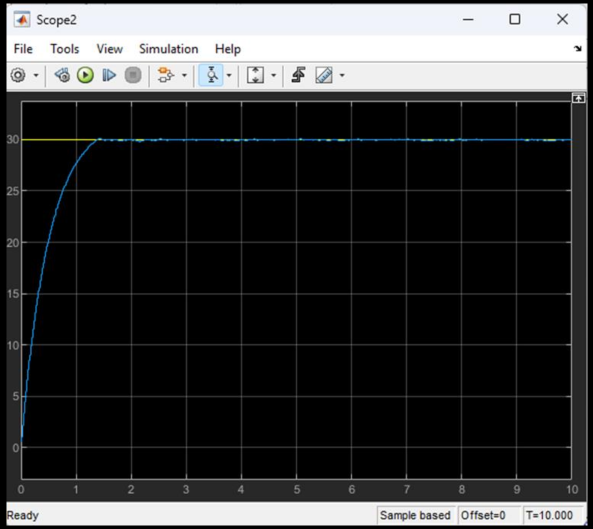
Contents

DC Motor Proportional and Integral (PI) Speed Control



Key Learning Points

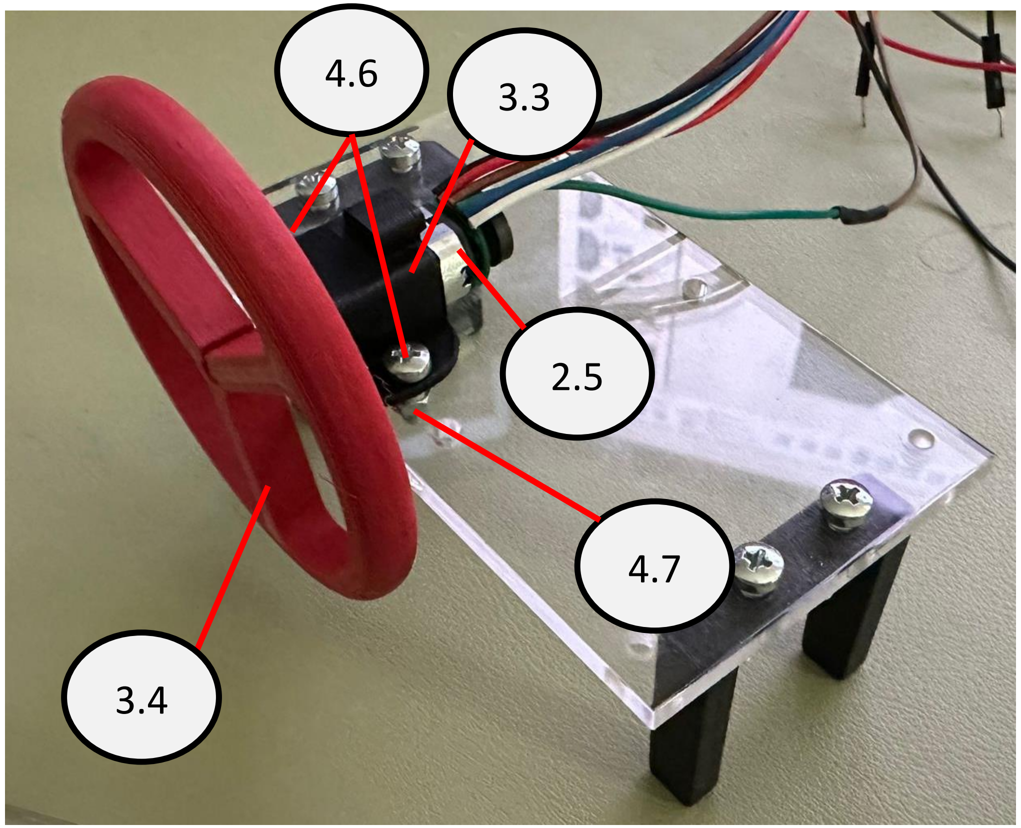
After this Lecture, you will be able to understand the following:

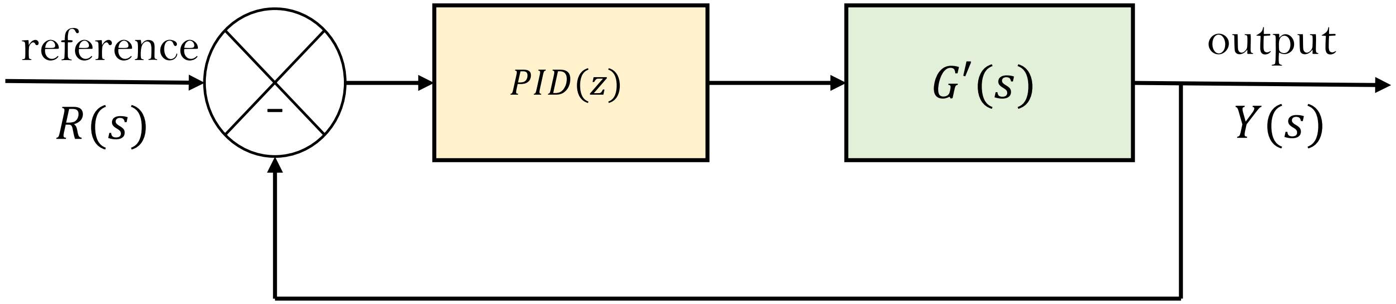
1. Use of least squares to develop a system model that includes the DC motor dynamics, load from a wheel and the signal processing
2. Use of Root Locus within MATLAB to design a PI controller
3. Implementation of a PI controller on hardware that meets a set of requirements, i.e., transient and steady state performance and disturbance rejection

4.1 Introduction

- To start, a system model is going to be developed that includes the DC motor dynamics, a wheel and a low pass filter, denoted $G'(s)$
- Set-up the CLB rig as illustrated to the right, using the following components:
    - 2.5: Brushed geared DC motor with encoder
    - 3.3: DC motor mount
    - 3.4 Wheel
    - 4.6: 2 x bolt, 16mm, M3
    - 4.7: 2 x nuts, M3

- The developed transfer function, denoted $G'(s)$ will therefore capture the DC motor dynamics including the wheel and low pass filter
- The transfer function $G'(s)$ will then be used to design the PID controller

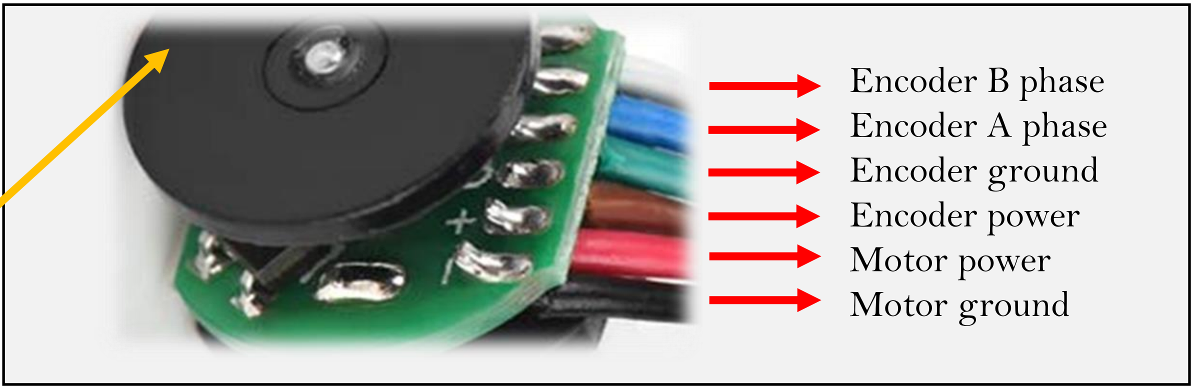




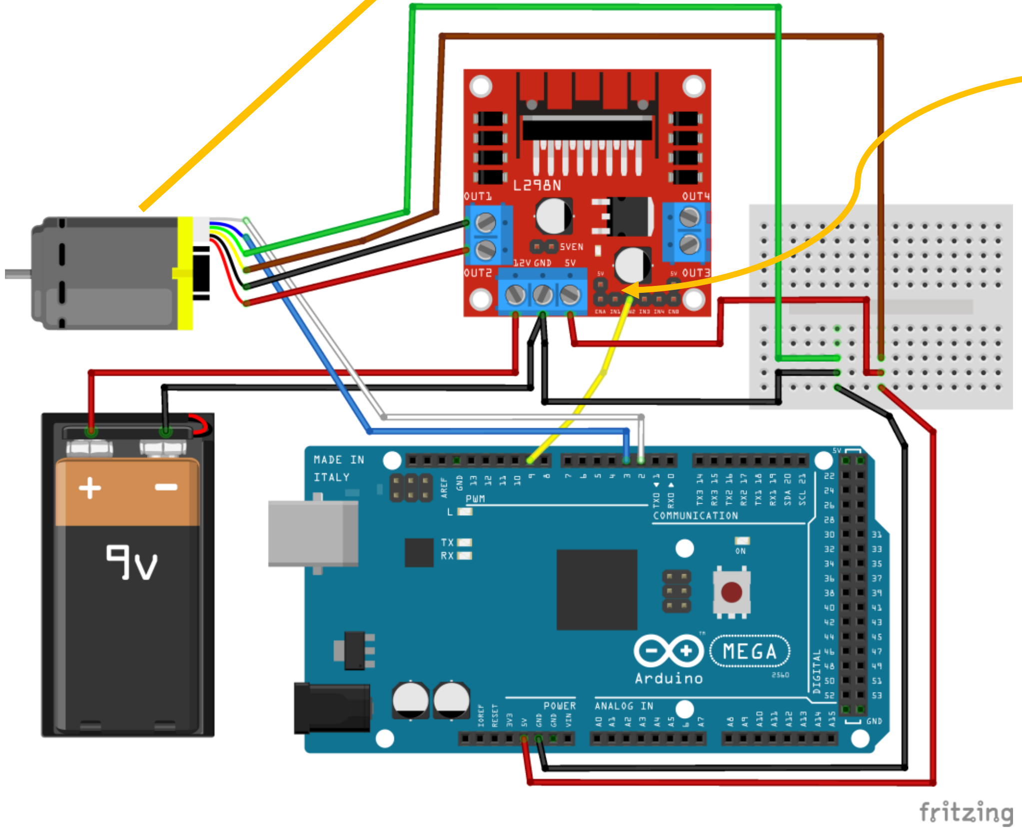
4.2 Hardware

- Same hardware used is in the system identification exercise
- The task involves connecting a H-bridge and DC motor with an encoder to an Arduino
- Required hardware for the exercise:
  i. Supported Arduino Mega 2560 board
  ii. USB cable
  iii. H-bridge
  iv. 6V DC motor with encoder
  v. 9V battery
  vi. 9V power jack
  vii. Various wires

Circuit diagram given here varies ever so slightly to the motor we are using

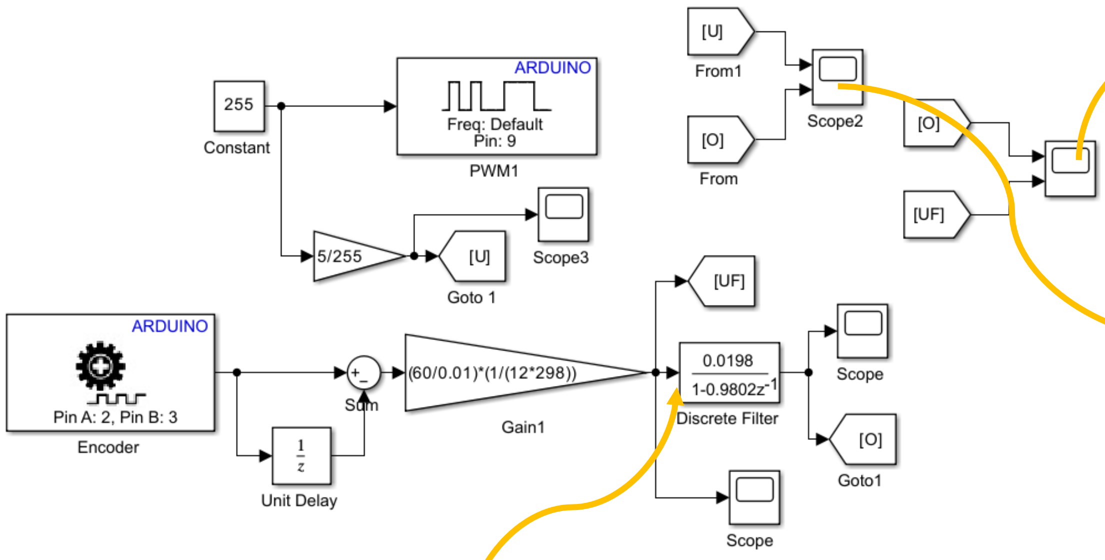


Note that 'In1' appears in a different position on the H-bridge we are using in class

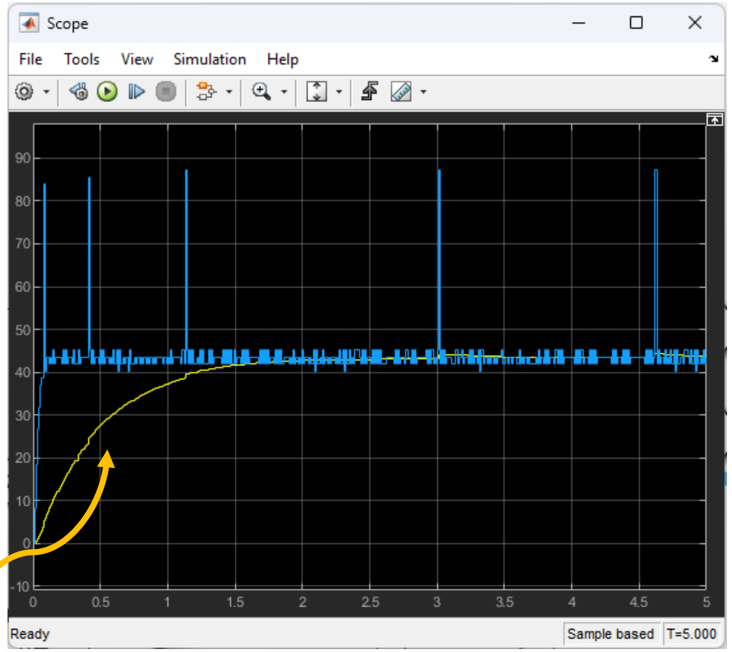


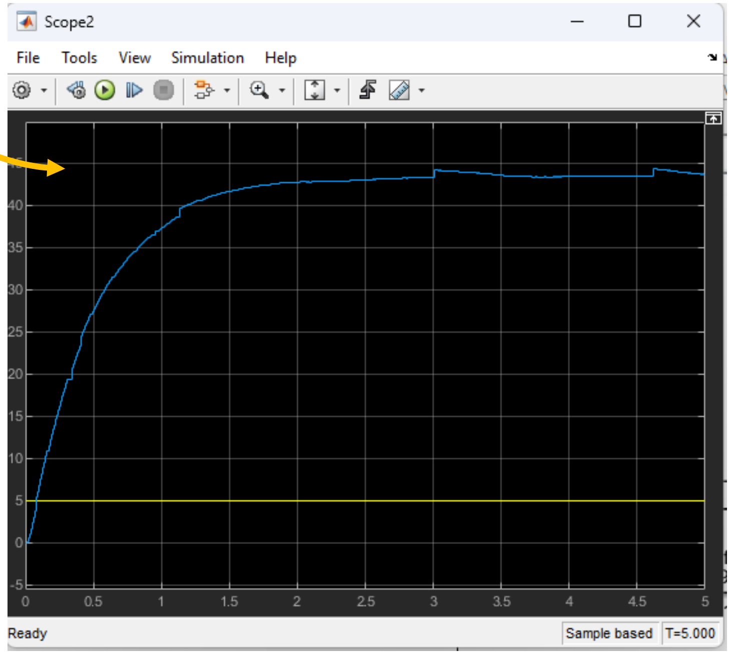
4.3 Algorithm Design

- To recall how to capture the data (etc.), see the ‘Least Squares for Parameter Estimation of a DC Motor’ notes
- Low pass filter is designed using a time constant, τ of 0.5



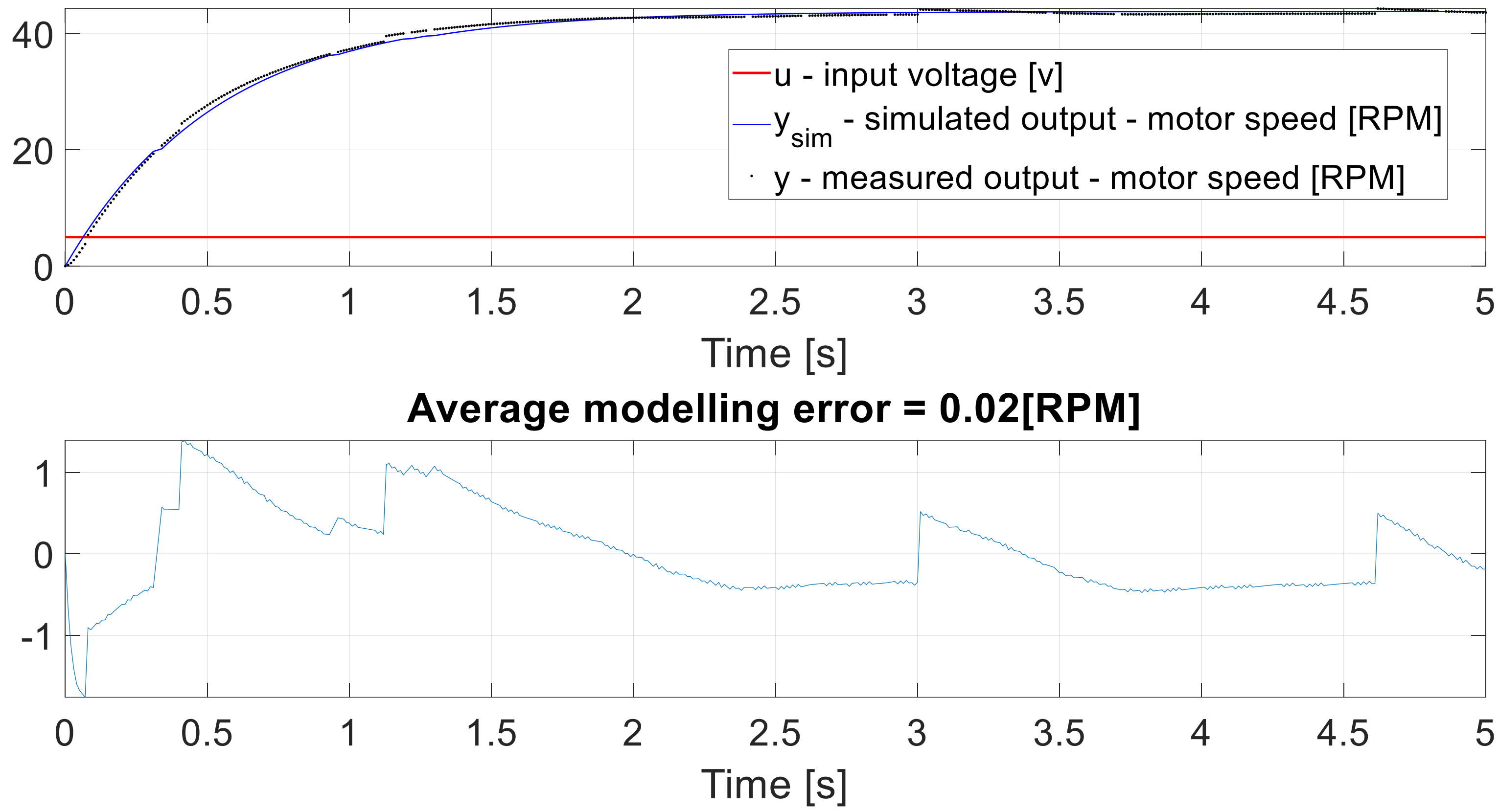
Low pass filter introduced





4.3 Algorithm Design

- Use the MATLAB script from the 'Least Squares for Parameter Estimation of a DC Motor' notes
- The graphical outputs are shown
- From visually inspection, it is clear the average modelling error is very small



4.4 Transfer Function

- The time constant, τ, and system gain , K, for the 1st order transfer function can be captured using MATLAB, i.e.,

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1}$$

- Using the MATLAB script, the following is determined:

$$G'(s) = \frac{\dot{\theta}(s)}{V(s)} = \frac{8.77}{0.59s + 1} \left[\frac{RPM}{V}\right]$$

The form here is given by:

$$\frac{b_0}{z - a_1}$$

```
%% Transfer function, time constant and system gain
a1 = theta(1);
b0 = theta(2)
Gd = tf([b0],[1 a1],0.01) %discrete-time transfer function
G = d2c(Gd)

b = G.Denominator{1,1};
b = b(1,2);
Tau = 1/b % 1 time constant (63.2% of yss)
FiveTau = 5*Tau % 5 time constants (99% of yss)
a = G.Numerator{1,1};
K = a(1,2)/b %system gain
```

Interesting: significant change to the time constant of the transfer function, i.e.,

$$G(s) = \frac{7.13}{0.029s + 1}$$

```
theta =
  -0.9809
   0.1676
>> untitled6
b0 =
   0.1676
Gd =
   0.1676
 ----------
 z - 0.9809
Sample time: 0.01 seconds
Discrete-time transfer function.
Model Properties
G =
  16.92
 --------
 s + 1.93
Continuous-time transfer function.
Model Properties
Tau =
   0.5182
FiveTau =
   2.5911
K =
   8.7669
```

4.5 PI Controller

- Transient requirements of the control system:
  i. Peak time less than 2 seconds
  ii. No overshoot
- Steady state requirements:
  i. Zero steady state error (SSE)
  ii. Disturbance rejection

- Following important properties from the system response:
  i. Type 0 system, i.e., system will contain steady state error (SSE)
  ii. No oscillation on the system response, therefore derivative (D) control gain is not required

- Recall the PI controller being given by:

$$C(s) = \frac{K_p s + K_i}{s} \tag{5-1}$$

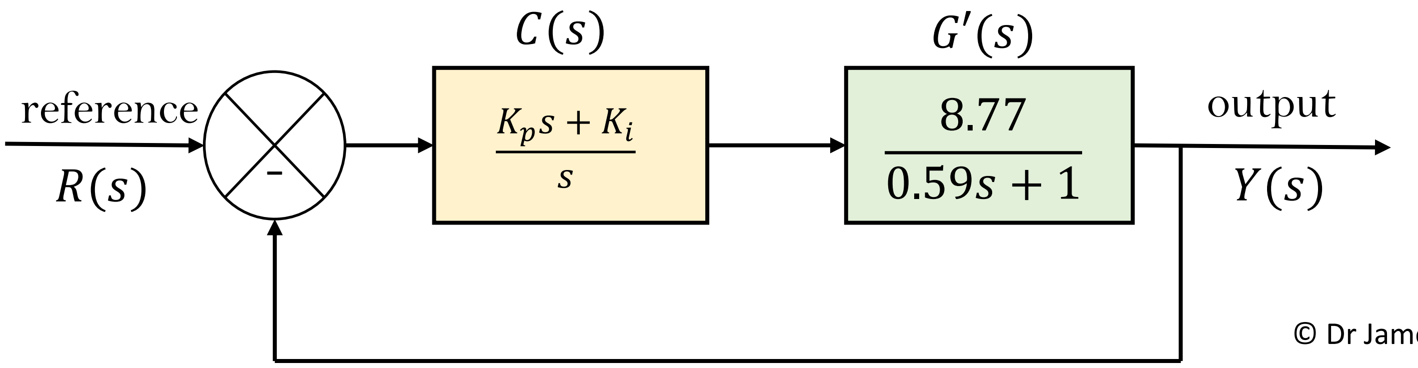
- Considering the closed-loop control system transfer function:

$$G_{CL}(s) = \frac{Y(s)}{R(s)} = \frac{C(s)G'(s)}{1 + C(s)G'(s)} \tag{5-2}$$

- Now consider the DC motor model, i.e., $G'(s) = \frac{8.77}{0.59s+1}$, the following is determined:

$$G_{CL}(s) = \frac{Y(s)}{R(s)} = \frac{\frac{(K_p s + K_i)8.77}{s(0.59s+1)}}{1 + \frac{(K_p s + K_i)8.77}{s(0.59s+1)}} = \frac{(K_p s + K_i)8.77}{0.59s^2 + s + (K_p s + K_i)8.77} \tag{5-3}$$

- Various approaches can be used to initially select the control gains of K_p and K_i to place the closed-loop poles in desired locations, e.g., algebraically and root locus



4.5 PI Controller

- Root locus is used to initially select the control gains of K_p and K_i
- Recall that Root Locus involves using the open-loop transfer function (i.e., in our case $G'(s)$) and assumes negative unity feedback with a proportional control gain, K_p
- Root locus is used where the controller in the feedforward path is 'lumped' with $G'(s)$

- Recall the PI controller being given by:

$$C(s) = \left(K_p + \frac{K_i}{s}\right)E(s) \tag{5-4}$$

- This is now given by the following form:

$$C(s) = K_p\left(1 + \frac{\widetilde{K}_i}{s}\right)E(s) \tag{5-5}$$

where $\widetilde{K}_i = \frac{K_i}{K_p}$

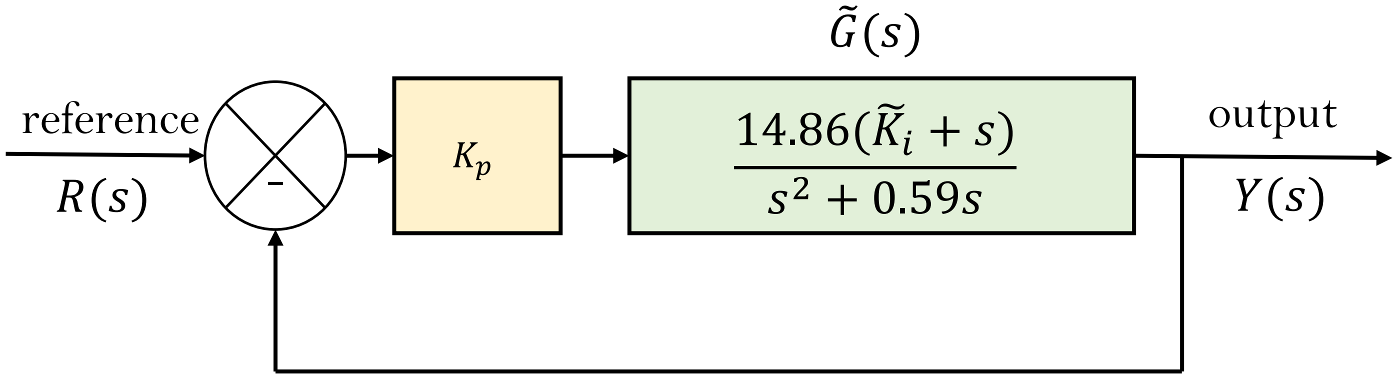
- The feed forward transfer function $C(s)G'(s)$ is given by:

$$K_p\tilde{C}(s)G'(s) \tag{5-6}$$

where $\tilde{C}(s) = \left(1 + \frac{\widetilde{K}_i}{s}\right)$

- This resulting in the following:

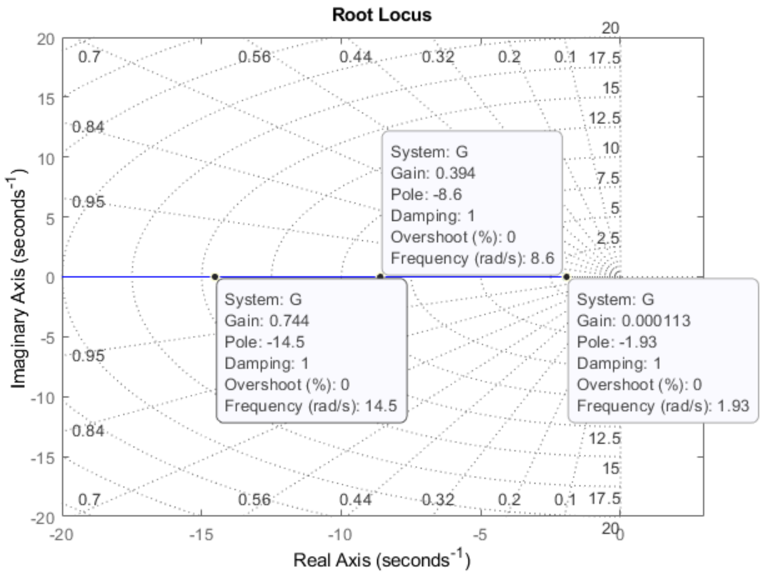
$$\tilde{G}(s) = \tilde{C}(s)G'(s) \tag{5-7}$$



4.5 PI Controller

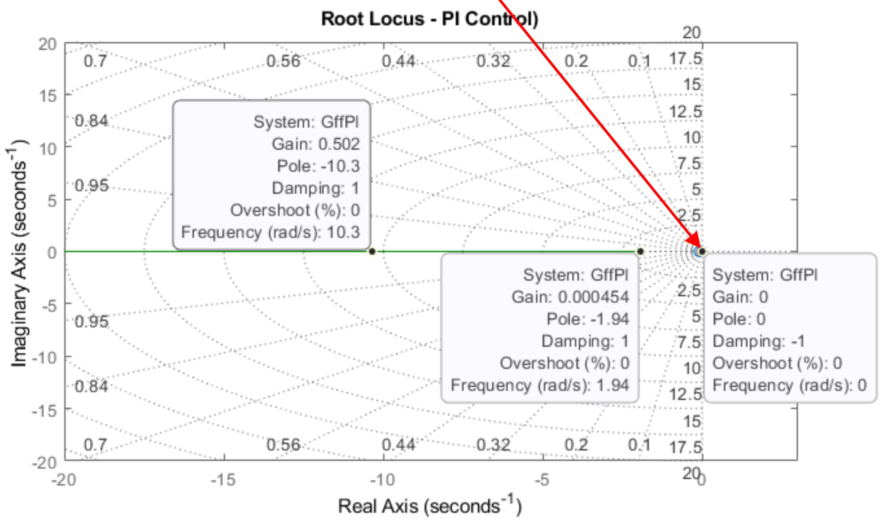
- Using MATLAB, Root locus is used here to investigate the proportional control gain, K_p (Top-Right) and the integral control gain, K_i (Bottom-Left)
- Using Root Locus, the following PI control gains are selected:
  - *i.* $K_p = 0.5$
  - *ii.* $K_i = 0.05$

```
clear; close all; clc;
%% System model (developed using least squares)
s = tf('s');
G = 16.92/(s+1.93);
%% Root Locus
rlocus(G);
grid on;
axis([-20 3 -20 20])
```




```
clear; close all; clc;
%% System model (developed using least squares)
s = tf('s');
G = 16.92/(s+1.93);
%% Control gains
Kp = 0.5;
Ki = 0.05;
%% Closed-loop control system
Kin = Ki/Kp;
GffPI = (1+Kin/s)*G;
%% Root Locus
rlocus(GffPI);
axis([-20 3 -20 20])
grid on;
title ('Root Locus - PI Control)')
```

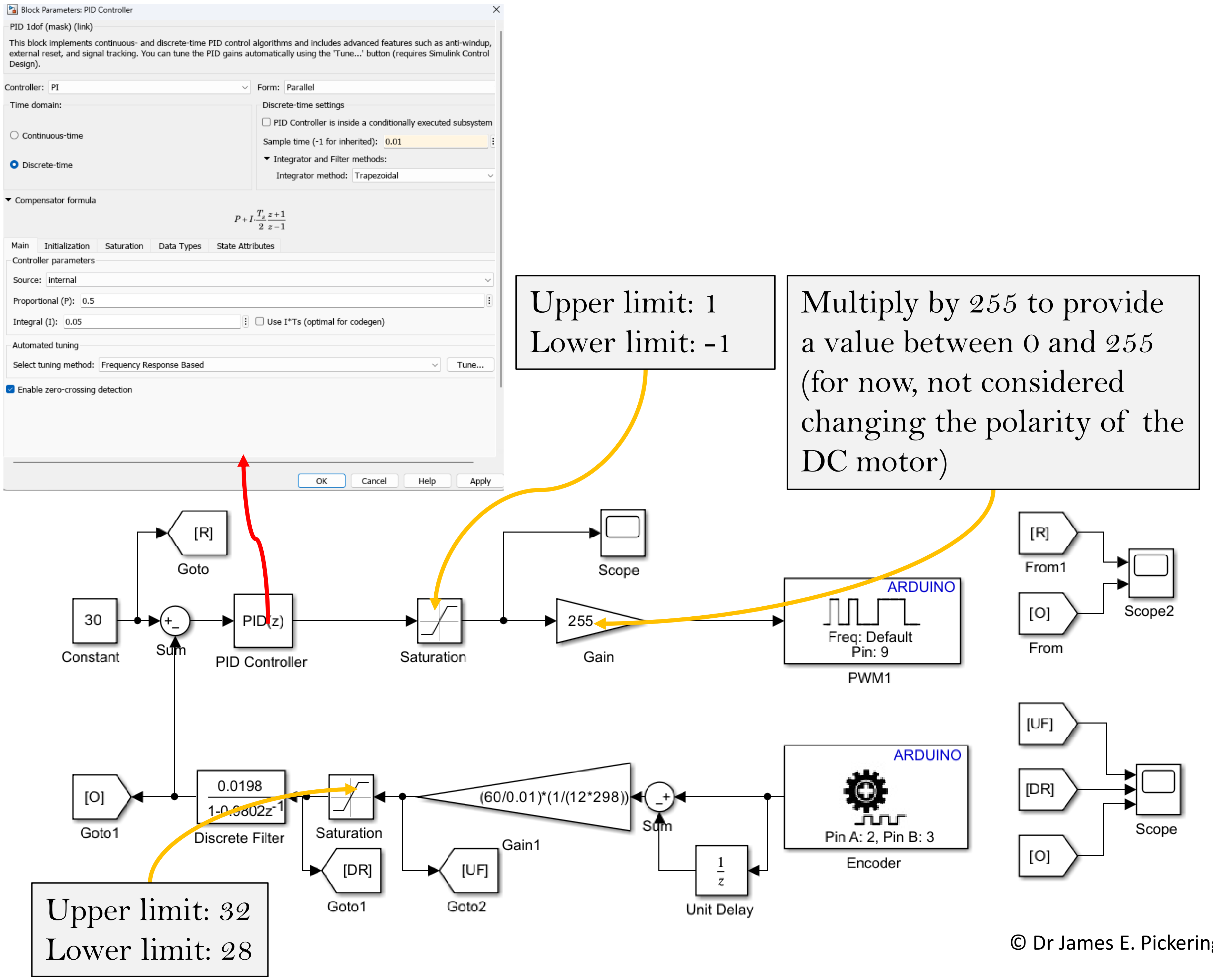
Zero is located at the origin of the s-plane, with this 'capturing' the pole



4.5 PI Control Algorithm Design

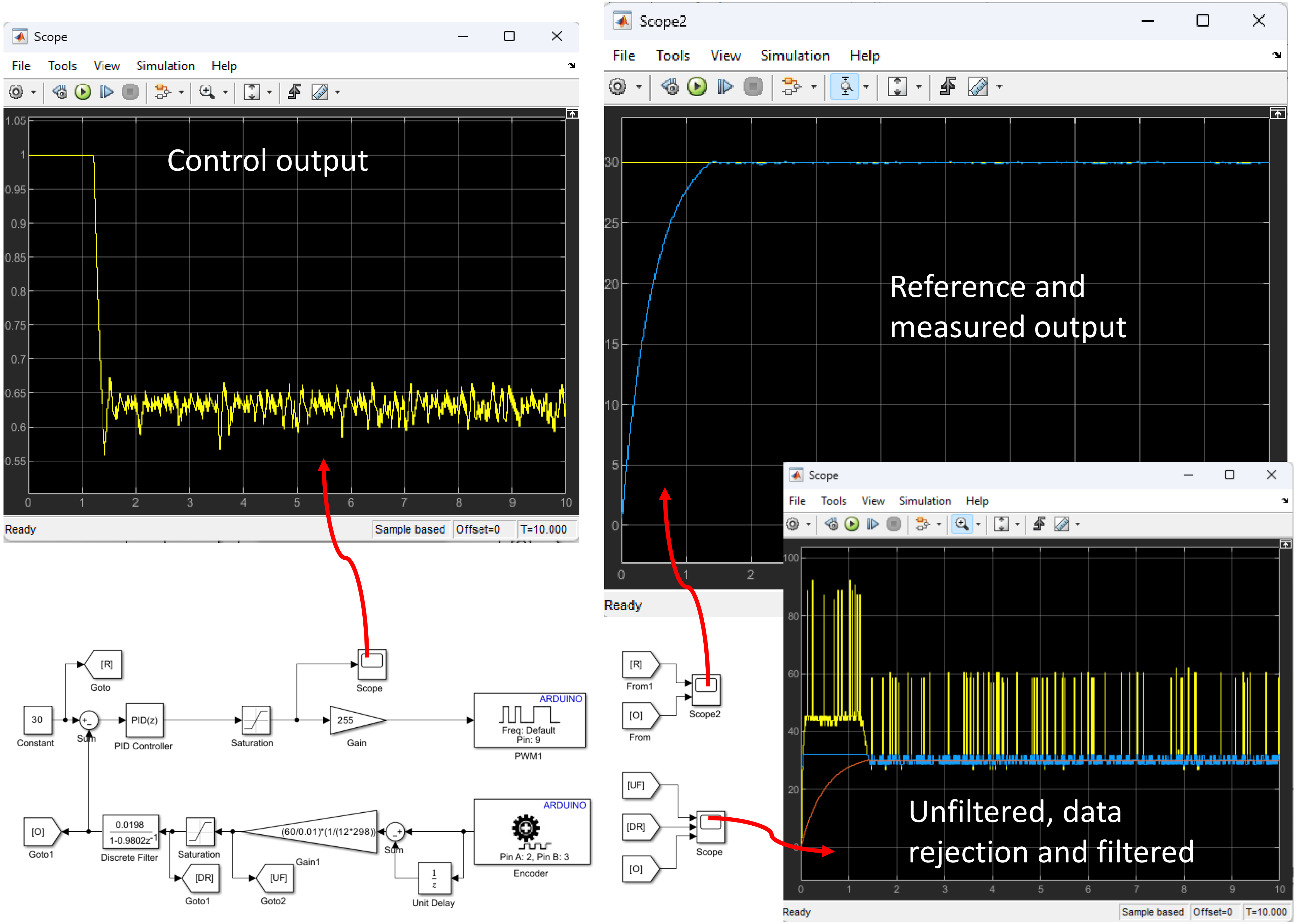
- A discrete-time PI controller is configured, with the following properties:
  i. Integrator method: Trapezoidal
  ii. Sample time (interval) of 0.01 seconds

- The following PI control gains are used:
  i. $K_p = 0.5$
  ii. $K_i = 0.05$



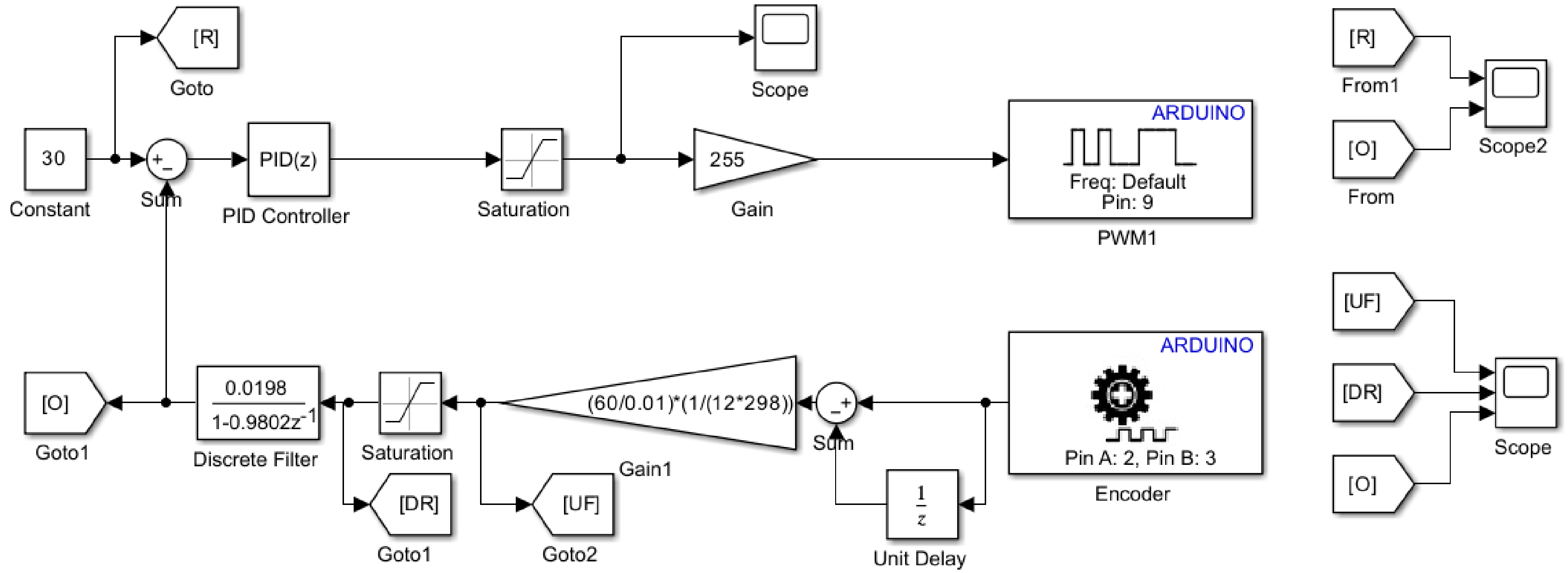
4.5 PI Control Algorithm Design

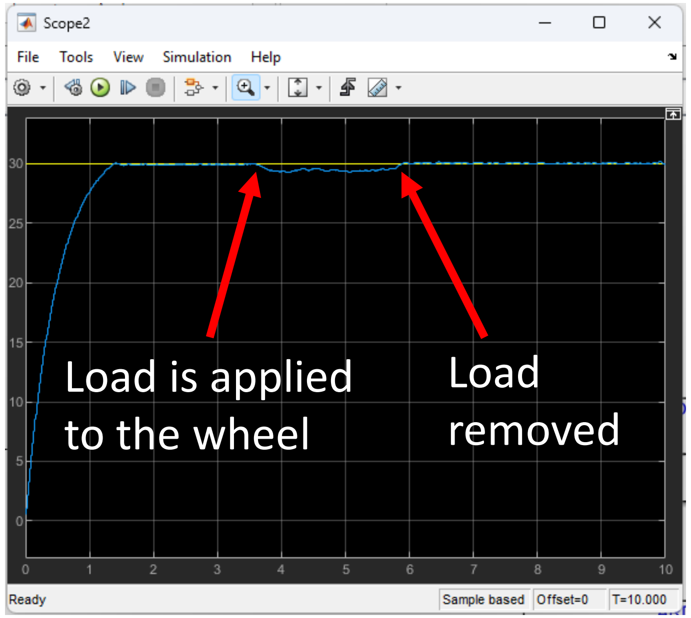
- Referring to the requirements:
  i. Peak time less than 2 seconds -> ACHIEVED
  ii. No overshoot -> ACHIEVED
- Steady state requirements:
  i. Zero steady state error (SSE) -> ACHIEVED
  ii. Disturbance rejection



4.6 Disturbance Rejection

- Referring to the requirements:
  i. Peak time less than 2 seconds -> ACHIEVED
  ii. No overshoot -> ACHIEVED
- Steady state requirements:
  i. Zero steady state error (SSE) -> ACHIEVED
  ii. Disturbance rejection -> ACHIEVED

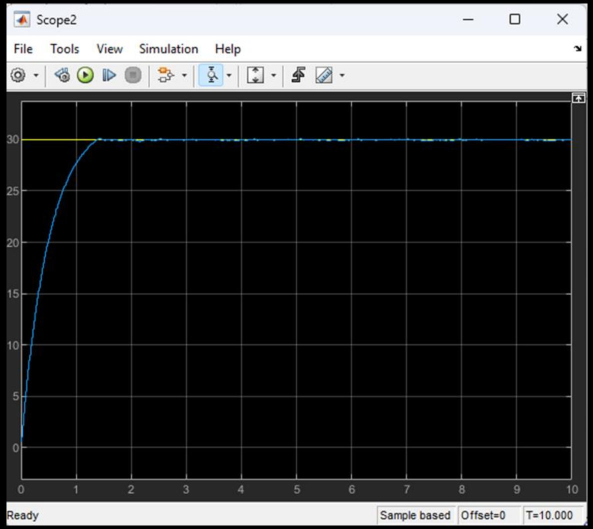




Contents

DC Motor Proportional and Integral (PI) Speed Control

4.7 Summary



1. The use of least squares to develop a system model that includes the DC motor dynamics, load from a wheel and the signal processing has been covered
2. The use of Root Locus within MATLAB to design a PI controller has been detailed
3. The implementation of a PI controller on hardware that meets a set of requirements (i.e., transient and steady state performance and disturbance rejection) has been covered