

Copyright © 2019 Acrome Inc. All rights reserved.

Acrome Inc.
ITU ARI4 Science Park
Maslak, Istanbul
Turkey
info@acrome.net
Phone: +90 212 807 04 56
Fax: +90 212 285 25 94

Printed in Maslak, Istanbul

For more information on the solutions Acrome Inc. Offers, please visit the web site at:

<http://www.acrome.net>

This document and the software described in it are provided subject to a license agreement. Neither the software nor this document may be used or copied except as specified under the terms of that license agreement. All rights are reserved and no part may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Acrome Inc.

ACKNOWLEDGEMENTS

Special thanks to Acrome myControl Team for their supports in embedded control and preparation of the courseware.

Contents

1	Components of the Helicopter System	4
1.1	Incremental Rotary Encoder	4
1.2	Brushed DC Motor	6
1.3	Arduino Mega 2560	7
1.4	Mechanics of 1-DOF Helicopter	8
2	Pulse Width Modulation	9
2.1	Understanding Pulse Width Modulation (PWM)	9
2.1.1	In-Lab Exercise	10
3	System Modelling and Cascade Control Structure	11
3.1	System Modelling	11
3.2	Cascade Control of 1-DOF Helicopter System	13
4	Performance Measures	14
4.1	Understanding Percentage Overshoot, Peak Time, Settling Time and Steady State Error	14
4.1.1	Damping Ratio (ξ)	14
4.1.2	Natural Frequency (ω_n)	14
4.1.3	Percentage Overshoot	14
4.1.4	Peak Time (t_p)	16
4.1.5	Settling Time (t_s)	16
4.1.6	Steady State Error	16
4.1.7	In-Lab Exercise	17
5	Control System Design	18
5.1	Academic PID controller	18
5.2	Parallel PID controller	19
5.3	Root Locus	19
5.4	P Controller	20
5.4.1	In-Lab Exercise	20
5.5	PD Controller	21
5.5.1	Sample Design with PD controller	22
5.5.2	In-Lab Exercises	23

5.6	PV Controller	25
5.6.1	Sample Design with PV controller.....	26
5.6.2	In-Lab Exercises.....	27
5.7	PID Controller	29
5.7.1	Sample Design with PID controller	30
5.7.2	In-Lab Exercises.....	32

1 Components of the Helicopter System

The main parts of “1-DOF Helicopter” can be seen in the figure below.

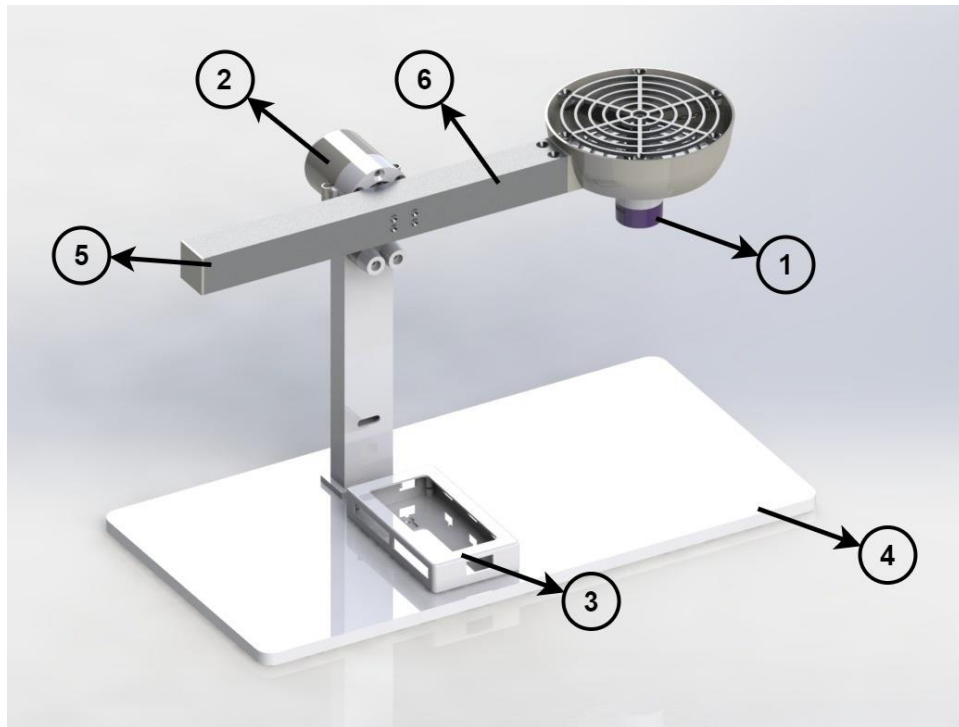


Figure 1-1 - 1-DOF Helicopter

Figure 1.1: Components: **1.** Brushed DC Motor **2.** Incremental Rotary Encoder **3.** Motor Driver Box **4.** Base of the System **5.** Balancing Block **6.** Beam

1.1 Incremental Rotary Encoder

When the encoder is rotated, an incremental rotary provides only cyclic outputs. Because the rotary encoders have a low cost and the ability to obtain information about position or velocity of motion, the most widely used of all rotary encoders are the incremental rotary encoders. Light optics are used by the optical encoder in order to identify unique positions. Optical encoders could have a 4 million count resolution per revolution. Therefore, the optical encoder is preferred.



Figure 1-2 - Incremental Rotary Encoder

Table 1.1: Technical Details for the Incremental Rotary Encoder

Electrical and Mechanical Specifications	
Slew Speed	3500rpm
Shaft Load	80N
Operating Temperature	-20 °C.... +80°C
Supply Voltage	5V DC, 8-24V DC or 5-24V DC
Shaft diameter	40mm
Certificate	IP54
Current Requirement	<40mA (24VDC)
Output Channels	Push-Pull, TTL or HTL Linedriver
Output Phase	A,B,Z or A,A';B,B';Z,Z'

An optical encoder is able to perform with a phase array under much tougher circumstances when a combination of durability and resolution is needed. In this helicopter system, out of the three output phases only two are required, which are A and B. The Z-Terminal which is called a zero or reference signal gives a single pulse for every revolution. This single pulse can be used to find out a reference position.

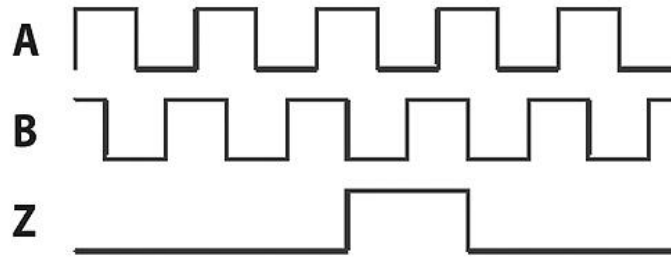


Figure -1-3: Quadrature Encoder Output Signals

1.2 Brushed DC Motor

Pololu 37D (No Gearbox) is powerful, high speed (11000rpm) brushed DC motor, and it is available with or without an integrated 64 CPR quadrature encoder on the motor shaft. The motor is intended for operation at 12 V, though in general, this kind of motor can run at voltages above and below the nominal voltage, and they can begin rotating at voltages as low as 1 V.



Figure 1-4 - Pololu 37D Brushed DC Motor (No Gearbox)

Table 1.2: Technical Details for the Brushed DC Motor

Electrical and Mechanical Specifications	
Gear ratio	1:1
No-load speed @ 12V	11000
No-load current @ 12V:	0.3 A
Stall current @ 12V	5 A
Stall torque @ 12V:	0.3 kg·cm

1.3 Arduino Mega 2560

Arduino is a portable embedded device that allows users to design and control robotic or mechatronic systems.



Figure 1-3 : Arduino Mega

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button

2 Pulse Width Modulation

2.1 Understanding Pulse Width Modulation (PWM)

Pulse Width Modulation (PWM) is a solid method to encode analog signals digitally by generating a square wave with a certain duty cycle that is modulated to encode the analog signal. PWM technique is widely used in various applications such as switching power supplies and motor control.

PWM technique produces a square PWM signal that has several properties like period, frequency, amplitude and duty cycle. Period is the time interval required to repeat the signal and frequency is the number of occurrences of these signals in a second. Amplitude shows the voltage level of the high or low state of the PWM signal. Duty cycle is used to describe high state of the signal as a percentage of the total period and finally, pulse width is the time that signal is at the high state.

A typical PWM signal and its properties are shown in the Figure 2.1:

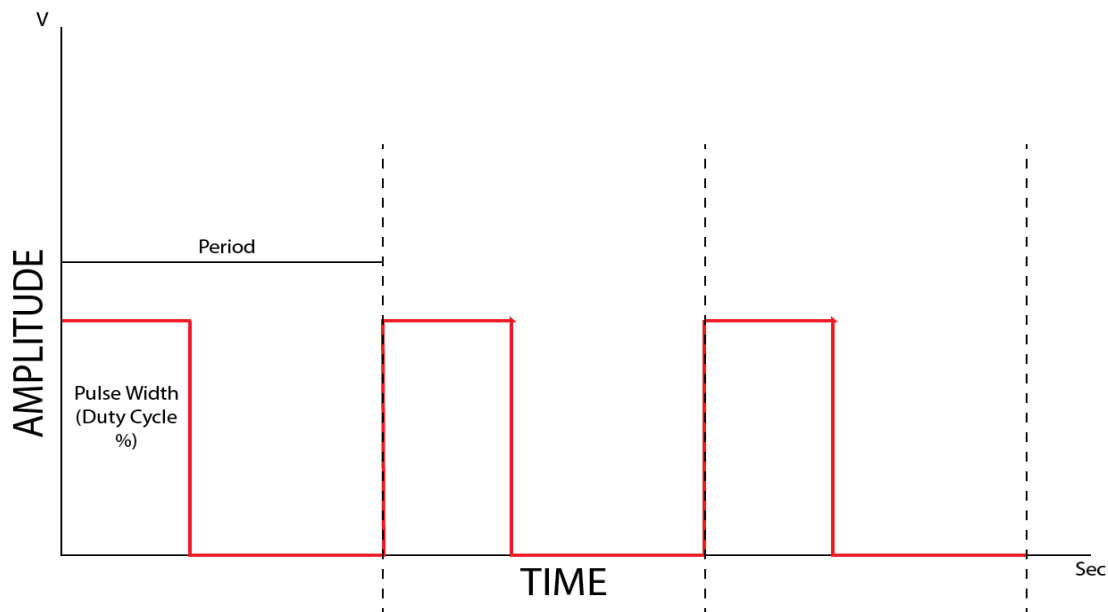


Figure 2.1: Typical PWM Signal with Its Properties

2.1.1 In-Lab Exercise

1. Open “GeneratingandReadingPWM.slx” where Figure 2.2 shows its block diagram.

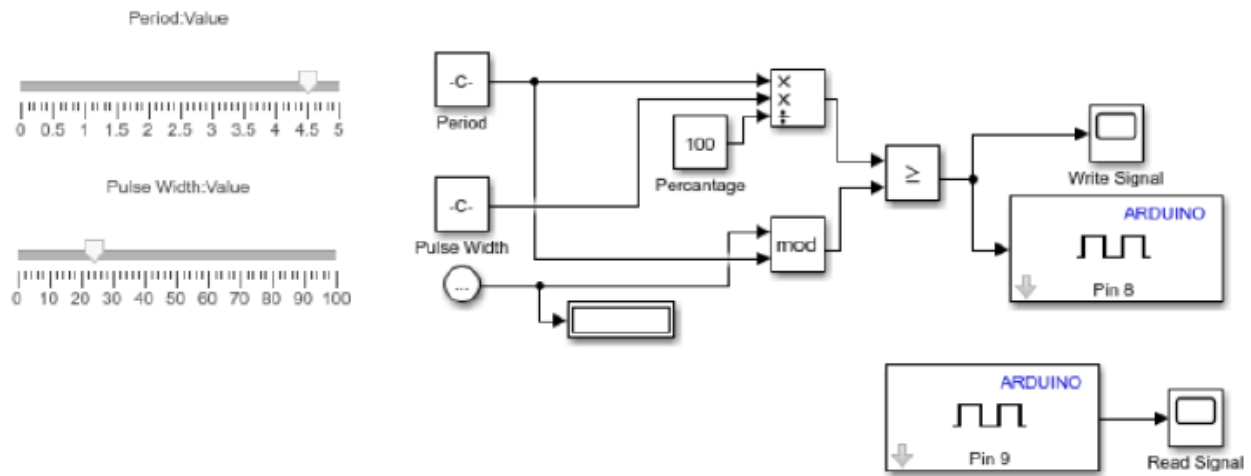


Figure 2.2: Front Panel of “GeneratingandReadingPWM.slx”

2. If you want to see generated signal, please first connect pins 8 and 9 of the Arduino and run the program
3. Change the percentage duty cycle and the period as desired,
4. Double click and open Read Signal and Write Signal Scope. Observe simulated PWM from the oscilloscope,
5. Compare the consistency of the performance obtained from sliders and scopes.

3 System Modelling and Cascade Control Structure

3.1 System Modelling

The physical model of 1-DOF helicopter system is shown in Figure 3.1. System parameters are explained in Table 3.1. These parameters will be used for obtaining the mathematical model of the 1-DOF Helicopter system. Beam is directly mounted to the encoder in order to measure the beam angle. When the motor is energized and F_h is greater than the weight of the helicopter $m_h \cdot g$, the helicopter rises.

Table 3.1: 1-DOF Helicopter Parameters

Symbols	Definition	Value	Unit
l_1	Distance between the balancing block and the pivot point	[Min – Max] [0.1365-0.1375]	[m]
l_2	Distance between the motor and the pivot point	0.203	[m]
m_h	Total mass of the DC motor and the cage	0.249	[kg]
m_b	Mass of the block	0.3565	[kg]
J_e	Moment of inertia of the system	3.5×10^{-5}	[kg * m ²]
g	Acceleration of gravity	9.81	[m/s ²]
I_s	Current for the motor	(Variable)	[ampere]
θ	Beam angle	(Variable)	[degrees]

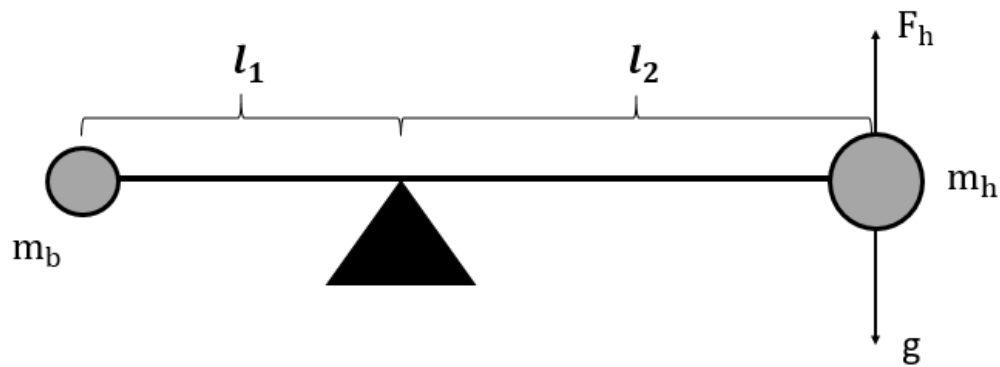


Figure 3.1: The Physical Model of the 1-DOF Helicopter

In order to obtain the model of the system, the moment equation should be written as:

$$J_e \cdot \ddot{\theta} = l_1 \cdot F_h - l_1 \cdot m_h \cdot g - k_f \cdot \dot{\theta} \quad (3.1)$$

k_f is the coefficient of viscous friction and its value is 7.589. F_h is the propeller motor total lift.

We can write G instead of $m_h \cdot g$ and $K \cdot I_s$ instead of F_h in the Equation 3.1:

$$J_e \cdot \ddot{\theta} = K \cdot l_1 \cdot I_s - l_1 \cdot G - k_f \cdot \dot{\theta} \quad (3.2)$$

K is the product of the force constant and the resistance of the motor and its value is 52.27. The inertia of the system about the pitch axis can be calculated as shown in below:

$$J_e = m_h \cdot l_1^2 + m_b \cdot l_2^2 \quad (3.3)$$

If we ignore $l_1 \cdot G$, the moment equation 3.1 can be obtained as:

$$J_e = m_h \cdot l_1^2 + m_b \cdot l_2^2 \quad (3.3)$$

When the Equation 3.3 is transposed to Equation 3.1, the moment equation turns to:

$$J_e \cdot \ddot{\theta} + k_f \cdot \dot{\theta} = K \cdot l_1 \cdot I_s \quad (3.4)$$

Then the Laplace transformation is applied:

$$J_e \cdot s^2 \cdot \theta(s) + k_f \cdot s \cdot \theta = K \cdot l_1 \cdot I_s(s) \quad (3.5)$$

The system transfer function is found as:

$$\frac{\theta(s)}{I_s(s)} = \frac{K \cdot l_1}{J_e \cdot s^2 + k_f \cdot s} = \frac{30}{s^2 + 2.2 s} \quad (3.6)$$

3.2 Cascade Control of 1-DOF Helicopter System

1-DOF Helicopter has cascade structure as shown in Figure 3.2.

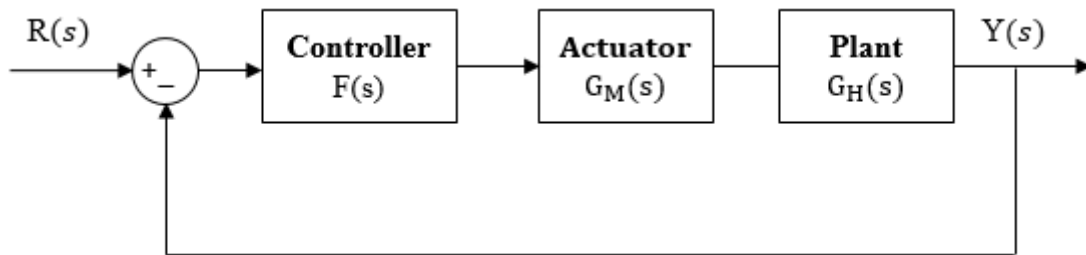


Figure 3.2: Block Diagram of the System

$G_M(s)$ is the model of the actuator and also equals to 1. Plant's transfer function was obtained in the previous section. When the parameters are replaced, block diagram is also shown as below:

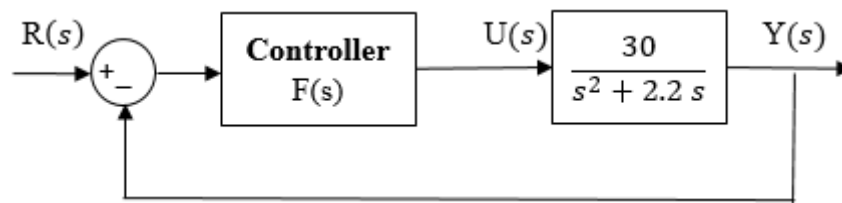


Figure 3.3: Block Diagram of the Control Loop

Designing of the controller will be explained in Chapter 5.

4 Performance Measures

Performance measures or criteria are defined for second order systems when a step input is applied. Different performance measures are determined considering the systems. Depending on the performance measures, damping ratio (ξ) and natural frequency (ω_n) are found. Also, if damping ratio and natural frequency are known, performance measures can be calculated. In this section, all parameters are investigated.

4.1 Understanding Percentage Overshoot, Peak Time, Settling Time and Steady State Error

4.1.1 Damping Ratio (ξ)

When step input is applied, the system gives a response depending on the damping ratio. Damping ratio and the system response coupling can be seen in Table 5.1. By using percentage overshoot (PO), damping ratio can be calculated.

$$\xi = \frac{-\ln(PO/100)}{\sqrt{\pi^2 + (\ln(PO/100))^2}}$$

4.1.2 Natural Frequency (ω_n)

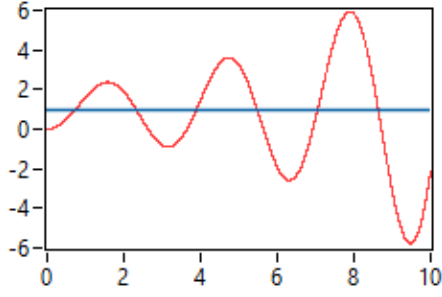
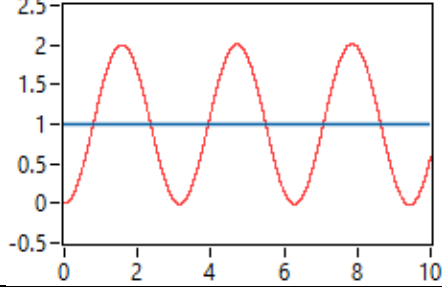
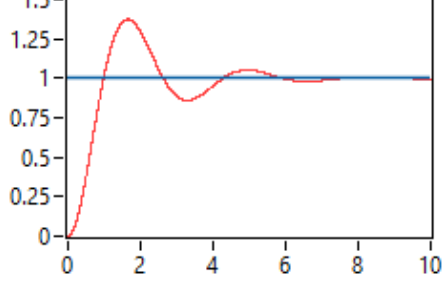
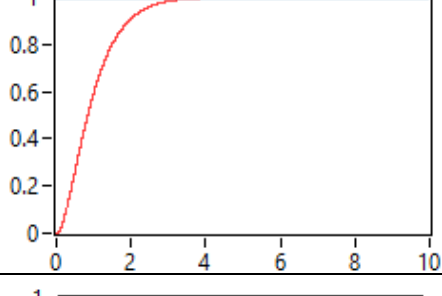
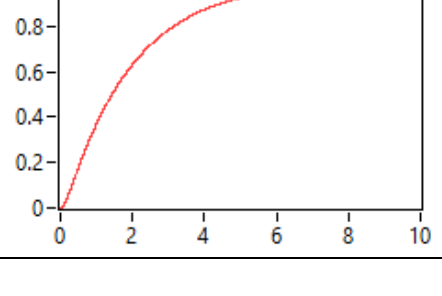
While damping ratio equals to “0”, natural frequency is the frequency of oscillation of the system. Natural frequency is calculated using the settling time or the peak time.

4.1.3 Percentage Overshoot

For a stable system, overshoot is the difference between maximum and final output values. Percentage overshoot is 100 multiplied by ratio of overshoot to final output value. Also, if there is a graph, percentage overshoot can be calculated, but if damping ratio is known, other formulas of percentage overshoot can be used.

$$\begin{aligned} \text{Percentage Overshoot} &= 100 * \frac{\text{Maximum Output Value} - \text{Final Output Value}}{\text{Final Output Value}} \\ &= 100 * e^{\frac{-\xi\pi}{\sqrt{1-\xi^2}}} \end{aligned}$$

Table 4.1: Damping Ratio, system behaviour and system response graph

Damping Ratio (ξ)	System Behaviour	System Input and Output
$\xi < 0$	Unstable	
$\xi = 0$	Undamped	
$0 < \xi < 1$	Underdamped	
$\xi = 1$	Critically Damped	
$1 < \xi$	Over damped	

4.1.4 Peak Time (t_p)

Peak time is the elapsed time between applying the step input until it has reached the maximum value. Also, peak time can be calculated with the help of the damping ratio and the natural frequency.

$$t_p = \frac{\pi}{\omega_n * \sqrt{1 - \xi^2}}$$

4.1.5 Settling Time (t_s)

Settling time is the time when the system enters 2% or 5% band of system response final value. 2% band will be used from now on. Also, there is a formula for calculating settling time as below:

$$t_s = \frac{4}{\xi * \omega_n}$$

4.1.6 Steady State Error

Steady state error is the difference between the reference input and the final output value. Most of the time, it is an undesirable situation where the system response does not settle to reference input. It is defined as steady state error. If the system has a steady state error, it can be removed easily with the help of an integral type of controller.

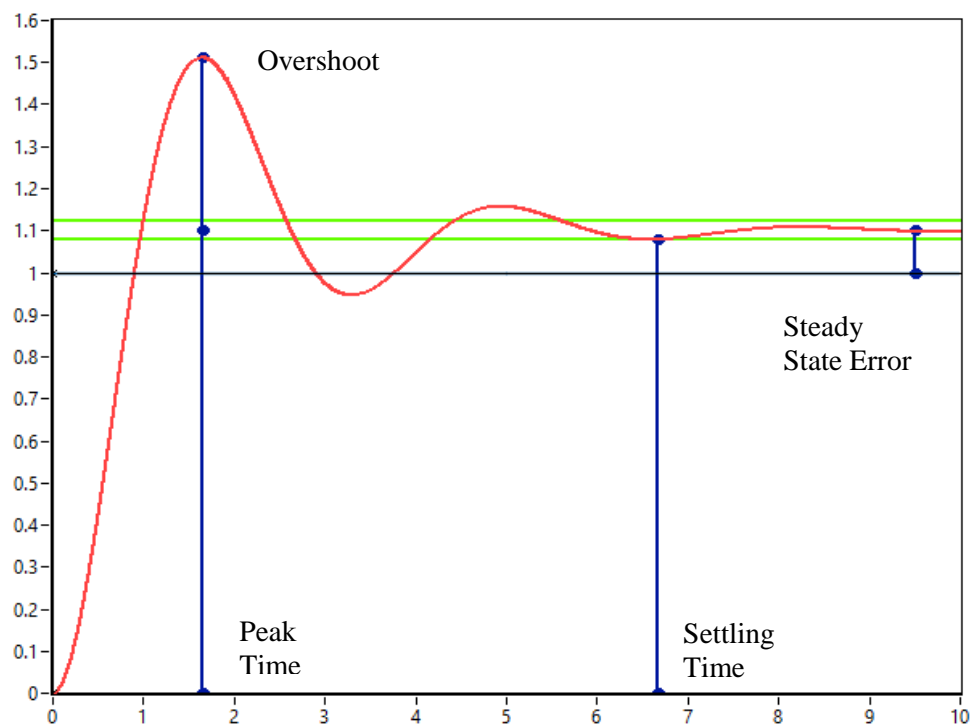


Figure 4.1: Overshoot, settling time, peak time and steady state error

Second order systems' general transfer function is below:

$$\frac{K \omega_n^2}{s^2 + 2\xi \omega_n s + \omega_n^2}$$

System response final value is the limit of the transfer function as “s” goes to “0”. With respect to the transfer function above, final value is “K” for unit step input.

4.1.7 In-Lab Exercise

1. Open “Helicopter_PerformanceMeasurements.slx” as follows.

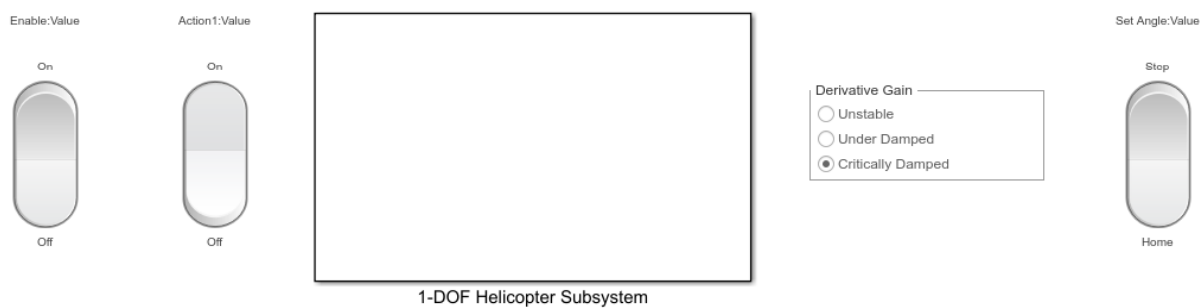


Figure 4.2: Block Diagram of Helicopter_PerformanceMeasurements.slx

2. Run the program. Switch on **Enable** and **Action** buttons
3. Select one option from **Unstable**, **Underdamped**, **Critically Damped** and click Step from Switch.
4. Calculate the following performance measures from graph: percentage overshoot, settling time, peak time and steady state error.
5. Observe the system input and output on the graph and determine the performance measures.
6. Click Home from switch and select a different option. Click Step then observe the changes in performance measures.

5 Control System Design

Open loop systems are called the manual control systems as the output has no effect upon the input. Although open loop structure is simple, economic and stable, it is inaccurate, unreliable and has no disturbance rejection. Closed loop systems are called automatic control systems and they have many advantages like disturbance rejection, noise attenuation, presence of nonlinearity and robustness, so closed loop system are preferred in control applications in general.

The purpose of the controller design with a closed loop system is the achievement of the previously specified performance measures. Choosing the type of controller depends on the system objectives and needs. In general, there are three tuning parameters; proportional (P), integral (I) and derivative (D). P-term depends on the present error, I-term accumulates the past errors and D-term predicts the future errors.

All linear controllers are based on combinations of these three terms on forward and feedback paths. There are many types of linear controllers such as P, PI, PD, PID, PI-PD, PV etc. PID controller is called three-mode controller and it is the most common control algorithm used in industry. In general, the derivative term adjusts the transient response, while the integral term eliminates the steady state error. There are also three major structures of PID algorithms: academic, parallel and serial form. Academic and serial forms of PID controller are described as interacting algorithms. On the other hand, parallel form is described a non-interacting algorithm. Academic and parallel forms of PID controllers will be used for Helicopter's position control. Also, other types of controller such as P, PI and PD are obtained by using academic and parallel forms of PID.

5.1 Academic PID controller

Academic PID controller is mathematically expressed as:

$$u(t) = K_c \left(e(t) + T_d \int e(t) dt + \frac{1}{T_i} \frac{de(t)}{dt} \right)$$

The proportional term is the gain of the controller so integral and derivative actions are dependent on value of K_c . K_c is real and has a finite value; T_d is derivative term time constant and T_i is the integral term time constant.

5.2 Parallel PID controller

Parallel PID controller is mathematically expressed as:

$$u(t) = K_p e(t) + K_d \int e(t) dt + K_i \frac{de(t)}{dt}$$

Proportional term is independent of integral and derivative terms. Three tuning parameters of parallel PID controller are determined separately. K_p , K_d and K_i are real and have finite values similar to the gain of academic PID controller. Academic PID controller form can be converted into the parallel form by using basic algebraic calculations. The inverse of this conversion is also possible.

5.3 Root Locus

Stability of the system can be determined directly from its closed loop poles. In order to achieve this W.R. Evans developed a graphical method called root locus to determine locations of all possible closed loop poles in s-domain with respect to gain (K).

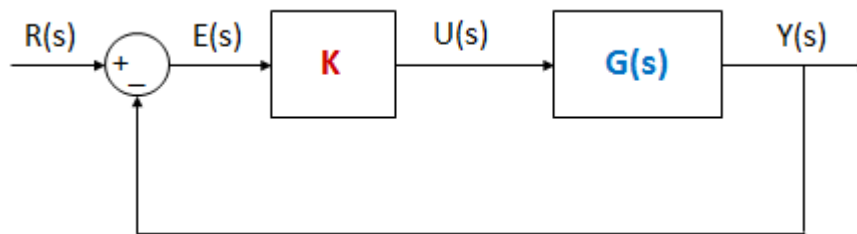


Figure 5.1: General Block Diagram of Root Locus

To draw a root locus, first the open loop transfer function should be obtained. The poles and the zeros of the open loop system are located on the complex plane. Secondly, the open loop system is multiplied by a proportional gain. Then the loop is closed with a unity feedback path. The denominator of the closed loop transfer function is described as the characteristic polynomial. Characteristic polynomial is a function of K and is shown below as:

$$C(s) = 1 + KG(s)$$

Finally, loci of the closed loop poles are calculated by solving the characteristic equation for each value of K. When all these poles are drawn on the complex plane, root locus is ready to

analyze. Root locus is also used for controller design, but our controller will not be designed using the root locus.

5.4 P Controller

P type controller is the simplest controller and can be easily implemented to a system. In addition, it does not change the order of the system. Closed loop system with P controller satisfies only one performance measure and only one closed loop pole is located as desired. Therefore, it is not sufficient to have more than one desired criteria. P controller does not eliminate steady state errors. With P controller, controller gain increases and the error is smaller but high increase in gain can cause a high increase in control signal, overshoot or even can cause instability of the system.

When the integral and derivative terms are zero, academic PID controller becomes P-type controller in a similar way. If the integral time constant is infinite and the derivative time constant is zero, parallel PID controller becomes P-type controller. General block diagram of the closed loop system with P type controller is below:

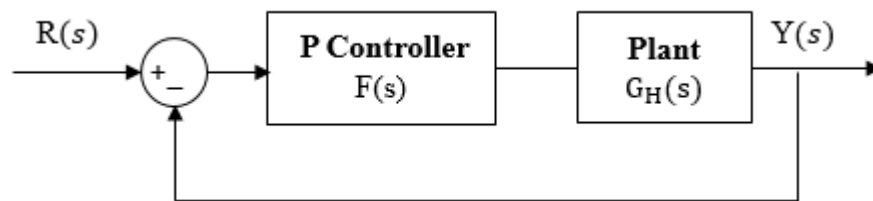


Figure 5.2: General Block Diagram with P Controller

For type P controller, our system is stable in very small range. Performance, which is expected from the system, could not be delivered with P controller. Because of these, P controller is not preferred for our system.

5.4.1 In-Lab Exercise

1. Open the "Helicopter_P_Design.slx",
2. You will see the following block diagram shown below:

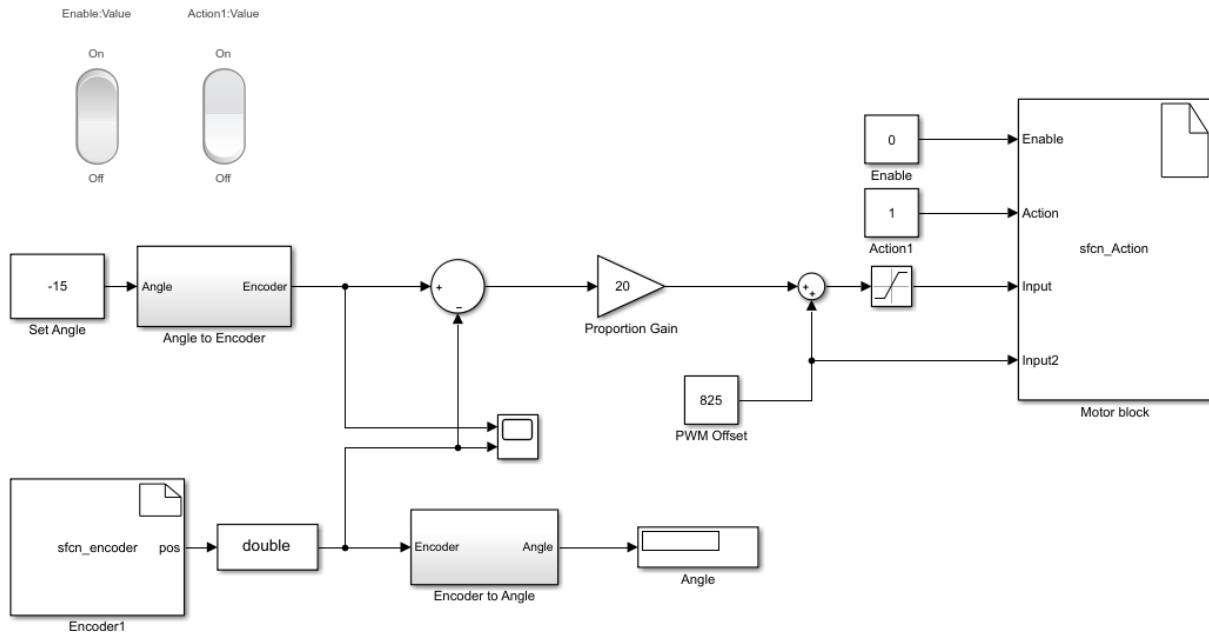


Figure 5.3: Block Diagram of "Helicopter_P_Design.slx"

3. Run the program. Switch on **Enable** and **Action** buttons
4. Double click to scope and open.
5. Observe real system responses.

5.5 PD Controller

PD controller is generally used for position control applications because of the plant's integrator. It approaches two of closed loop system poles to desired locations and adds a zero at the same time while it does not change the order of the system. PD controller adds one zero to the system. When the step input is applied at the beginning, derivative of the initial error approaches infinite in a split second. This pulse is described as a derivative kick. Design of the PD controller on the forward path results in a derivative kick. The other disadvantage of PD controller is that it cannot reject disturbance and eliminate the steady state error. Closed loop system block diagrams with PD controller can be seen as follows:

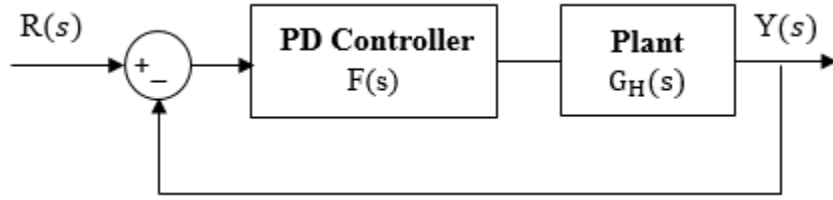


Figure 5.4: General Block Diagram of the Closed Loop System with PD Controller



Figure 5.5: Closed Loop System with PD Controller

Another version of PD controller is $K_p + K_d s$. In order to obtain this version, some translations should be applied.

5.5.1 Sample Design with PD controller

We will design a PD controller that satisfies 10% percentage overshoot and 3 seconds settling time with no steady state error. First, the closed loop transfer function and its characteristic polynomial is obtained and shown as below:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{F(s) * G(s)}{1 + F(s) * G(s)} = \frac{30 K_c (1 + T_d s)}{s^2 + (2.2 + 30 K_c T_d) s + 30 K_c}$$

$$P_C(s) = s^2 + (2.2 + 30 K_c T_d) s + 30 K_c$$

Secondly, ξ and ω_n are calculated from percentage overshoot (PO) and settling time (t_s):

$$\xi = \frac{-\ln(PO)}{\sqrt{\pi^2 + (\ln(PO))^2}} = \frac{-\ln(0.1)}{\sqrt{\pi^2 + (\ln(0.1))^2}} = 0.59$$

$$\omega_n = \frac{4}{\xi * t_s} = \frac{4}{0.59 * 3} = 2.25$$

Thirdly, the desired characteristic polynomial is determined:

$$P_D(s) = (s^2 + 2\xi\omega_n s + \omega_n^2) = (s^2 + 2.66 s + 5.08)$$

Finally, when desired and designed characteristic polynomials are equalized to each other, K_c and T_d are calculated.

$$K_c = 0.17 \quad T_d = 0.09$$

If some translations are applied, K_p and K_d can be find:

$$K_p = K_c$$

$$K_d = K_c * T_d$$

$$K_p = 0.17 \quad K_d = 0.016$$

As you can see on the graph, when the unit step input is applied to the system, percentage overshoot and settling time are almost same with our desired performance measures with a steady-state error that is about 0.7. PO equals to 16% and t_s is almost 3 seconds.

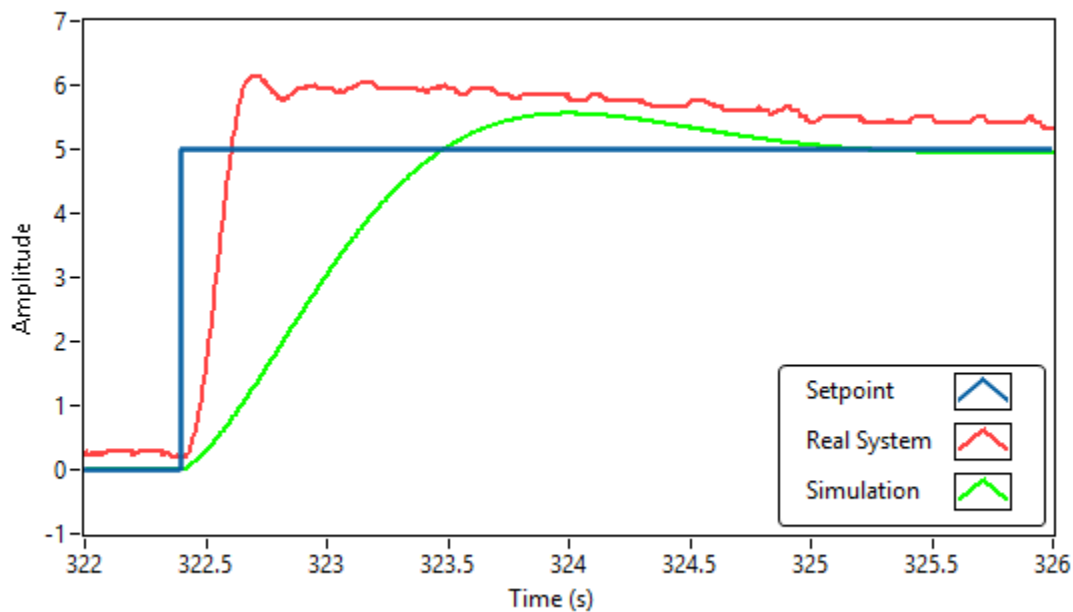


Figure 5.6: The Real-Time, Simulation and Step Input Graphs with PD Controller

5.5.2 In-Lab Exercises

1. Open the "Helicopter_PD_Design.slx",
2. You will see the following block diagram shown below:

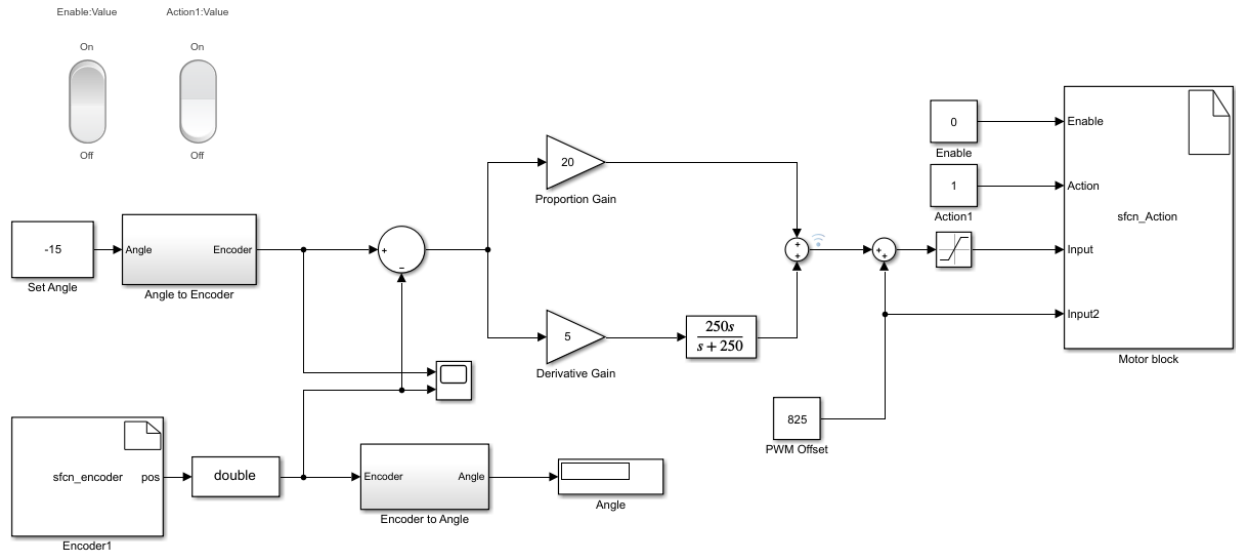


Figure 5.7: Block Diagram of "Helicopter_PD_Design.slx"

3. Run the program. Switch on **Enable** and **Action** buttons
4. Double click to scope and open.
5. Observe real system responses.

Answer the following questions for the block diagram as shown Figure 5.5.

1. Determine the order of the open loop transfer function.
2. Find the closed loop transfer function of the system with PD controller.
3. Obtain the characteristic polynomial of the closed loop system.
4. Determine the performance measures and calculate ξ and ω_n (Choose $10\% \leq OS \leq 20\%$ and $2 \leq t_s \leq 5$ seconds).
5. Obtain the desired characteristic polynomial with the help of Step 4 (Does residue polynomial become a necessity for PD controller?).
6. Equate the polynomials which are calculated at Step 3 and Step 5 and find the controller parameters.
7. What is the order of the system after controller? Does it change?

8. Draw the step response with designed PD controller via “Helicopter_PD_Design.slx”. Is the closed loop system stable?
9. What is the value of steady state error?
10. Before applying the designed controller parameters simulation and real time system, you have to enter an appropriate PWM Neutral value (between 775 and 825) and be sure that it is working. In order to be sure, you can regulate the PWM Neutral value during your first run. Then, apply the designed controller parameters on simulation and real time system via “Helicopter_PD_Design.slx”. Discuss the differences the response of simulation and real system via “Helicopter_PD_Design.slx”. Are they consistent?

5.6 PV Controller

PV controller is one of the most popular servo position controllers. Contrary to PD controller, PV controller does not add a zero and a pole to the system. It only changes the root locus of the system. In other words, poles, which belong to the closed loop system, are moved to the more stable region on the root locus by PV controller. Moreover, PV controller prevents derivative kicks because the derivative term is on the feedback path. Most of the time, P type controller is not sufficient; therefore, root locus is easily changed with a derivative term. Block diagrams are shown below:

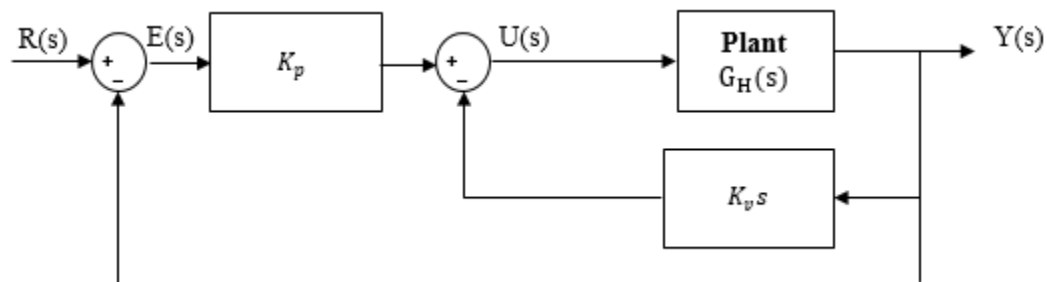


Figure 5.8: General Block Diagram of Closed Loop System with PV Controller

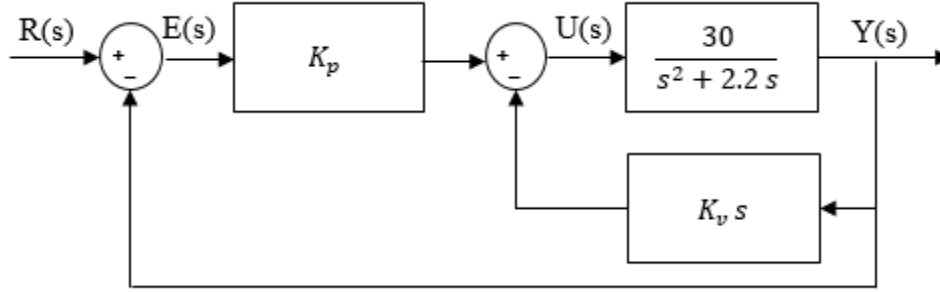


Figure 5.9: Closed Loop System with PV Controller

5.6.1 Sample Design with PV controller

We will design a PV controller with respect to 5% percentage overshoot, 2 seconds settling time and no steady state error. Gain part of the PV controller on the forward path represents $F_1(s)$ and derivative part of the PV controller on feedback path represents $F_2(s)$. The inner loop transfer function is obtained and is shown below:

$$T_{inner}(s) = \frac{G_H(s)}{1 + F_2(s) * G_H(s)} = \frac{30}{s^2 + (2.2 + 30)K_v s + 1}$$

Now, we obtain the outer loop transfer function using the inner loop transfer function.

$$T_{outer}(s) = \frac{F_1(s) * T_{inner}(s)}{1 + F_1(s) * T_{inner}(s)} = \frac{30K_p}{s^2 + (30K_v + 2.2)s + 30K_p}$$

PV controller is practically a PD controller because the characteristic polynomials of PV and PD controllers are the same.

$$P_C(s) = s^2 + (30K_v + 2.2)s + 30K_p$$

First of all, ξ and ω_n are calculated.

$$\xi = \frac{-\ln(PO)}{\sqrt{\pi^2 + (\ln(PO))^2}} = \frac{-\ln(0.05)}{\sqrt{\pi^2 + (\ln(0.05))^2}} = 0.69$$

$$\omega_n = \frac{4}{\xi * t_s} = \frac{4}{0.69 * 2} = 2.89$$

Secondly, second order desired characteristic polynomial is obtained.

$$P_D(s) = (s^2 + 2\xi\omega_n s + \omega_n^2) = (s^2 + 4s + 8.39)$$

Finally, the unknown parameters, a , K_p and K_v , are found.

$$K_p = 0.27 \quad K_v = 0.06$$

Closed loop system has a steady state error because of the modeling error. The steady stated error can be calculated as 0.15. The settling time of the closed loop system is 2.06 seconds and overshoot is 5%.

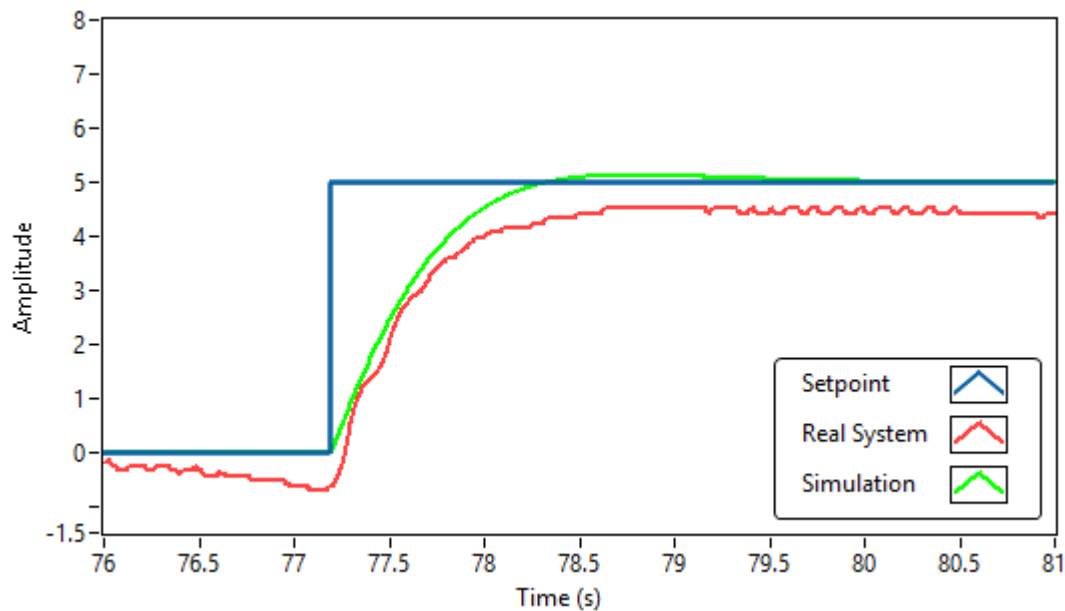


Figure 5.10: The Real-Time, Simulation and Step Input Graphs with PV Controller

5.6.2 In-Lab Exercises

1. Open the "Helicopter_PV_Design.slx",
2. You will see the following block diagram shown below:

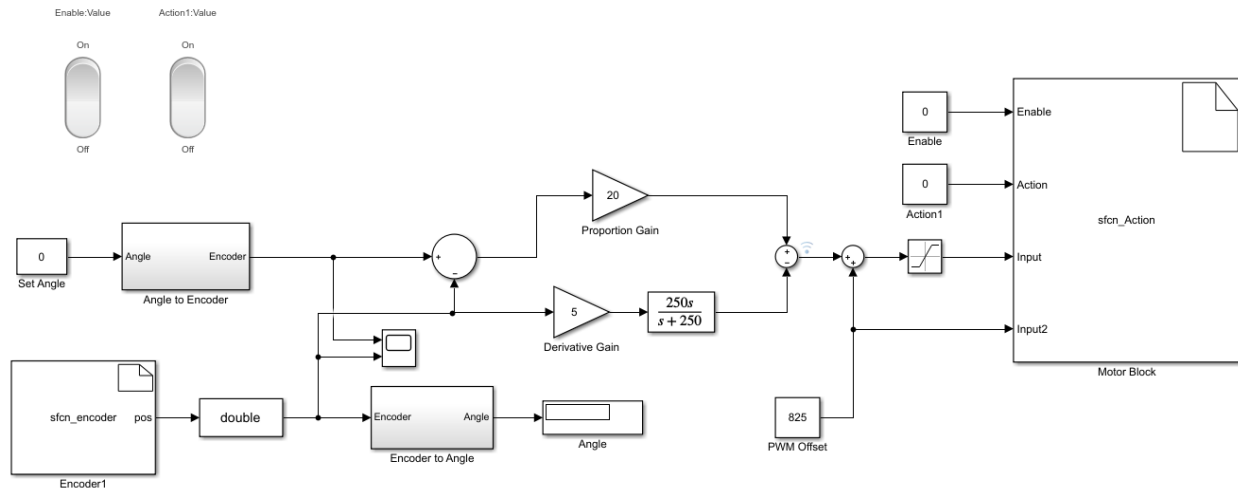


Figure 5.11: Block Diagram of "Helicopter_PV_Design.slx"

3. Run the program. Switch on **Enable** and **Action** buttons
4. Double click to scope and open.
5. Observe real system responses.

Answer the following questions for the block diagram as shown Figure 5.9.

1. Determine the order of the open loop transfer function.
2. Find the closed loop transfer function of the system with PV controller.
3. Obtain the characteristic polynomial of the closed loop system.
11. Determine the performance measures and calculate ξ and ω_n (Choose $10\% \leq OS \leq 20\%$ and $2 \leq t_s \leq 5$ seconds).
4. Obtain the desired characteristic polynomial with the help of Step 4 (Does residue polynomial become necessity for PV controller?),
5. Equate the polynomials which are calculated at Step 3 and Step 5 and find the controller parameters.
6. What is the order of the system after adding the controller? Did it change?

7. Draw the step response with designed PV controller via “Helicopter_PV_Design.slx”. Is the closed loop system stable?
8. With respect to PD controller, what does it change?
9. What is the value of steady state error?
10. Before applying the designed controller parameters on simulation and real time system, you have to enter an appropriate PWM Neutral value (between 775 and 825) and be sure that it is working. In order to be sure, you can regulate the PWM Neutral value during your first run. Then, apply the designed controller parameters on simulation and real time system via “Helicopter_PV_Design.slx”. Discuss the differences the response of simulation and the real system. Are they consistent?

5.7 PID Controller

PID controller is the most popular controller even when using in non-linear applications. The types of non-linear controllers are complicated and are hardly implemented, so PID controller is preferred even if the system is linear. The controller adds two zeros and one pole to the system. Mainly, the controller places the closed loop system poles to a desired location. Nevertheless, like PI controller, PID controller does not usually return accurate result for position controller. The characteristic polynomial poles are stable in a small region within plant's integrator poles. Block diagram with PID controller is shown below:

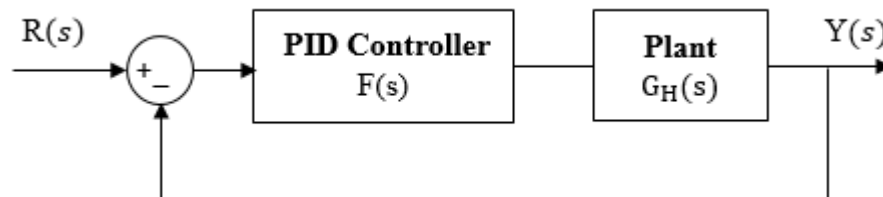


Figure 5.12: Closed Loop System with PID Controller

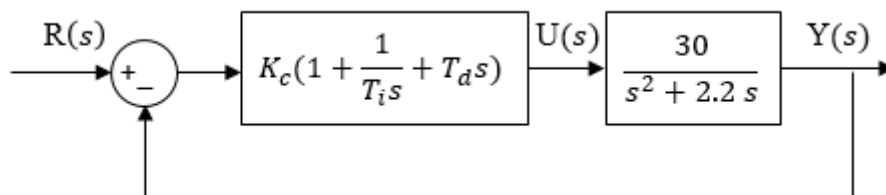


Figure 5.13: Closed Loop System with PID Controller

When parallel PID parameters are increased independently, Table 5.1 shows the qualification results of the performance measures.

PID controller can be written as $K_p + \frac{K_i}{s} + K_d s$ which is academic version. In order to obtain this version, some translations should be applied.

Table 5.1: Qualification Results Corresponding Increase in PID Parameters

PID Parameters	Peak Time	Overshoot	Settling Time	Steady State Error
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	No effect

5.7.1 Sample Design with PID controller

A PID controller will be designed that satisfies 5% percentage overshoot and 3 seconds settling time with no steady state error. Thanks to integrator, there is no steady state error. Firstly, closed loop transfer function and its characteristic polynomial are obtained in below:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{F(s) * G(s)}{1 + F(s) * G(s)} = \frac{30K_c(T_i T_d s^2 + T_i s + 1)}{T_i s^3 + (2.2T_i + 30K_c T_i T_d)s^2 + 30K_c T_i s + 30K_c}$$

$$P_c(s) = T_i s^3 + (2.2T_i + 30K_c T_i T_d)s^2 + 30K_c T_i s + 30K_c$$

Secondly, ξ and ω_n are calculated from percentage overshoot (PO) and settling time (t_s):

$$\xi = \frac{-\ln(PO)}{\sqrt{\pi^2 + (\ln(PO))^2}} = \frac{-\ln(0.05)}{\sqrt{\pi^2 + (\ln(0.05))^2}} = 0.69$$

$$\omega_n = \frac{4}{\xi * t_s} = \frac{4}{0.69 * 3} = 1.93$$

We find the controller parameters with equal coefficients of the designed and desired characteristic polynomials, so the order of both of them must be equal. If the order of the designed characteristic polynomial is higher than two, we add a multiplication polynomial described as the residue polynomial. For the design of the PID controller, designed characteristic polynomial is

third-order but we obtain a second-order desired characteristic polynomial from the performance measures. Therefore, the desired characteristic polynomial must contain a first order residue polynomial. Residue polynomial is chosen as $(as + b)$, desired characteristic polynomial is determined:

$$P_D(s) = (as + b) * (s^2 + 2\xi\omega_n s + \omega_n^2) = (as + b) * (s^2 + 2.66s + 3.73)$$

Finally, when desired and designed characteristic polynomials are equalized to each other a, b, K_c, T_i and T_d are calculated:

$$a = 1.23 \quad b = 2.38 \quad K_c = 0.29 \quad T_d = 0.026 \quad T_i = 1.23$$

If some translations are applied, K_p, K_i , and K_d can be find:

$$K_p = K_c$$

$$K_i = \frac{K_c}{T_i}$$

$$K_d = K_c * T_d$$

$$K_p = 0.29 \quad K_d = 0.079 \quad K_i = 0.24$$

Controller parameters are written for the closed loop system; simulation step response and real system response are shown in Figure 5.14.

Overshoot is 22% and settling time is 4.5 seconds. Closed loop system is almost satisfied our performance measures. It has no steady state error for step input with the PID controller.

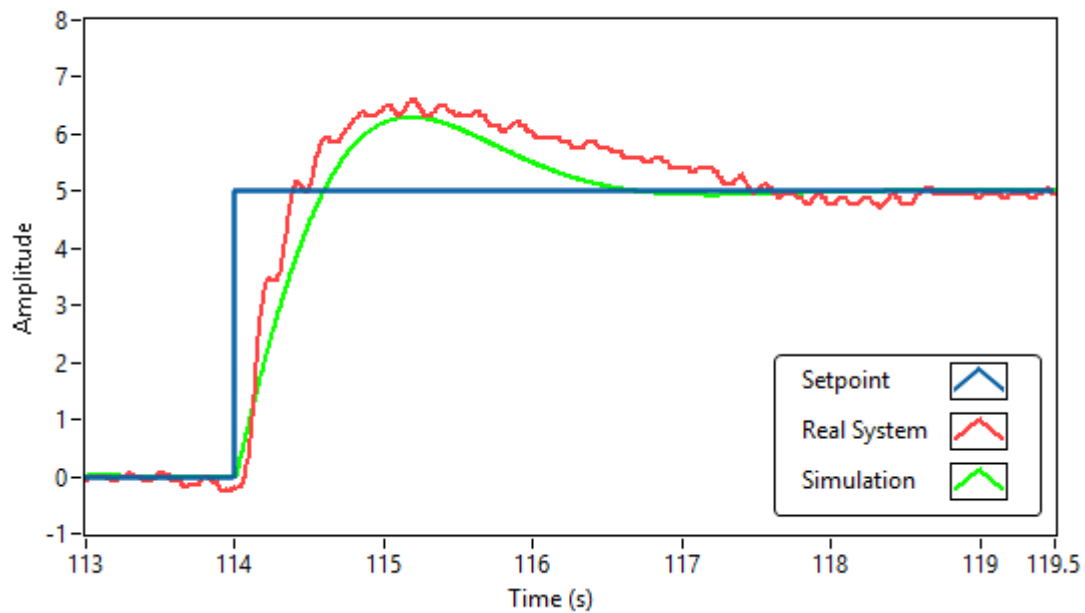


Figure 5.14: The Real-Time, Simulation and Step Input Graphs with PID Controller

5.7.2 In-Lab Exercises

1. Open the “Helicopter_PID_Design.slx”,
2. You will see the following block diagram shown below:

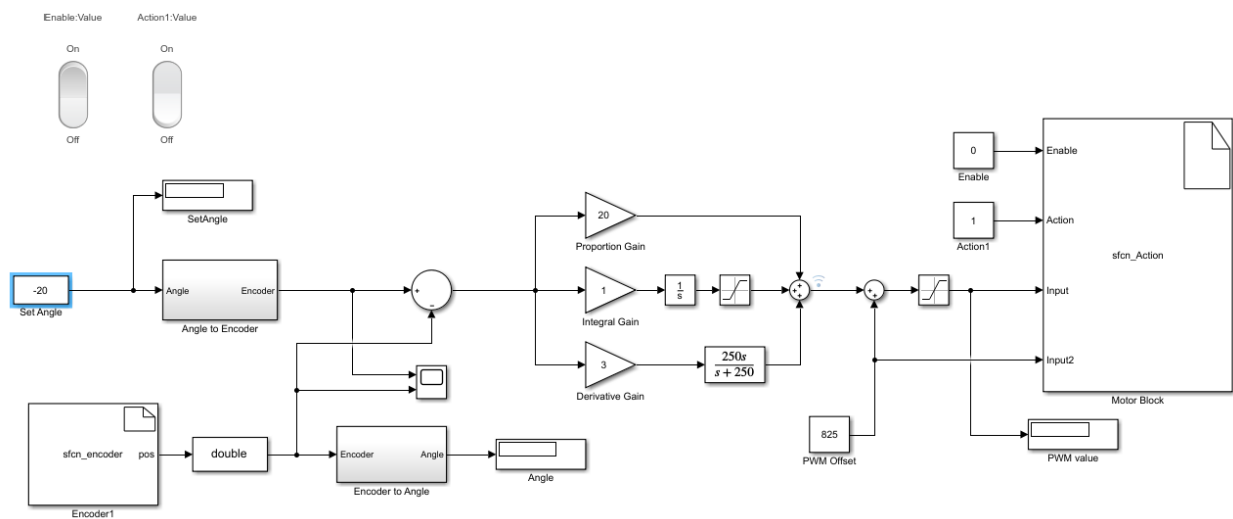


Figure 5.15: Block Diagram of “Helicopter_PID_Design.slx”

3. Run the program. Switch on **Enable** and **Action** buttons
4. Double click to scope and open.

5. Observe real system responses.

Answer the following questions for the block diagram as shown Figure 5.13.

1. Determine the order of the open loop transfer function.
2. Find the closed loop transfer function of the system with PID controller.
3. Obtain the characteristic polynomial of the closed loop system.
12. Determine the performance measures and calculate ξ and ω_n (Choose $10\% \leq OS \leq 15\%$ and $2 \leq t_s \leq 5$ seconds).
4. Determine the residue polynomial and obtain the desired characteristic polynomial with the help of Step 4.
5. Equate the polynomials which are calculated at Step 3 and Step 5 and find controller parameters.
6. What is the order of the system after controller? Does it change?
7. Draw the step response with designed PID controller via "Helicopter_PID_Design.slx". Is the closed loop system stable?
8. What is the value of steady state error?
9. Before applying the designed controller parameters simulation and real time system, you have to enter an appropriate PWM Neutral value (between 775 and 825) and be sure that it is working. In order to be sure, you can regulate the PWM Neutral value during your first run. Then, apply the designed controller parameters on simulation and real time system via "Helicopter_PID_Design.slx". Discuss the differences the response of the simulation and the real system. Are they consistent?