeling of a small scale unmanned rotorcraft"**, Journal of American helicopter society, January 2002

Weilenmann M.F. and Geering H.P. **"A test bench for rotorcraft hover control",** Journal of Guidance, Control, and Dynamics, 17:729–736, 1994.

http://www.sciencedaily.com/releases/2008/09/080902171117.htm - **"Autonomous helicopters with artificial technique"**

http://rtime.felk.cvut.cz/helicopter- "**RAMA UAV Control System project details"**

http://www.rotomotion.com-**"Rotomotion mini UAV vehicles website details"**

http://www.procerusuav.com**- "Procerus Technologies autopilot System for mini and micro Unmanned Aerial Vehicles website details"**

http://www.micropilot.com-**"Micropilot miniature autopilot website"**

http://www.auav.net/ezi-nav.htm- **"Details of EZI_NAV autopilot system"**

http://www.o-navi.com-**"O-navi inertial sensor website details"**

http://www.microstrain.com/3dm-gx1.aspx **- "Microstrain orientation sensor details"**

# APPENDICES

## Appendix –A: Matlab code

**Matlab code for system identification and to design the controller**

```
% loading the data servo values and roll angle values

clear all;

clc;

load 'servo.txt';

load 'rollpitch.txt';

load 'angular.txt';

roll1 = rollpitch(:,1);

pitch1 = rollpitch(:,2);

servo1=servo(:,1);

servo2=servo(:,2);

servo3=servo(:,3);

%% plant1

%Finding the helicopter (plant) transfer function for the input S1,output


plant11cont= d2c (plant11);

plant21cont= d2c (plant21);

plant =tf (plant11cont*plant21cont);

plant = tfdata(plant);

num = plant(1);

den= plant(2);

plant1 = tf (num,den);

% Designing the compensator or controller
```

```
ltiview(plant1);

sisotool(plant1);

%%

%%plant2
%Finding the helicopter (plant) transfer function for the input S2,output

plant12cont= d2c (plant12);

plant22cont= d2c (plant22);

plant =tf (plant12cont*plant22cont);

plant = tfdata(plant);

num = plant(1);

den= plant(2);

plant2 = tf (num,den);

% Designing the compensator or controller

ltiview(plant2);

sisotool(plant2)

%%

%Finding the helicopter (plant) transfer function for the input S3,output

plant13cont= d2c (plant13);

plant23cont= d2c (plant23);

plant =tf (plant13cont*plant23cont);

plant = tfdata(plant);

num = plant(1);

den= plant(2);

plant3 = tf (num,den);

% Designing the compensator or controller
```

```
ltiview(plant3);
sisotool(plant3)
```

### Matlab code for experimental data gathering

```
%% start
close all;
clear all;
clc;
s = serial('COM6','BaudRate',115200,'Timeout',10);
command_reset = hex2dec('30');  %ascii 0
command_go = hex2dec('31');     %ascii 1
replyLength = 15;          % returns 15 bytes
loopLength = 1500;         % read length
% anonymous function to calculate the unsigned 16bit int
un = @(msb,lsb) double(256*msb+lsb);
%% Connect and read from the IMU
    fopen(s);              % Open the serial connection
  fwrite(s,command_reset);     % Put the IMU in polling mode (if not already)
  pause(0.2);
  if s.BytesAvailable > 0 % If there are bytes in the buffer, remove them.I think it is "clear s"
      fread(s,s.BytesAvailable);
  end

  % Initialise storage to make it run faster
%    x= zeros(loopLength,replyLength);%23x10 matrix
```

```
%    userData=zeros(loopLength,8);
%    scaling1 = [ones(loopLength,3)];scaling2=[ 360/65536*ones(loopLength,2)];
%        scaling3=[8500/32768000*ones(loopLength,2)];scaling4=[ ones(loopLength,1)];
%    userData2 = zeros(loopLength,12);
   %raw_data = zeros(replyLength*loopLength,1);
```

**%% Data Gathering**

```
fwrite(s,command_go);
pause(0.1);
for i=1:loopLength*replyLength
    raw_data(i)=double(fread(s,1));
end


%% data manipulation
ii=0;
fwrite(s,command_reset);
n=1;
for i=1:1:replyLength*loopLength
    n=1;
    if((raw_data(i) == 65)&&(raw_data(i+14) == 90))
        ii = ii + 1;
%       for jj=2:1:12
%           userData2(ii,jj-1) = raw_data(i+jj-1);
%       end
%       userData2(ii,12) = 256*raw_data(i+12) + raw_data(i+13);
            for jj=2:1:4
```

```
            userData(ii,jj-1) = raw_data(i+jj-1);
        end
        for jj=2:1:5
            kk = 2*jj + 1;
            if(raw_data(i+kk-1)>= 128)
                raw_data(i+kk-1) = raw_data(i+kk-1) - 128;
                flag = 1;
            else
                flag = 0;
            end
            userData(ii,jj+2) = 256*raw_data(i+kk-1) + raw_data(i+kk+1-1);
            if(flag == 1)
                userData(ii,jj+2) = -1 * userData(ii,jj+2);
            end
        end
        userData(ii,8) = 256*raw_data(i+12) + raw_data(i+13);
        %i = i + 14;
        n=14;
    end
    end

%% Cleanup the serial port.

    fclose(s);% close the serial connection
    fclose all;
    delete(s);
```

```
dfile1=fopen('rollpitch.txt','wt');
dfile2=fopen('servo.txt','wt');
dfile3=fopen('angular.txt','wt');
dfile4=fopen('time.txt','wt');

for a=1:length(userData)
    fprintf(dfile2,'%6.4f %6.4f %6.4f\n',userData(a,1),userData(a,2),userData(a,3));
    fprintf(dfile1,'%6.4f %6.4f \n',userData(a,4),userData(a,5));
    fprintf(dfile3,'%6.4f %6.4f \n',userData(a,6),userData(a,7));
    fprintf(dfile4,'%6.4f \n', userData(a,8));
end
%% Close files
 fclose(dfile1);
 fclose(dfile2);
 fclose(dfile3);
 fclose(dfile4);

%% Calc time

for a=1:length(userData)
    T(a)=0.0065536*userData(a,8);
    Samp(a)=a;
end
T=T';
%% Display servo data
```

```
figure(1);
plot(T,userData(:,1),T,userData(:,2),...
   T,userData(:,3));
xlabel('time(ticks)');ylabel('time (ms)');title('servo values');
legend('S1.ele','s2.ail','s3.aux');
axis tight;
%% Display the euler angle data
figure(2);


plot(T,userData(:,4),T,userData(:,5)  );
xlabel('time(ticks)');ylabel('angle (degrees)');title('Teuler angle data');
legend('roll','pitch');
axis tight;
%% Plot the angular rate data
figure(3);
 plot(T,userData(:,6),T,userData(:,7));
xlabel('time(ticks)');ylabel('Angular rate (deg/s)');title('Angular Velocity');
legend('X-ang rate','Y-ang rate');
axis tight;
fclose all;
%% Plot Time
figure(4);
plot(Samp,T);
xlabel('sample');ylabel('time(sec)');title('timerticks*0.01');
axis tight;
delete(s)
```

```
%% Everything for Roll
figure(5);
subplot(2,1,1);
    plot(T,userData(:,1),T,userData(:,2),T,userData(:,3));
    axis tight;
    legend('S1 Elev','S2 Aile','S3 Aux1')
    title('Everything Roll');ylabel('Servo widths(0.01ms)');
subplot(2,1,2);
    plot(   T,userData(:,4),...
            T,userData(:,6));
    axis tight;
    legend('Roll','roll rate');
    xlabel('Time(seconds)');ylabel('Roll Stuff');
```

# Appendix –B: Hardware components

**T-Rex 600E Helicopter**

An electric motor operated T-Rex 600E Helicopter has been assembled and a payload is attached to carry the required components like sensors, circuit board, batteries etc. The motor has the capacity of lifting about 10lbs including its own weight. The Figure 5 shows assembled helicopter platform having the payload at its bottom.

Figure 37: T-Rex 600E Helicopter

**Processing Platform**

FPGA is selected as a processing platform because of its ability of parallel processing that means it executes the commands in parallel unlike micro controller which executes sequentially. For testing purpose Spartan-3E is used and Virtex4-FX12 for the final design. The Figure 6 shows the Avnet Virtex4-FX12 board which is attached to the baseboard and it is shown separately in the small window.

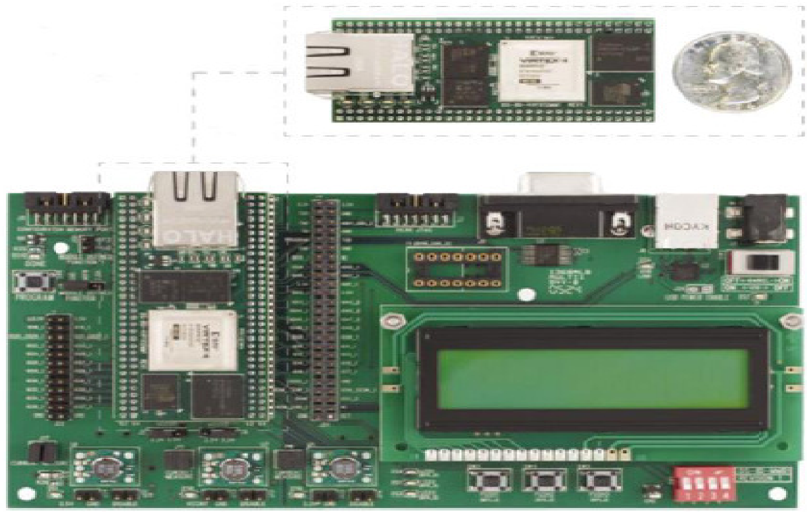

Figure 38: Avnet Virtex4-FX12 board

**Parallax PING)) Ultrasonic distance sensor**

Parallax PING)) Ultrasonic distance sensor is acquired for altitude measurements during takeoff and landing states. It has the capacity of measuring the distance from 2cm to 3 m which is enough for safe take off and landing.


Figure 39: Parallax PING)) Ultrasonic distance sensor

**Microstrain -3DMgx1 orientation sensor (IMU)**

The IMU is used to get the information about the orientation of the helicopter. This IMU is capable of providing the information about Euler angles, Angular velocities and accelerations in

X, Y, Z directions in the form of data sentence according the specific input commands. The Figure 8 shows the IMU and the table gives the details of input data and conversion factors.

Figure 40: Microstrain -3DMgx1 orientation sensor

| OUTPUTS | INDEX # | VALUE BYTE 1 | VALUE BYTE 2 | FACTOR FOR UNIT CONVERSION | UNITS |
|---|---|---|---|---|---|
| HEADER | 0 | 0X31 | NONE | | |
| ROLL | 1-2 | MSB | LSB | VALUE DIVIDED BY (360/65536) | DEGREES |
| PITCH | 3-4 | MSB | LSB | | |
| YAW | 5-6 | MSB | LSB | | |
| X_ACC | 7-8 | MSB | LSB | VALUE DIVIDED BY (32768000 / 7000) | GRAVITY VECTORS G = 9.8 $m/s^2$ |
| Y_ACC | 9-10 | MSB | LSB | | |
| Z_ACC | 11-12 | MSB | LSB | | |
| X_ANG_RATE | 13-14 | MSB | LSB | VALUE DIVIDED BY (32768000 / 8500) | RADIANS PER SECOND |
| Y_ANG_RATE | 15-16 | MSB | LSB | | |
| Z_ANG_RATE | 17-18 | MSB | LSB | | |
| TIMER TICKS | 19-20 | MSB | LSB | TIMERTICKS * 0.01 | SECONDS |
| CHECKSUM | 21-22 | MSB | LSB | | |

Table 9: Microstrain -3DMgx1 sentence decoding information