UNIVERSITÉ Concordia UNIVERSITY

GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

Department of Mechanical, Industrial, Aerospace Engineering

MECH 473 – Control System Design

Fall 2020

Final Report

Name	**Alexandro Tomassini**
ID	**40029074**

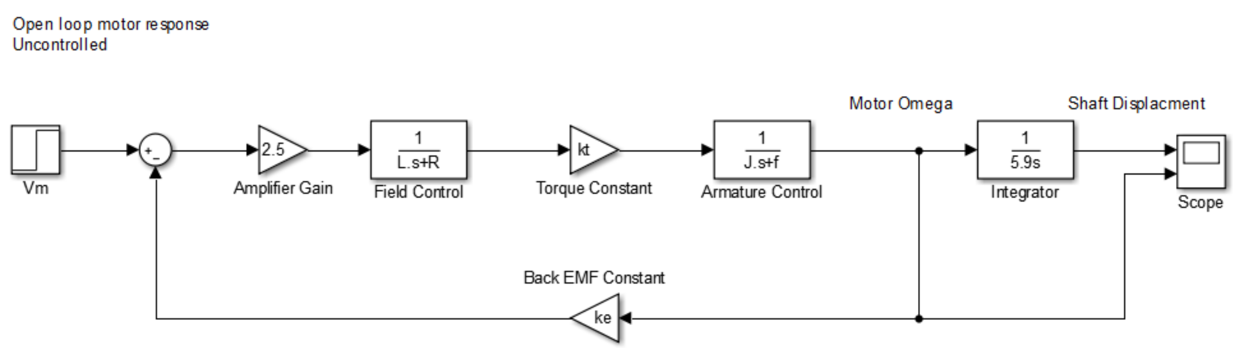
The project objectives are to:

1. Provide a mathematical model for the DC motor and loads, for the driving circuit, and for the feedback. The project coordinator will provide you with specs, and will assign a feedback (tachometer or encoder) scheme, to each group.

Session #3 of the lab manual discussed the method to model a DC motor. The values found below were provided by the manufacture with respect to the motor used. The inertia of the system was calculated by multiplying the rotor inertia by the gear reduction ratio to the second power. All other disc inertia values were added to create a total system inertia value which is found below.

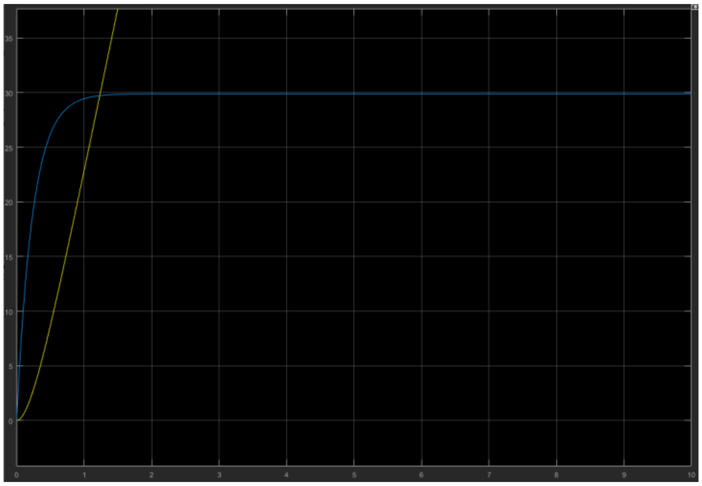
```
kt = 2.29*10^-2; % Torque Constant UNIT: N-m/A
ke = 2.29*10^-2; % Back-EMF Constant UNIT: V/rad/s
R = 0.71; % Unit: Ohm
L = 0.66*10^-3; % Inductance UNIT: Conversion mH ---> H
f = 8.5*10^-4; % Damping Constant UNITS: N-m-s
J = ((7.1*10^-6)*5.9^2)+(1.4*10^-4)+(2.6*10^-4); % Total System Inertia UNITS: kg-m^2
```

The dynamic motor block diagram can be found below. The step input represents the voltage input used to set the motor to the desired position. A gain is applied to the input voltage to drive the motor. The newly stepped up voltage is sent into the field control block where the voltage input is converted to a current. The current is multiplied by a constant which will determine the amount of torque generated by the motor. The torque value will then be multiplied by the armature control block where a motor rotational velocity will be generated. Due to the rotation of the electric motor, a back EMF is created, and the voltage will oppose the rotation of the motor. The magnitude of the back EMF voltage is proportional to the speed of the motor rotating. This can be determined by multiplying the motor velocity with the back EMF constant. The velocity of the motor is sent into an integrator block which will determine the position of the shaft. The velocity of the shaft is less than the motor velocity due to the gear reduction ratio which causes the shaft to rotate once for every 5.9 rotations done by the motor.



```
motor_response_tf =

                   0.05725
  ---------------------------------------
  2.52e-06 s^3 + 0.002714 s^2 + 0.006655 s
```

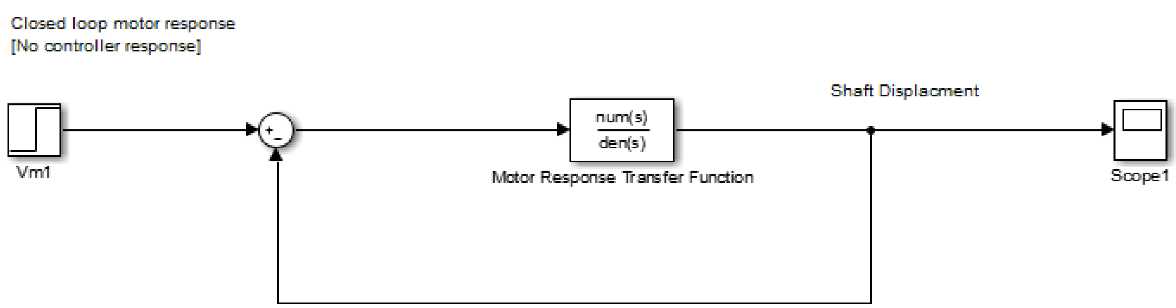
Scope response:



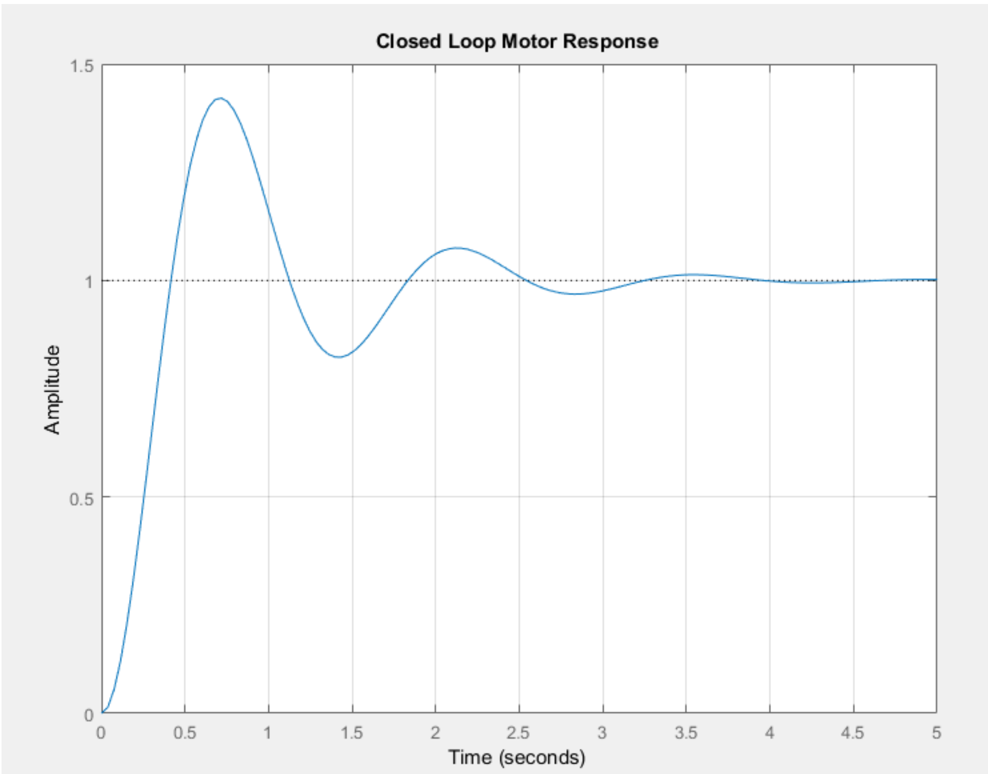
Channel 1 (Top input) represents the position of the shaft (Yellow line)
Channel 2 (Bottom input) represents the velocity of the shaft (Blue line)

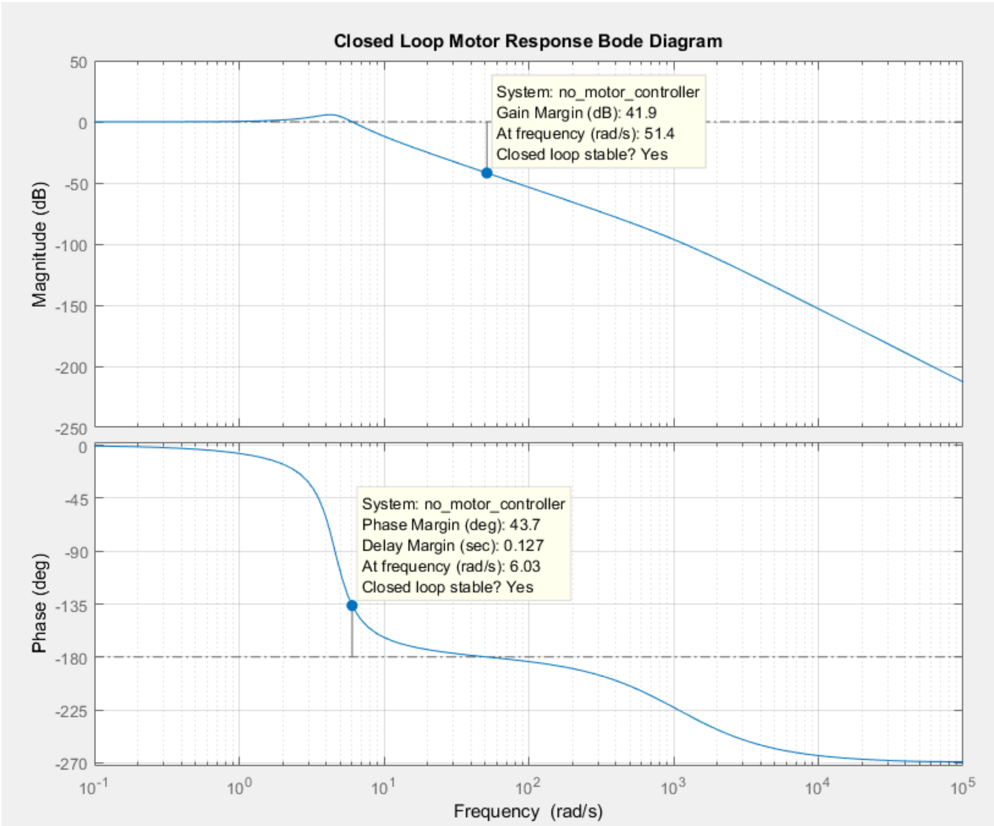
2. Assuming a continuous model with no controller (i.e.: unity forward proportional gain only), select a generator load (resistance) so the system feedback response to a position step input is highly oscillatory (underdamped). Using SIMULINK, show the system step response. Provide a Bode diagram and show/discuss the correlation between the phase margin and damping ratio.

In the response below, we notice the overshoot is 42.13% and the response has large amounts of oscillations.



```
no_motor_controller =

                         0.05725
  ------------------------------------------------
  2.52e-06 s^3 + 0.002714 s^2 + 0.006655 s + 0.05725
```





In the bode diagram representing the closed loop motor response, we notice that the phase margin is 43.7 degrees. If the phase margin was to decrease, the damping ratio would also decrease. If the phase margin becomes negative, the damping ratio will also become negative. In this case, the system response will grow making the system unstable. In order to reduce the system overshoot, increasing the phase margin by shifting the gain crossover to the left will help decrease the systems overshoot due to a higher damping ratio.

Preliminary_Report_Matlab.m

```matlab
clc
kt = 2.29*10^-2; % Torque Constant UNIT: N-m/A
ke = 2.29*10^-2; % Back-EMF Constant UNIT: V/rad/s
R = 0.71; % Unit: Ohm
L = 0.66*10^-3; % Inductance UNIT: Conversion mH ---> H
f = 8.5*10^-4; % Damping Constant UNITS: N-m-s
J = ((7.1*10^-6)*5.9^2)+(1.4*10^-4)+(2.6*10^-4); % Total System Inertia UNITS: kg-m^2

%***Motor Response***

field_block = tf([1],[L R]) %Modeling the field control
armature_block = tf([1],[J f])%Modeling the armature control
%Combined the torque...
%...constant,field block and armature block into a single block.
combined_blocks = field_block*armature_block*kt;
%gear_reduction = tf([1],[1 0])
%Built the EMF feedback response and combined an integrator...
%...(gear reduction) to get the position response. An amp gain is added.
motor_response_tf = feedback(combined_blocks,ke)*gear_reduction*2.5
%... A closed loop motor response was built. Without a controller, the
%response is underdamped.
no_motor_controller = feedback(motor_response_tf,1)

%***Motor Response Charts***

stepplot(motor_response_tf)
title('Open Loop Motor Response')
figure;
stepplot(no_motor_controller)
title('Closed Loop Motor Response')
grid on
figure;
stepinfo(no_motor_controller)
pole(no_motor_controller)
bode(no_motor_controller)
title('Closed Loop Motor Response Bode Diagram')
grid on
figure;
%}

%{...%}
```

Command Window

```
ans =

        RiseTime: 0.2792
    SettlingTime: 3.0450
     SettlingMin: 0.8228
     SettlingMax: 1.4213
       Overshoot: 42.1292
      Undershoot: 0
            Peak: 1.4213
        PeakTime: 0.7179
```

3. According to the frequency domain method, design a cascade lag-type controller for 20% overshoot. For the selected controller, simulate, show and discuss the system position step response and correlate with its Bode diagram.

Equation relating percent overshoot with damping ratio:

$$\zeta = \sqrt{\frac{\left(\ln\left(\frac{PO}{100}\right)\right)^2}{\pi^2 + \left(\ln\left(\frac{PO}{100}\right)\right)^2}}$$

$$\zeta = \sqrt{\frac{\left(\ln\left(\frac{20}{100}\right)\right)^2}{\pi^2 + \left(\ln\left(\frac{20}{100}\right)\right)^2}}$$

$$\zeta = 0.4559$$

Equation relating damping ratio with phase margin:

$$\phi_m = \tan^{-1}\frac{2\zeta}{\sqrt{-2\zeta^2 + \sqrt{1 + 4\zeta^4}}}$$

$$\phi_m = \tan^{-1}\frac{2(0.4559)}{\sqrt{-2(0.4559)^2 + \sqrt{1 + 4(0.4559)^4}}}$$

$$\phi_m = 48.14^o$$

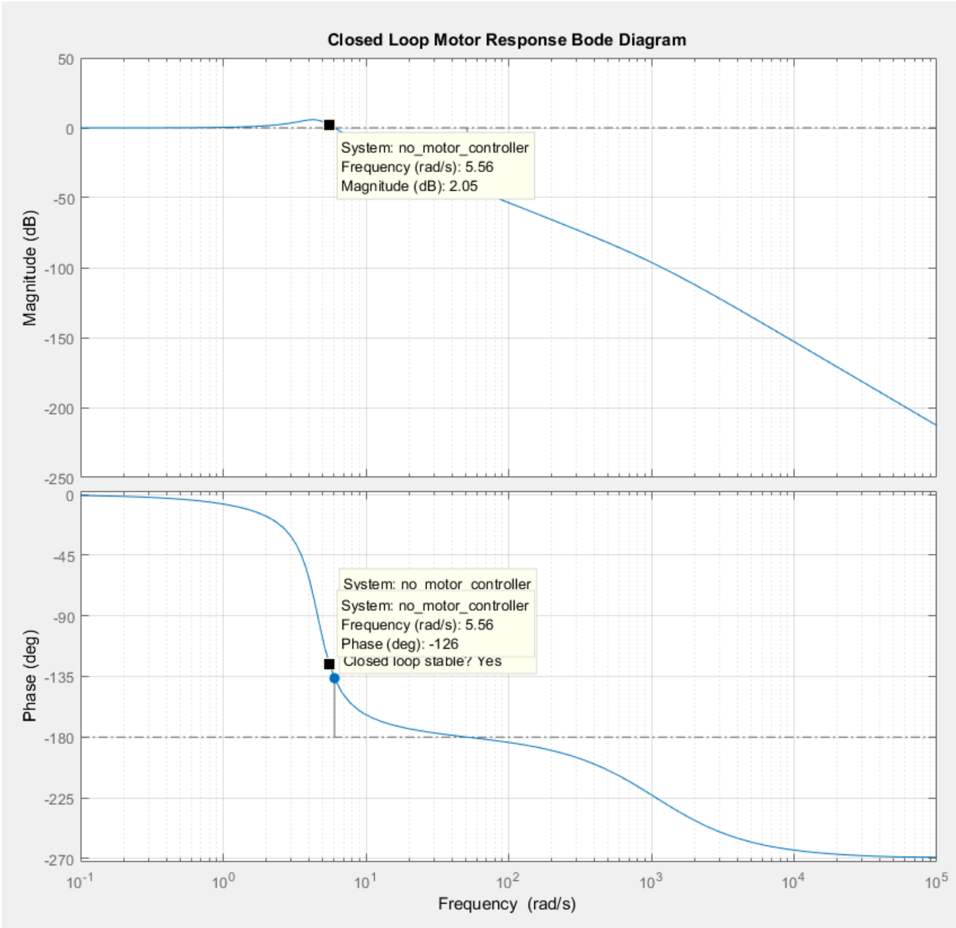
Desired phase margin:

$$\phi_g = \phi_m + \varepsilon$$
$$\phi_g = 48.14^o + 5^o$$
$$\phi_g = 53.14^o$$

Solving for the angle required to get the desired phase margin:

$$Phase\ Angle = 180 - \phi_g$$
$$Phase\ Angle = 180^o - 53.14\,^o$$

$$Phase\ Angle = 126.86^o$$



Gain reduction calculation:

$$Gain\ Reduction = 20\ \log a$$

$$-2.05\ dB = 20\ \log a$$

$$a = 0.7898$$

Calculating the pole and zero placement:

$$Crossover\ Frequency\ (\omega_g) = 5.56\frac{rad}{s} @\ 126^o$$

$$Zero\ (\omega_z) = \frac{\omega_g}{2} = 2.78\frac{rad}{s} = \frac{1}{aT_1}$$

$$T_1 = 0.4554$$

$$\boldsymbol{Zero\ (\omega_z) = 2.78\frac{rad}{s}}$$

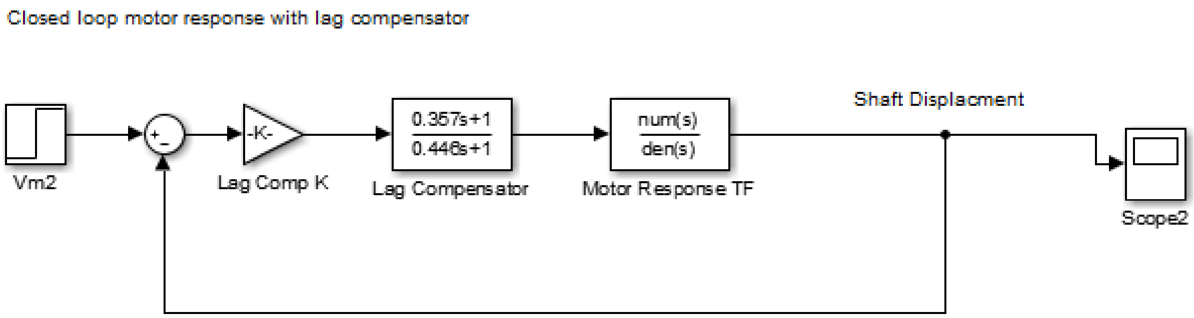
$$Pole\ (\omega_p) = \omega_z * a$$

$$Pole\ (\omega_p) = 2.197\frac{rad}{s}$$

Lag compensator:

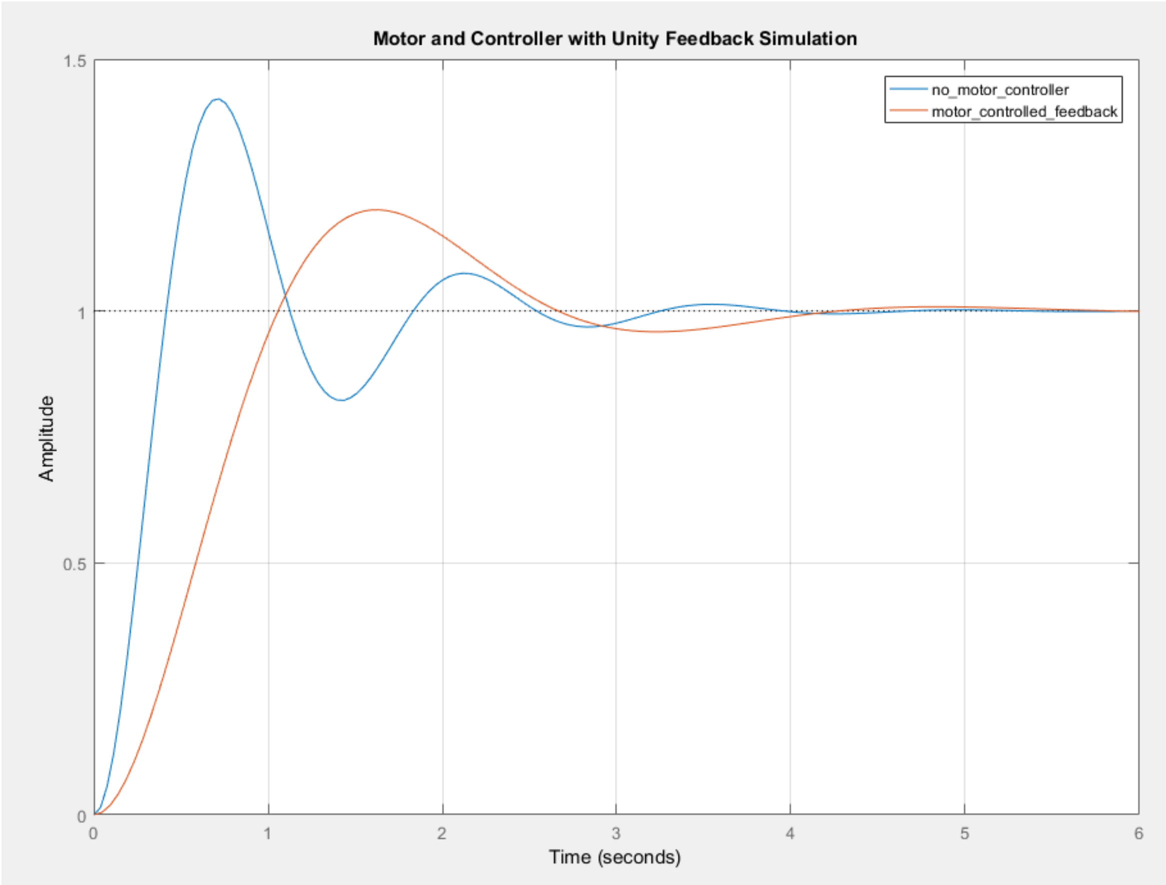
$$Lag\ Compensator = \frac{aT_1 s + 1}{T_1 s + 1}$$

$$Lag\ Compensator = \frac{0.3597s + 1}{0.4554s + 1}$$

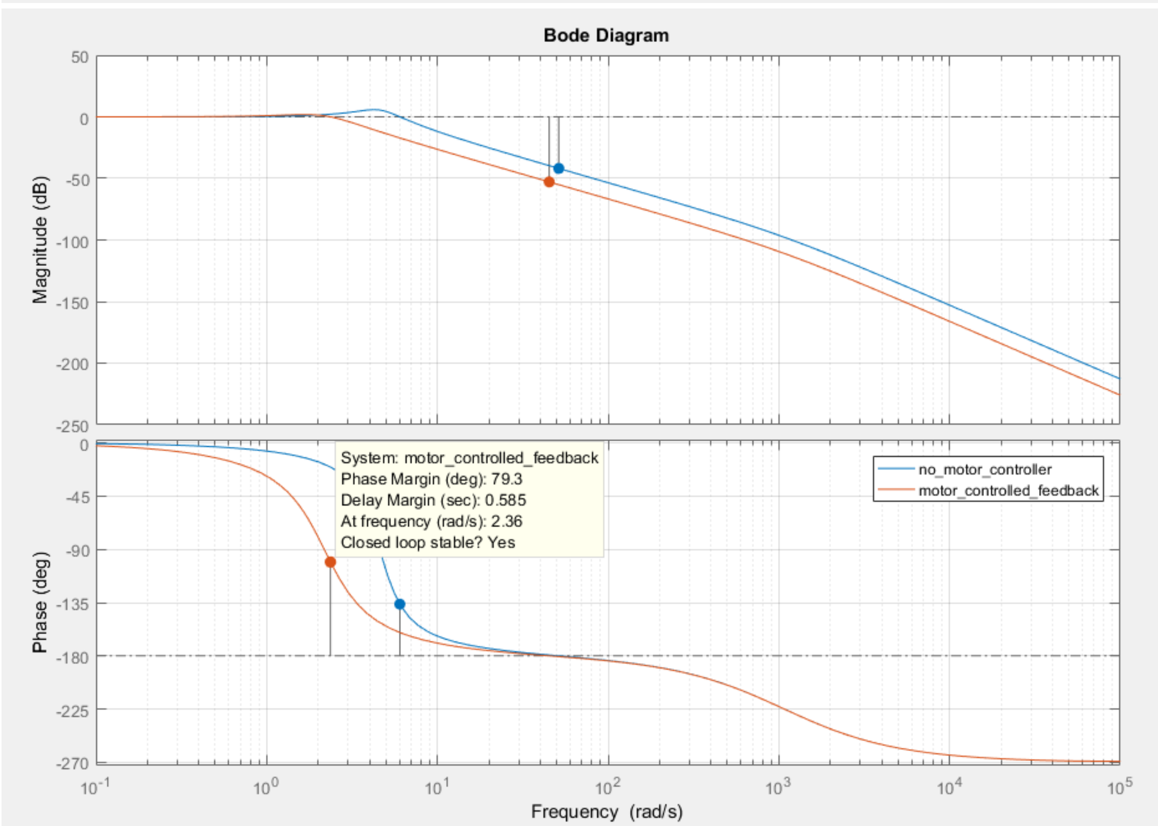


When implementing the lag compensator into the block diagram, the desired overshoot value was not attained. In order to reach the 20% overshoot requirement, a gain of 0.275 was included in the model.

The graph below demonstrates the difference in response behaviour for a unity feedback motor with and without a lag compensator controller. We notice the response for the motor without the controller has a higher overshoot than the response with the lag controller which is too be expected because the controller was designed to achieve a 20% overshoot. Another observation which can be made is the time required to hit the maximum value is shorter for the motor response without a lag compensator. This is to be expected because the lag compensator slows the motor response causing the controlled response to lag the non controlled response. A final observation can be made where the motor response without the controller oscillates more than the controlled motor response. We can establish that the damping ratio for the non controlled response is less than the damping ratio for the controlled response. This statement matches the observation we can make when observing the bode diagram. It is possible to notice that the phase margin from the controlled motor is high than the phase margin for the non controlled motor response.



Motor and Controller with Unity Feedback Simulation
no_motor_controller
motor_controlled_feedback
Amplitude
Time (seconds)
1.5
1
0.5
0
0
1
2
3
4
5
6



Bode Diagram
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
System: motor_controlled_feedback
Phase Margin (deg): 79.3
Delay Margin (sec): 0.585
At frequency (rad/s): 2.36
Closed loop stable? Yes
no_motor_controller
motor_controlled_feedback
50
0
-50
-100
-150
-200
-250
0
-45
-90
-135
-180
-225
-270
10^-1
10^0
10^1
10^2
10^3
10^4
10^5

Preliminary_Report_Matlab.m

```matlab
clc
kt = 2.29*10^-2; % Torque Constant UNIT: N-m/A
ke = 2.29*10^-2; % Back-EMF Constant UNIT: V/rad/s
R = 0.71; % Unit: Ohm
L = 0.66*10^-3; % Inductance UNIT: Conversion mH ---> H
f = 8.5*10^-4; % Damping Constant UNITS: N-m-s
J = ((7.1*10^-6)*5.9^2)+(1.4*10^-4)+(2.6*10^-4); % Total System Inertia UNITS: kg-m^2

%***Motor Response***

field_block = tf([1],[L R]) %Modeling the field control
armature_block = tf([1],[J f])%Modeling the armature control
%Combined the torque...
%...constant,field block and armature block into a single block.
combined_blocks = field_block*armature_block*kt;
%gear_reduction = tf([1],[1 0])
%Built the EMF feedback response and combined an integrator...
%...(gear reduction) to get the position response. An amp gain is added.
motor_response_tf = feedback(combined_blocks,ke)*gear_reduction*2.5
%... A closed loop motor response was built. Without a controller, the
%response is underdamped.
no_motor_controller = feedback(motor_response_tf,1)

%***Motor Response Charts***
%{...%}
%{...%}

%***Lag Comp***
%A higher phase margin is required to reduce the overshoot. This is
%...achieved because the damping ratio is reduced. A gain value is added to
%...reduce the overshoot further.
lag = tf([0.357 1],[0.446 1])
motor_controlled = motor_response_tf*lag*0.275
motor_controlled_feedback = feedback(motor_controlled,1)

stepplot(no_motor_controller,motor_controlled_feedback)
title('Motor and Controller with Unity Feedback Simulation')
grid on
stepinfo(motor_controlled_feedback)
pole(motor_controlled_feedback)
figure;
bode(no_motor_controller,motor_controlled_feedback)
grid on
```

Command Window

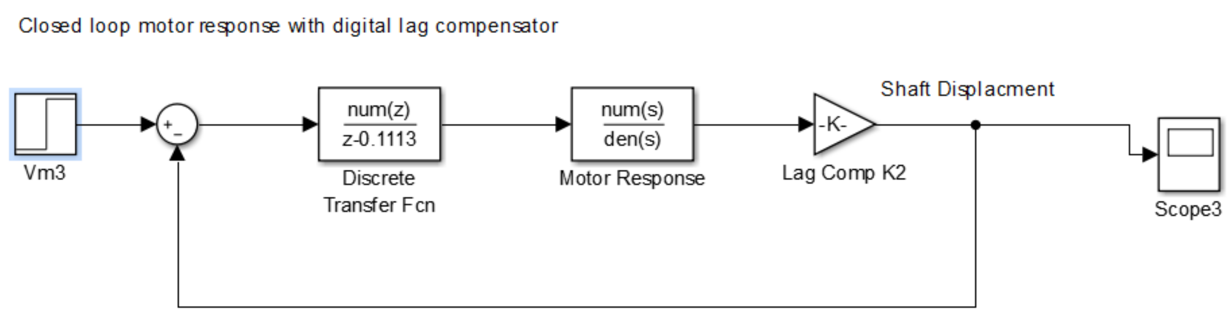
```
    SettlingTime: 3.8199
     SettlingMin: 0.9382
     SettlingMax: 1.2009
       Overshoot: 20.0882
      Undershoot: 0
            Peak: 1.2009
        PeakTime: 1.6366
```

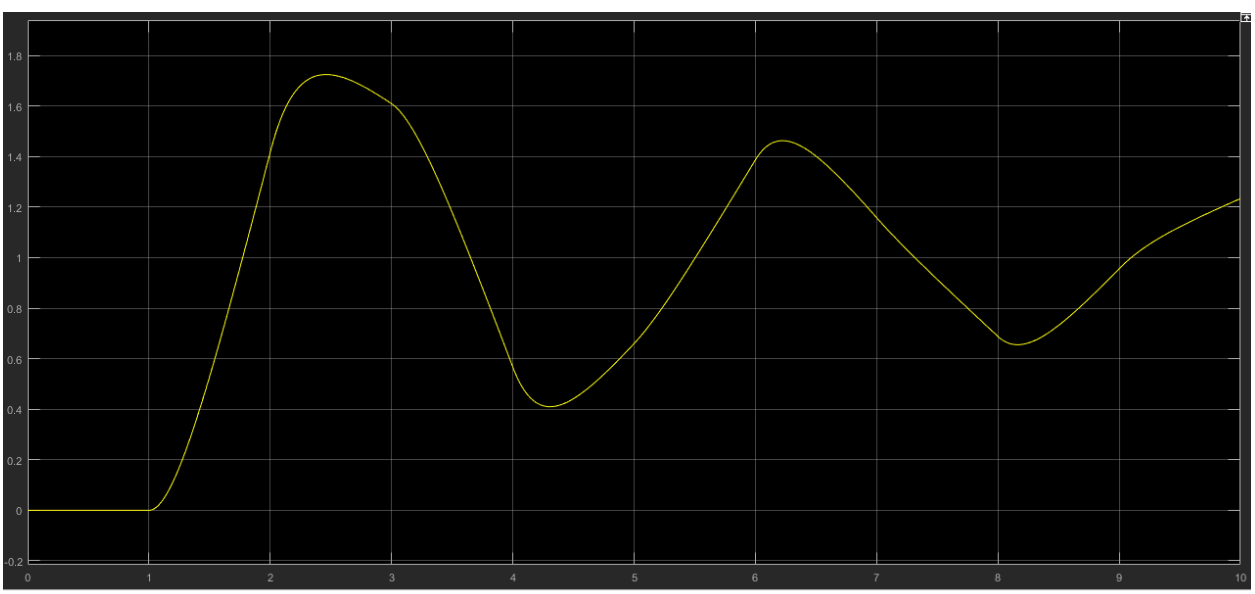
4. Once the proper controller has been selected in 3, convert the continuous controller into a digital controller (see text: Chapter 13, up to page 526). Using SIMULINK, simulate the digital controller and continuous plant. Provide responses/discussions for a variety of proper and improper sampling times.

Matlab was used to calculate the continuous to discrete conversion of the lag compensator. The c2d function was used to convert the lag compensator transfer function and three different sample times where selected (1s, 500ms and 100ms). The response for each sample time can be found below which will allows for a comparison between the different sample times used.

Using a sample time of 1s:

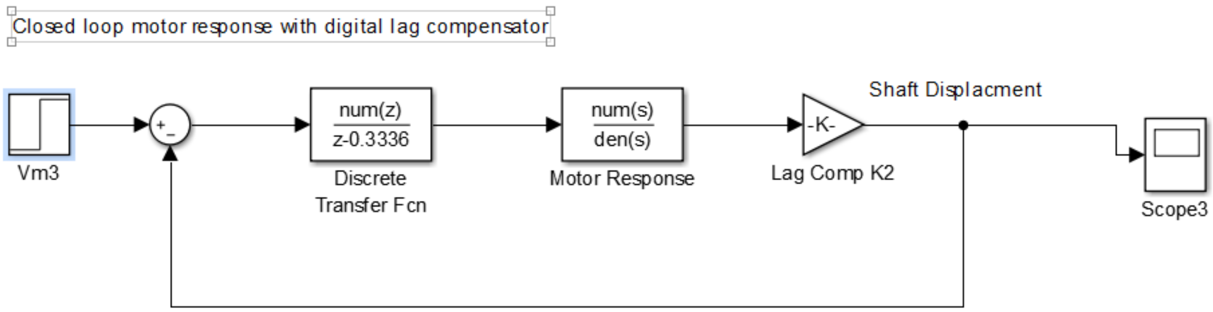
```
sampling_1000_ms =

  0.7899 z + 0.09888
  ------------------
      z - 0.1113
```

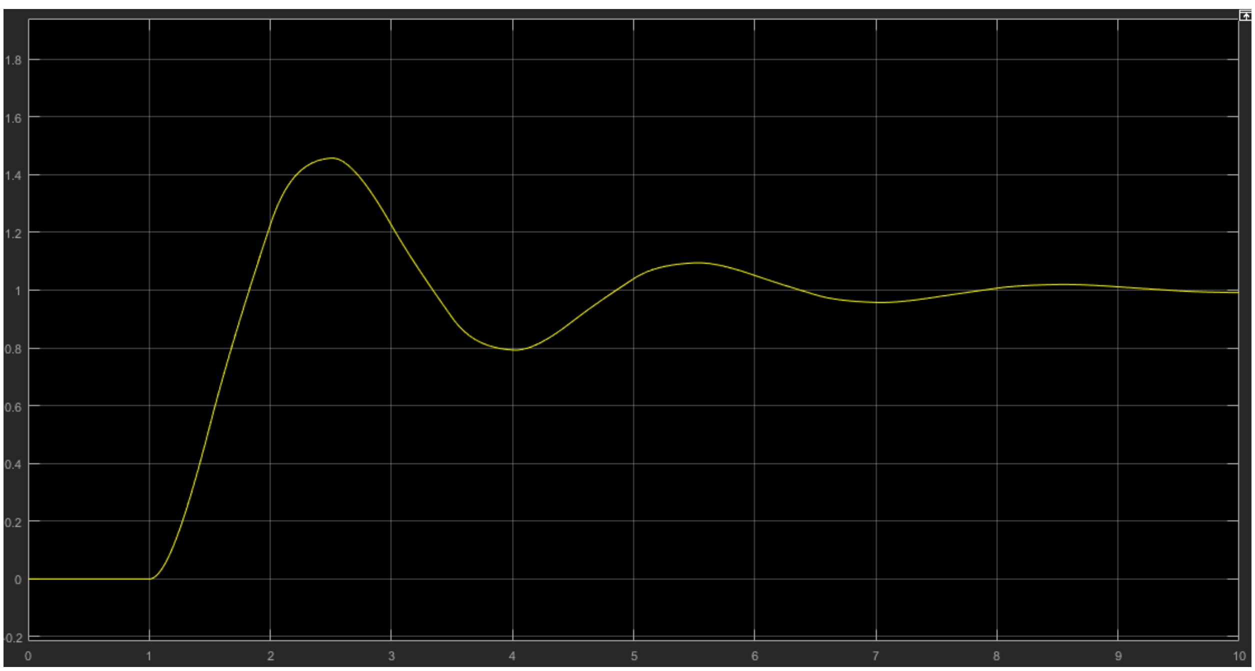




Using a sample time of 500 ms:

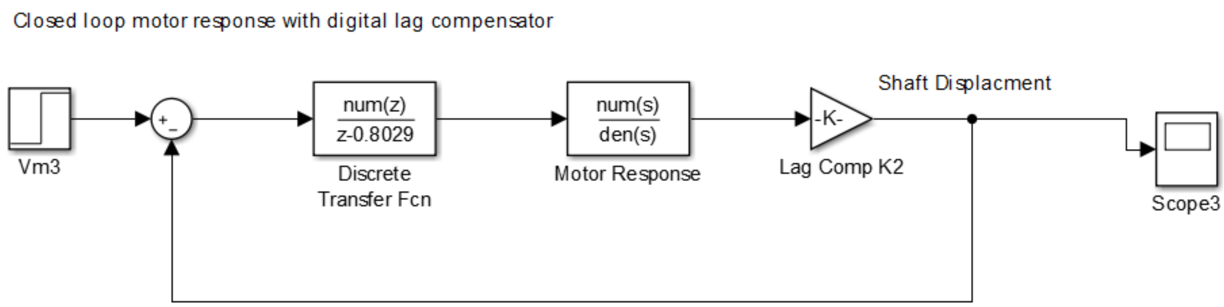
```
sampling_500_ms =

  0.7899 z - 0.1234
  -----------------
     z - 0.3336
```

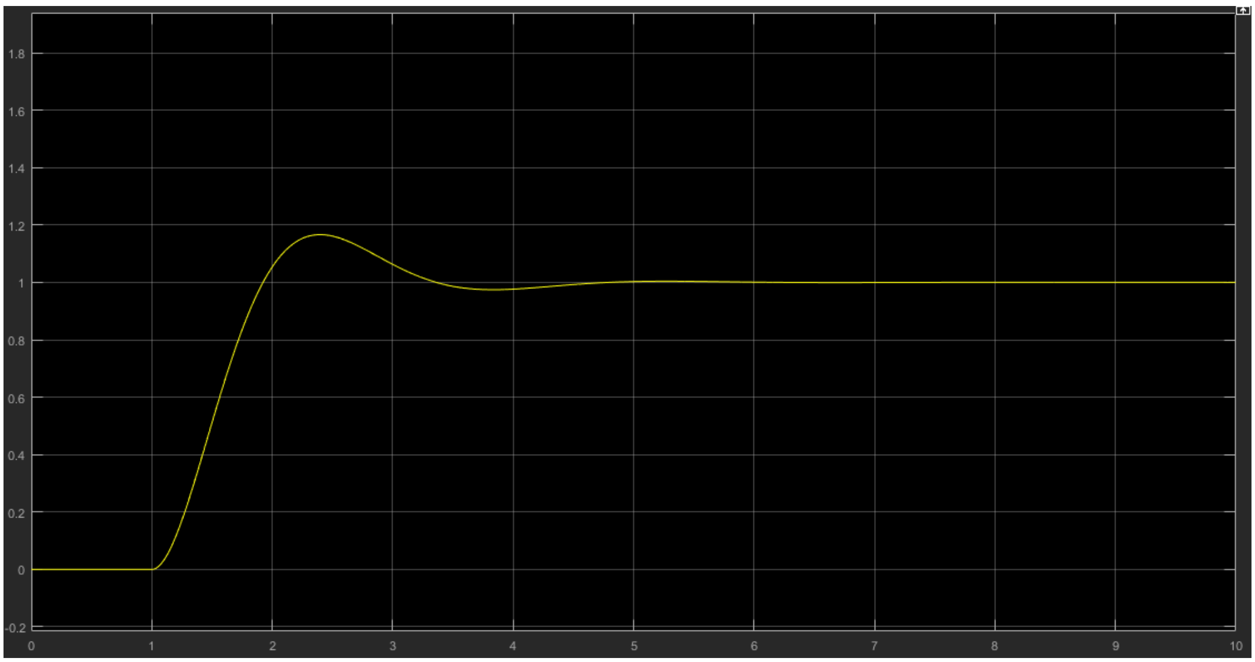




Sampling at 100 ms:

```
sampling_100_ms =

  0.7899 z - 0.5927
  -----------------
     z - 0.8029
```





We notice that as the sample time increases, the motor response achieves a similar shape to the continuous response. For the sample time of 1s, we notice that the overshoot is larger than the uncontrolled motor response and we notice that the oscillations are much more pronounced. Looking at the graph, we can estimate the overshoot to be 75% for the 1s sample time. For the sample time of 500 ms, we notice a large overshoot of about 50% and the oscillations have been largely reduced when compared to the 1s sample time. The final sample time of 100 ms shows a response which closely follows the continuous response. The 100 ms sample time response overshoot can be approximated to 18% which is slightly below the continuous response overshoot value.

5. For the variety of sampling times, convert the controllers found in 4 into a difference equation (see text: Section 12.4 p. 481). Implement the difference equation in C-language on the digital computer controller (Matlab/Simulink Real Time Controller) and store the step response on the oscilloscope. Compare the experimental results with those found in 4. The project coordinator will show you how to implement your program within the interface software provided and will show you how to change the sampling time on the computer.

The continuous lag compensator was found to be:

$$Lag\ Compensator = \frac{0.3597s + 1}{0.4554s + 1}$$

Three sample times (1000ms, 500ms and 100 ms) were used to convert the continuous controller found above into a discrete controller. Using the csd function in Matlab at different sampling times, the continuous lag compensator was converted into different discrete lag compensators which have different responses to step inputs. The method used for converting the continuous controller into the discrete controller was the zero-order hold method.

Discrete lag compensator using 1000 ms sampling time:

```
ZOH_sampling_1000_ms =

  0.7899 z + 0.09888
  ------------------
      z - 0.1113
```

Difference equation calculation:

$$\frac{Y(z)}{U(z)} = \frac{0.7899z + 0.09888}{z - 0.1113}$$

Multiply the numerator and denominator by z^{-1}

$$\frac{Y(z)}{U(z)} = \frac{0.7899 + 0.09888z^{-1}}{1 - 0.1113z^{-1}}$$

Cross multiply

$$Y(z)(1 - 0.1113z^{-1}) = U(z)(0.7899 + 0.09888z^{-1})$$

$$Y(z) - Y(z)0.1113z^{-1} = U(z)0.7899 + U(z)0.09888z^{-1}$$

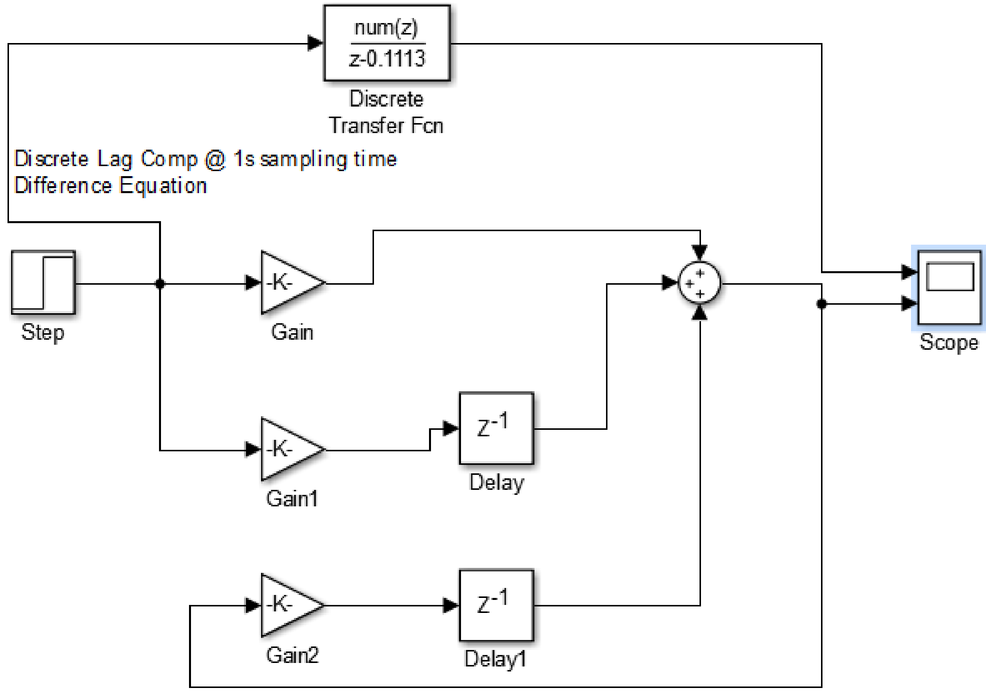
Inverse Z-transform

$$Y(k) - Y(k)0.1113(k-1) = U(k)0.7899 + U(z)0.09888(k-1)$$

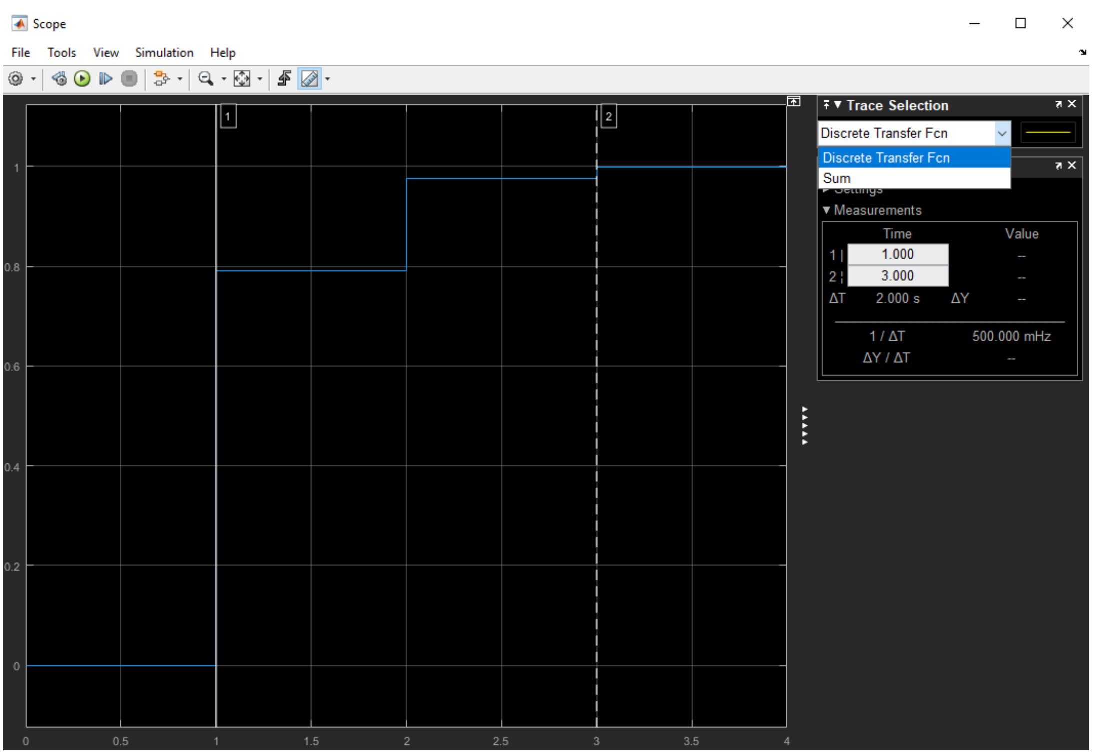
Isolate Y(k)

$$\boldsymbol{Y(k) = U(k)0.7899 + U(z)0.09888(k-1) + Y(k)0.1113(k-1)}$$

Simulink block diagram representing the difference equation and the associated discrete transfer function:



Difference equation response to a step input:



The scope above plotted the difference equation and the associated discrete controller. It demonstrated that both responses are identical because the lines overlap each other. This proves that the difference equations derived have identical response characteristics to the respective discrete response.

Discrete lag compensator using 500 ms sampling time:

```
ZOH_sampling_500_ms =

  0.7899 z - 0.1234
  -----------------
     z - 0.3336
```

Difference equation calculation:

$$\frac{Y(z)}{U(z)} = \frac{0.7899z - 0.1234}{z - 0.3336}$$

Multiply the numerator and denominator by z^{-1}

$$\frac{Y(z)}{U(z)} = \frac{0.7899 - 0.1234z^{-1}}{1 - 0.3336z^{-1}}$$

Cross multiply

$$Y(z)(1 - 0.3336z^{-1}) = U(z)(0.7899 - 0.1234z^{-1})$$

$$Y(z) - Y(z)0.3336z^{-1} = U(z)0.7899 - U(z)0.1234z^{-1}$$

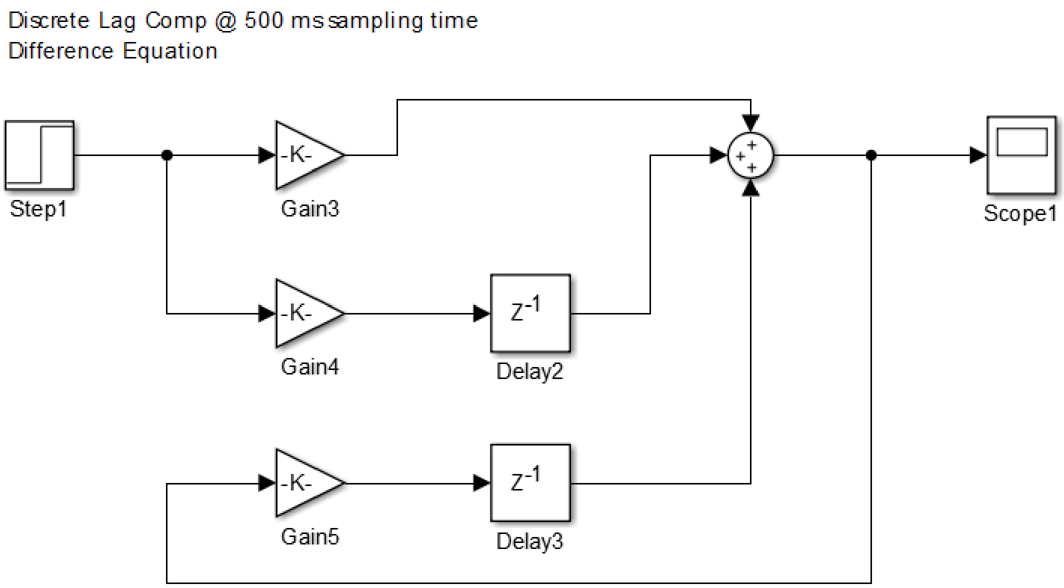
Inverse Z-transform

$$Y(k) - Y(k)0.3336(k-1) = U(k)0.7899 - U(k)0.1234(k-1)$$

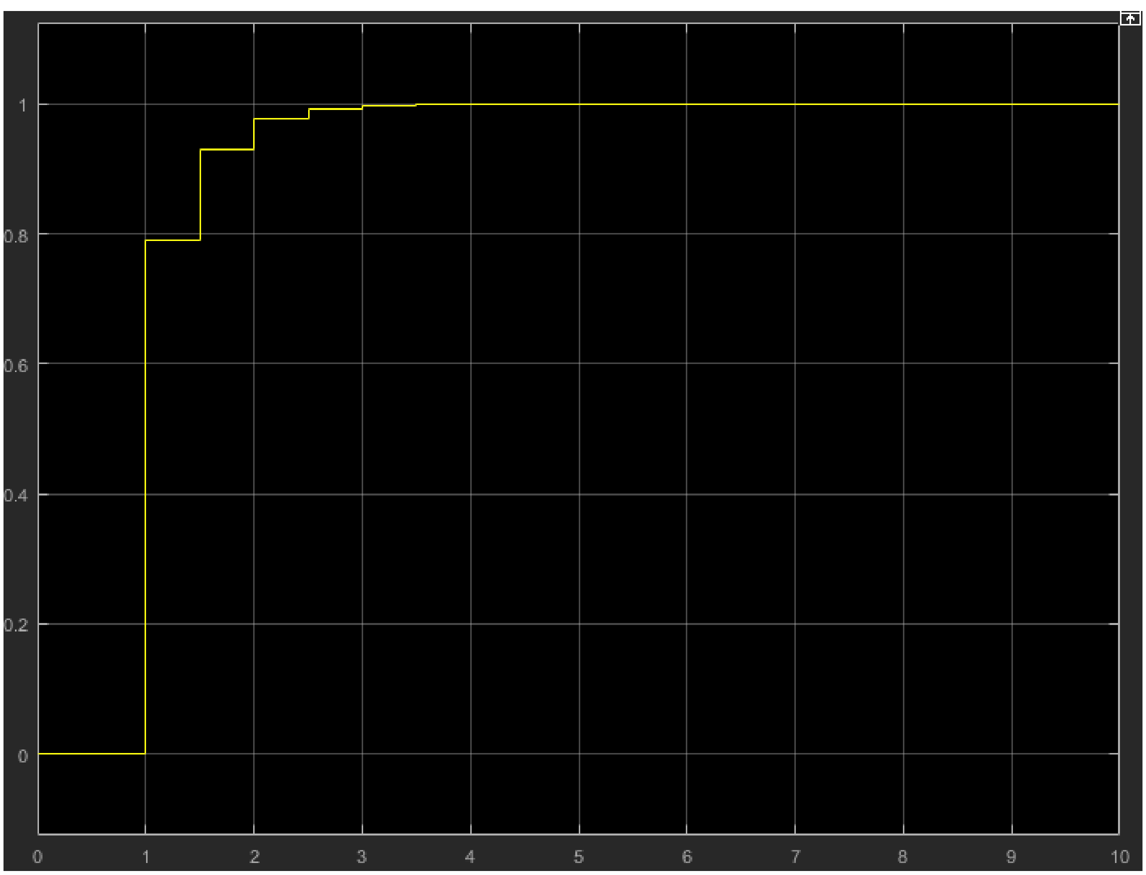
Isolate Y(k)

$$\boldsymbol{Y(k) = U(k)0.7899 - U(k)0.1234(k-1) + Y(k)0.3336(k-1)}$$

Simulink block diagram representing the difference equation:



Difference equation response to a step input:



Discrete lag compensator using 100 ms sampling time:

```
ZOH_sampling_100_ms =

  0.7899 z - 0.5927
  -----------------
     z - 0.8029
```

Difference equation calculation:

$$\frac{Y(z)}{U(z)} = \frac{0.7899z - 0.5927}{z - 0.8029}$$

Multiply the numerator and denominator by z^{-1}

$$\frac{Y(z)}{U(z)} = \frac{0.7899 - 0.5927z^{-1}}{1 - 0.8029z^{-1}}$$

Cross multiply

$$Y(z)(1 - 0.8029z^{-1}) = U(z)(0.7899 - 0.5927z^{-1})$$

$$Y(z) - Y(z)0.8029z^{-1} = U(z)0.7899 - U(z)0.5927z^{-1}$$

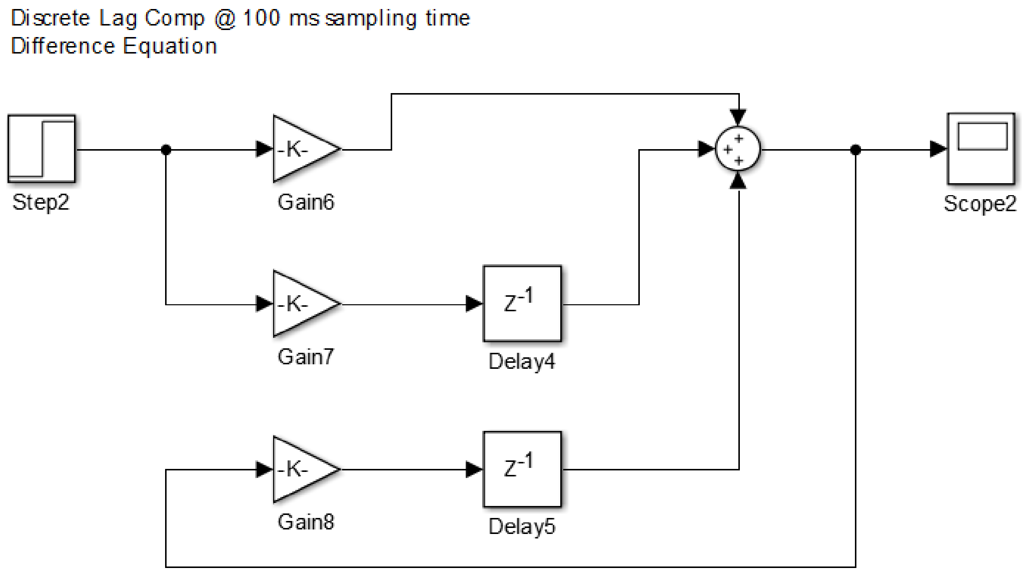
Inverse Z-transform

$$Y(k) - Y(k)0.8029(k-1) = U(k)0.7899 - U(k)0.5927(k-1)$$

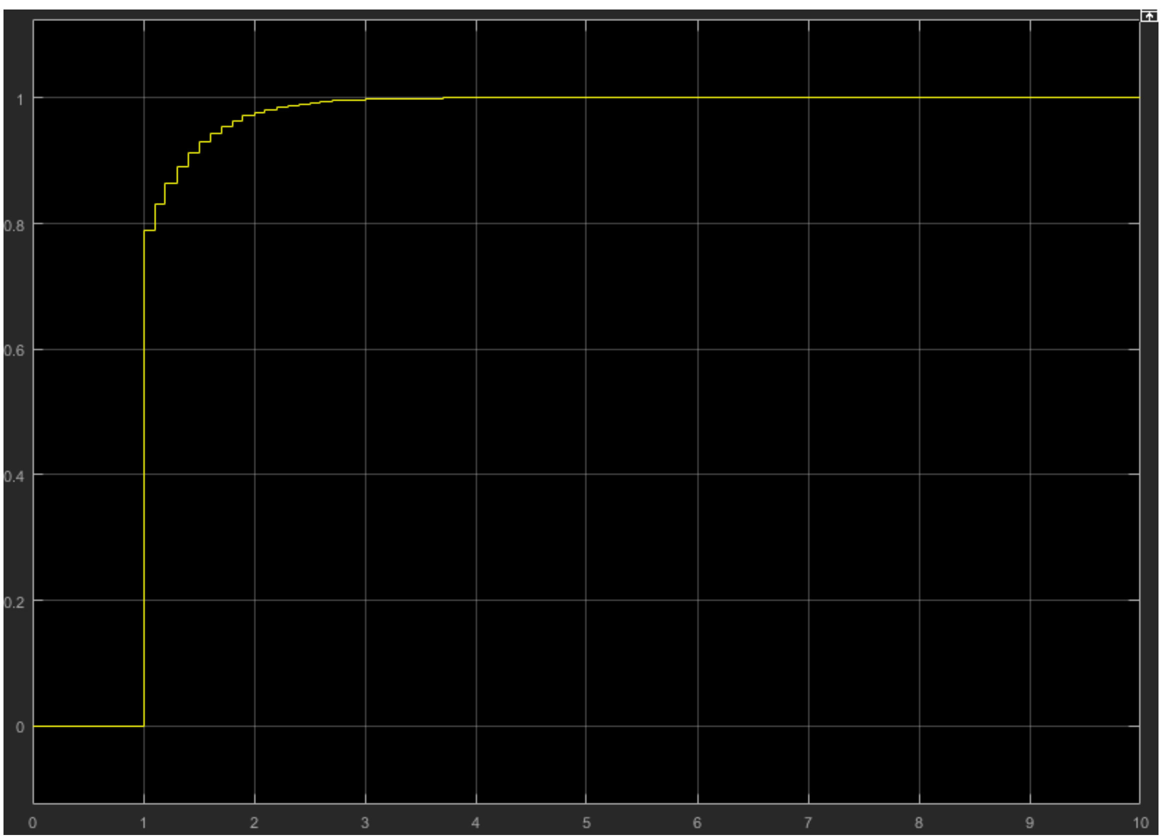
Isolate Y(k)

$$\boldsymbol{Y(k) = U(k)0.7899 - U(k)0.5927(k-1) + Y(k)0.8029(k-1)}$$

Simulink block diagram representing the difference equation:



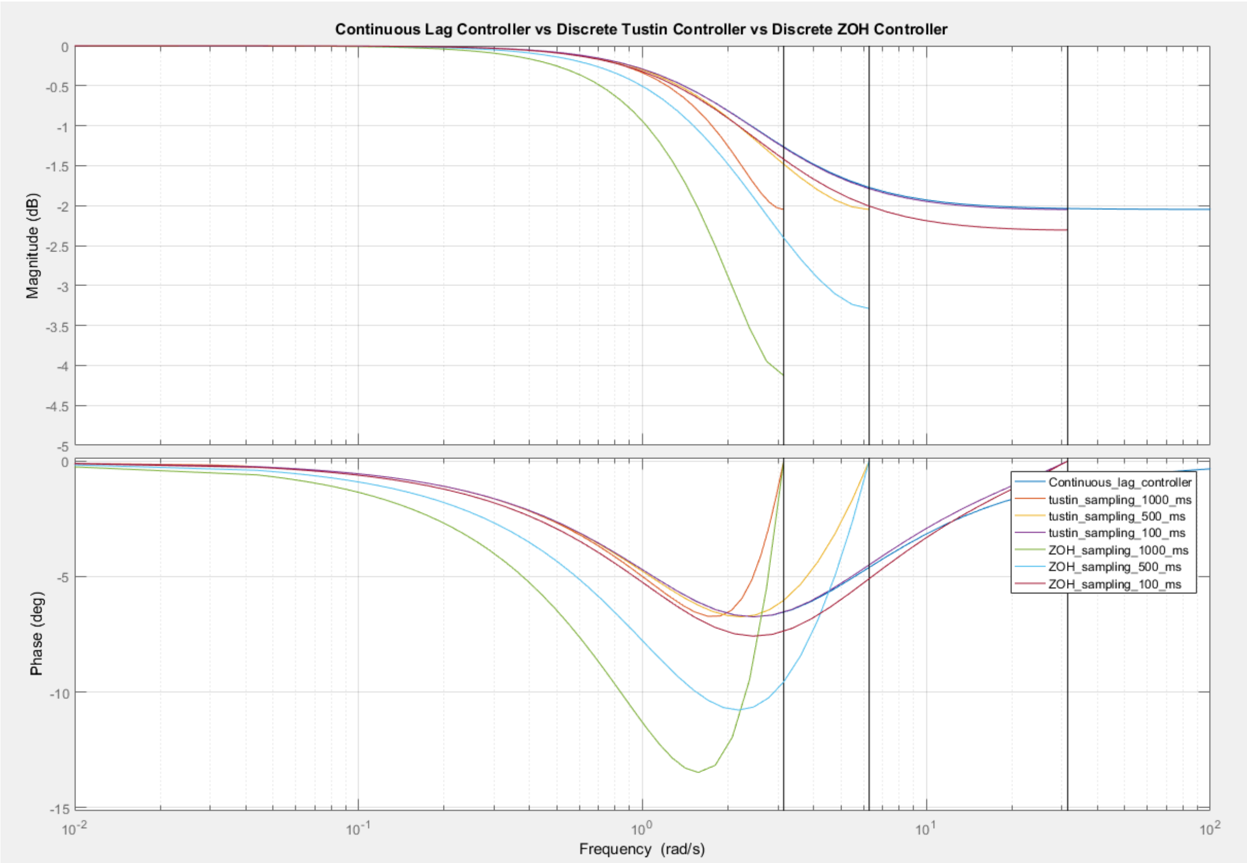
Difference equation response to a step input:



When observing the different responses, it is possible to see the difference equations behave in the same manner with respect to a step response. Each difference equation had identical responses to the discrete transfer function. As the sampling intervals become smaller and occur more often, the response behaves closer to the continuous response. When comparing the 1000 ms sampling time to the 100 ms sampling time, it is possible to notice that the 100 ms sampling response follows a curved response which resembles the continuous response while the 1000 ms response has no specific shape to it and it is very choppy due to the large sample time.

6. After obtaining the system model from step 2 above, convert the continuous model into an equivalent digital model (see text: Section 13.4 p. 526). Using the frequency domain method of Section 13.6, p. 540, design a digital controller for 20% overshoot. A variety of properly and improperly selected sampling times should be performed for SIMULINK simulation (see text: Section 12.4.1 p.484, Section 13.4.1 p. 528).

Designing a controller using the frequency domain method was conducted in part 3 with success. In order to design a discrete controller for a discrete plant, the frequency domain method cannot be directly applied [2]. The lag compensator designed for part 3 will be converted into a discrete controller using the Tustin conversion method.



The above chart compared the zero-order hold method and the Tustin method at different sampling times to the continuous response of the lag controller. The frequency response to the step response was observed. When looking at the chart above, the Tustin method demonstrates a closer response to the continuous lag controller as opposed to the zoh method. As the sampling time increases, the response begins to diverge from the continuous response at higher frequencies. The continuous to discrete conversion using the zero-order hold method does not follow the continuous method but the divergence from the original continuous system is less pronounced for smaller sampling times. When comparing the Tustin sampling time of 100 ms to the zoh sampling time of 100 ms, the Tustin method more closely follows the continuous response. When observing the phase chart, the Tustin response begins to diverge from the continuous response at 3 rad/s while the zoh method diverged at 10^{-1} rad/s. For the magnitude plot, it is possible to notice that the Tustin method does not diverge whatsoever at 100 ms sampling time while the zoh method began to diverge at 1 rad/s.

The Tustin method was used to discretize the plant/motor transfer function at different sampling times as well as the lag controller found in part 3. The Tustin method was used due to its frequency response being closer to the continuous response.

Discretized motor transfer function using 1000 ms sampling time:

```
motor_tf_sampling_1000_ms =

  2.367 z^3 + 7.101 z^2 + 7.101 z + 2.367
  ---------------------------------------
   z^3 + 0.0989 z^2 - 0.9967 z - 0.1022
```

Discrete transfer function representing the entire discrete system using 1000 ms sampling time. The discrete controller, plant and the unity feedback were combined to simplify the transfer function into a single block.

```
Discrete_motor_controlled_feedback_1000ms =

  0.5804 z^4 + 1.836 z^3 + 2.025 z^2 + 0.8646 z + 0.09472
  -------------------------------------------------------
   1.58 z^4 + 1.982 z^3 + 1.033 z^2 + 0.7158 z + 0.08995
```

Discretized motor transfer function using 500 ms sampling time:

```
motor_tf_sampling_500_ms =

  0.8154 z^3 + 2.446 z^2 + 2.446 z + 0.8154
  -----------------------------------------
    z^3 - 0.2463 z^2 - 0.9908 z + 0.2371
```

Discrete transfer function representing the entire discrete system using 500 ms sampling time. The discrete controller, plant and the unity feedback were combined to simplify the transfer function into a single block.

```
Discrete_motor_controlled_feedback_500ms =

  0.1921 z^4 + 0.5416 z^3 + 0.4725 z^2 + 0.08839 z - 0.03456
  -----------------------------------------------------------
   1.192 z^4 + 0.004135 z^3 - 0.4466 z^2 + 0.614 z - 0.1036
```

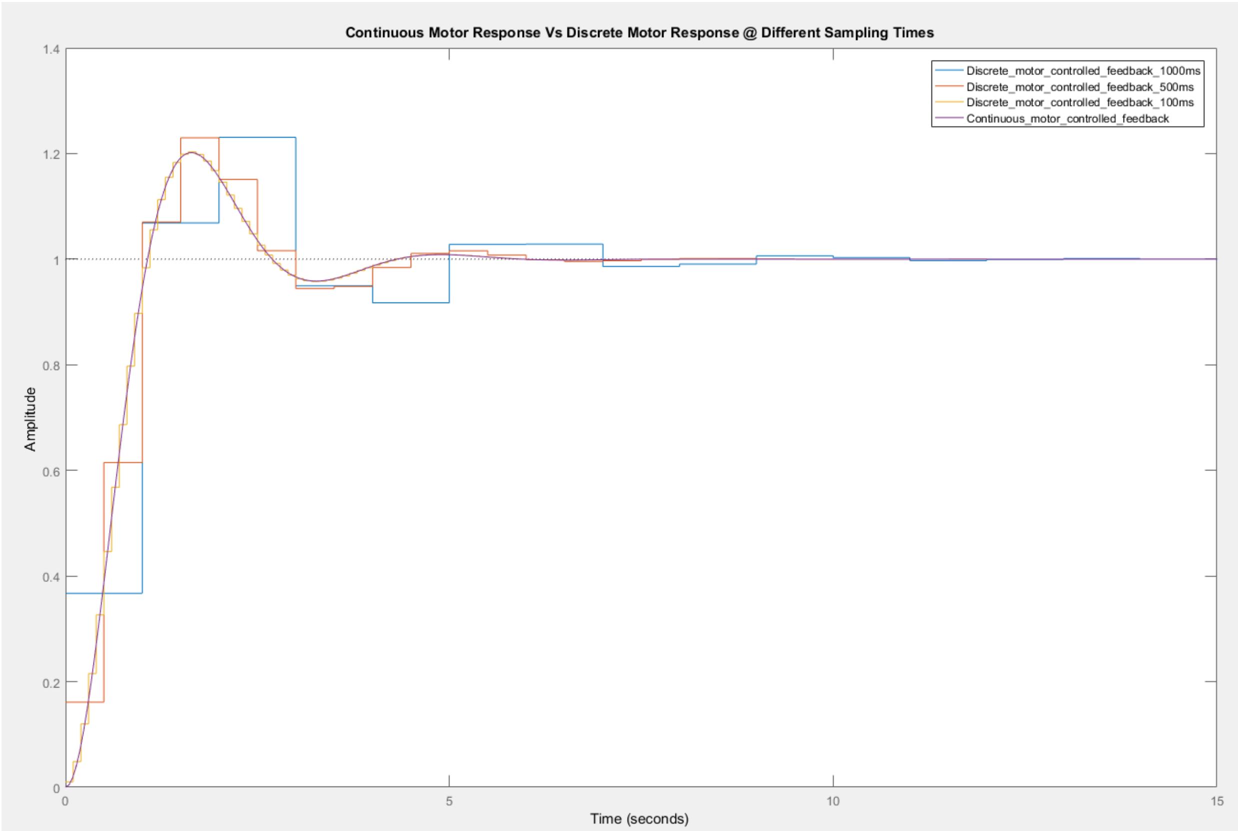
Discretized motor transfer function using 100 ms sampling time:

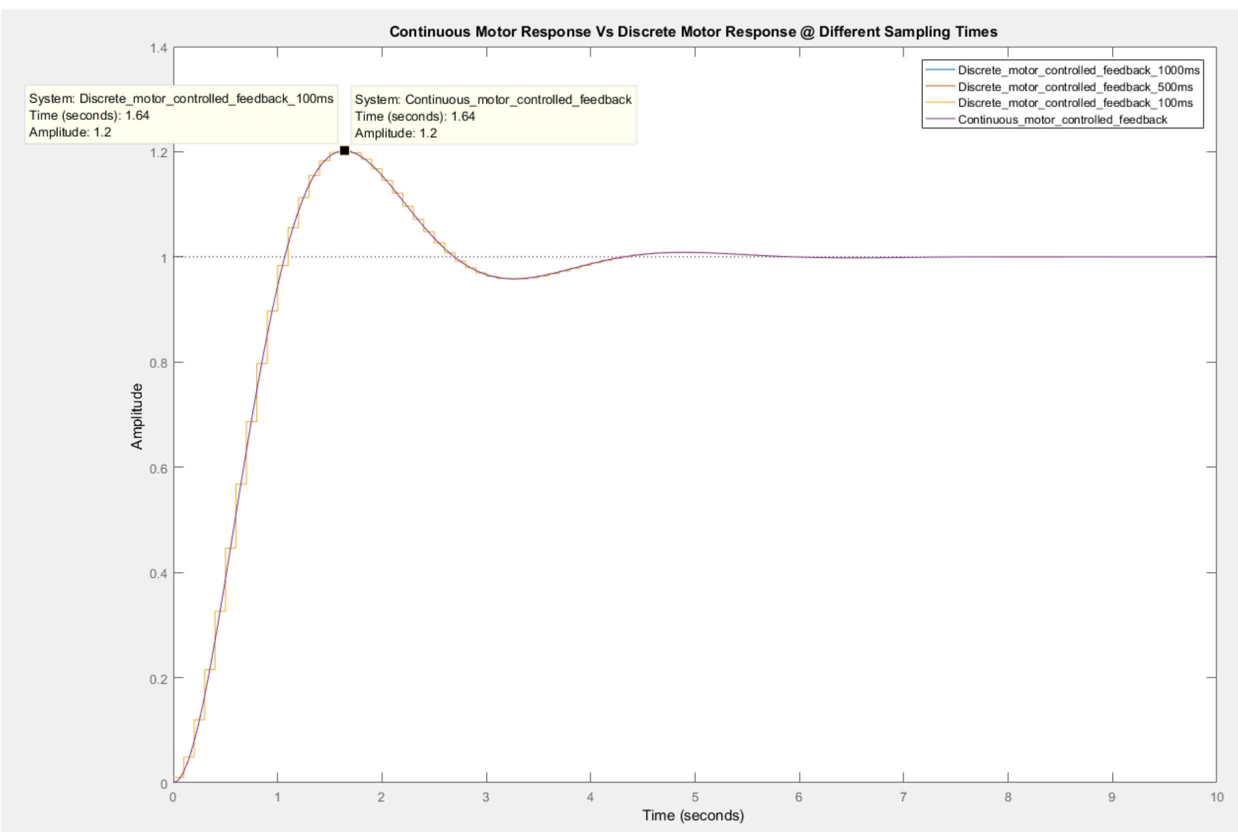
```
motor_tf_sampling_100_ms =

  0.04621 z^3 + 0.1386 z^2 + 0.1386 z + 0.04621
  ---------------------------------------------
      z^3 - 0.8177 z^2 - 0.9349 z + 0.7526
```

Discrete transfer function representing the entire discrete system using 100 ms sampling time. The discrete controller, plant and the unity feedback were combined to simplify the transfer function into a single block.

```
Discrete_motor_controlled_feedback_100ms =

  0.01021 z^4 + 0.02291 z^3 + 0.007474 z^2 - 0.01294 z - 0.007716
  ----------------------------------------------------------------
        1.01 z^4 - 1.597 z^3 - 0.2715 z^2 + 1.49 z - 0.6114
```

The chart below compares the continuous motor and controller response to the discretized motor and controller response. It is evident that as the sampling time decreases, the response behaves closer to the continuous observation. When looking at the 1000 ms and 500 ms sampling times, it is possible to notice that the overshoot for both responses is higher than 20%. When looking at the continuous response in relation to the 100 ms sampling time response, it is possible to notice that the overshoots are identical which meets the design criteria.


Continuous Motor Response Vs Discrete Motor Response @ Different Sampling Times
Discrete_motor_controlled_feedback_1000ms
Discrete_motor_controlled_feedback_500ms
Discrete_motor_controlled_feedback_100ms
Continuous_motor_controlled_feedback
Amplitude
Time (seconds)


Continuous Motor Response Vs Discrete Motor Response @ Different Sampling Times
System: Discrete_motor_controlled_feedback_100ms
Time (seconds): 1.64
Amplitude: 1.2
System: Continuous_motor_controlled_feedback
Time (seconds): 1.64
Amplitude: 1.2
Discrete_motor_controlled_feedback_1000ms
Discrete_motor_controlled_feedback_500ms
Discrete_motor_controlled_feedback_100ms
Continuous_motor_controlled_feedback
Amplitude
Time (seconds)

7. Convert the digital controllers into difference equations and repeat step 5.

The controller used in part 6 was the continuous controller found in part 3 which has been converted using the Tustin method in Matlab.

Discrete lag compensator using 1000 ms sampling time:

```
tustin_sampling_1000_ms =

  0.8998 z + 0.1468
  -----------------
     z + 0.04668
```

Difference equation calculation:

$$\frac{Y(z)}{U(z)} = \frac{0.8998z + 0.1468}{z + 0.04668}$$

Multiply the numerator and denominator by z^{-1}

$$\frac{Y(z)}{U(z)} = \frac{0.8998 + 0.1468z^{-1}}{1 + 0.04668z^{-1}}$$

Cross multiply

$$Y(z)(1 + 0.04668z^{-1}) = U(z)(0.8998 + 0.1468z^{-1})$$

$$Y(z) + Y(z)0.04668z^{-1} = U(z)0.8998 + U(z)0.1468z^{-1}$$

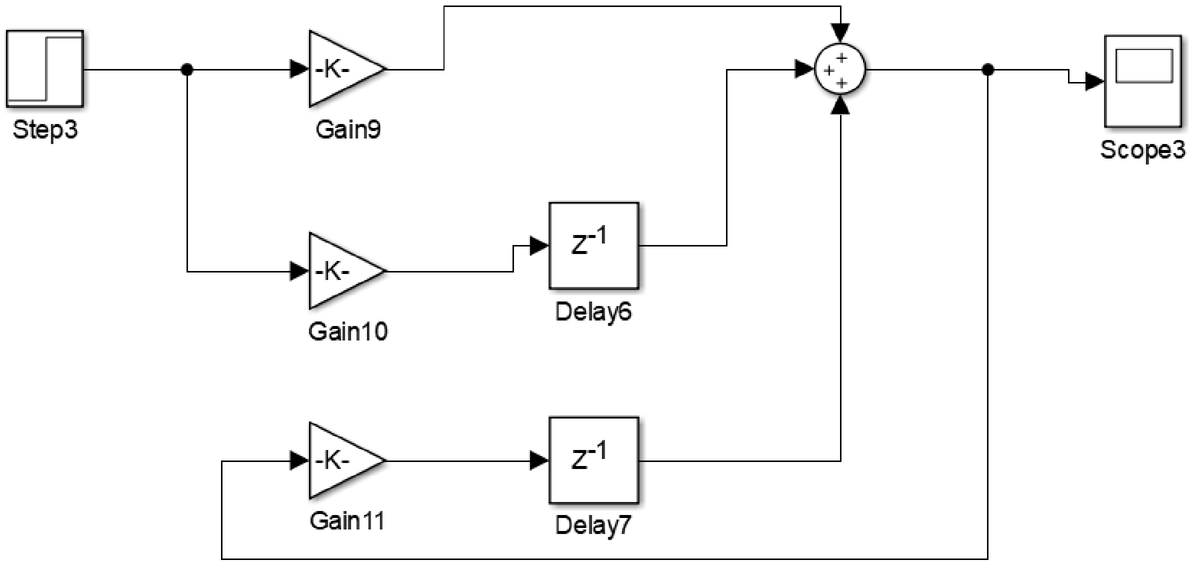
Inverse Z-transform

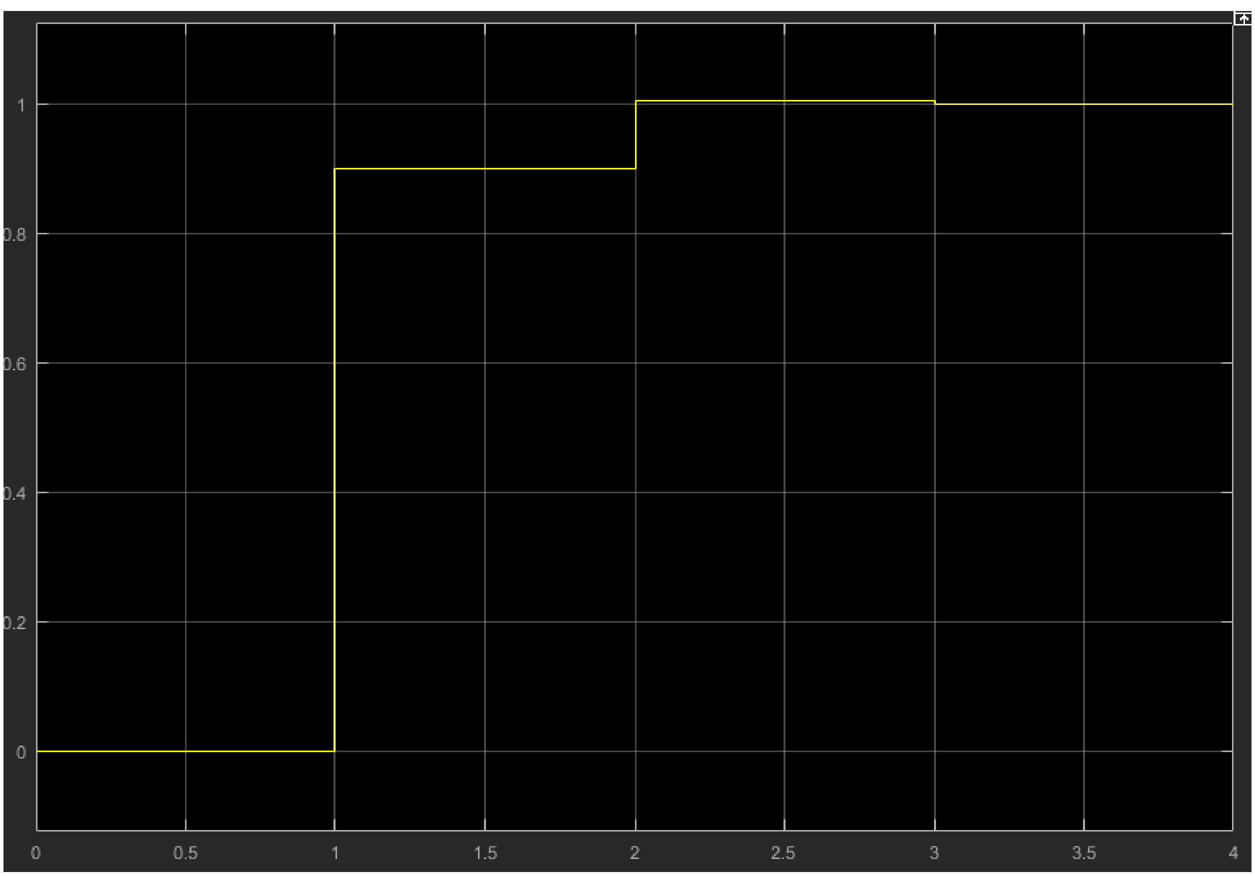
$$Y(k) + Y(k)0.04668(k-1) = U(k)0.8998 + U(k)0.1468(k-1)$$

Isolate Y(k)

$$\boldsymbol{Y(k) = U(k)0.8998 + U(k)0.1468(k-1) - Y(k)0.04668(k-1)}$$

Discrete Lag Comp using Tustin @ 1s sampling time
Difference Equation





Discrete lag compensator using 500 ms sampling time:

```
tustin_sampling_500_ms =

  0.8643 z - 0.1555
  -----------------
     z - 0.2912
```

Difference equation calculation:

$$\frac{Y(z)}{U(z)} = \frac{0.8643z - 0.1555}{z - 0.2912}$$

Multiply the numerator and denominator by z^{-1}

$$\frac{Y(z)}{U(z)} = \frac{0.8643 - 0.1555z^{-1}}{1 - 0.2912z^{-1}}$$

Cross multiply

$$Y(z)(1 - 0.2912z^{-1}) = U(z)(0.8643 - 0.1555z^{-1})$$

$$Y(z) - Y(z)0.2912z^{-1} = U(z)0.8643 - U(z)0.1555z^{-1}$$

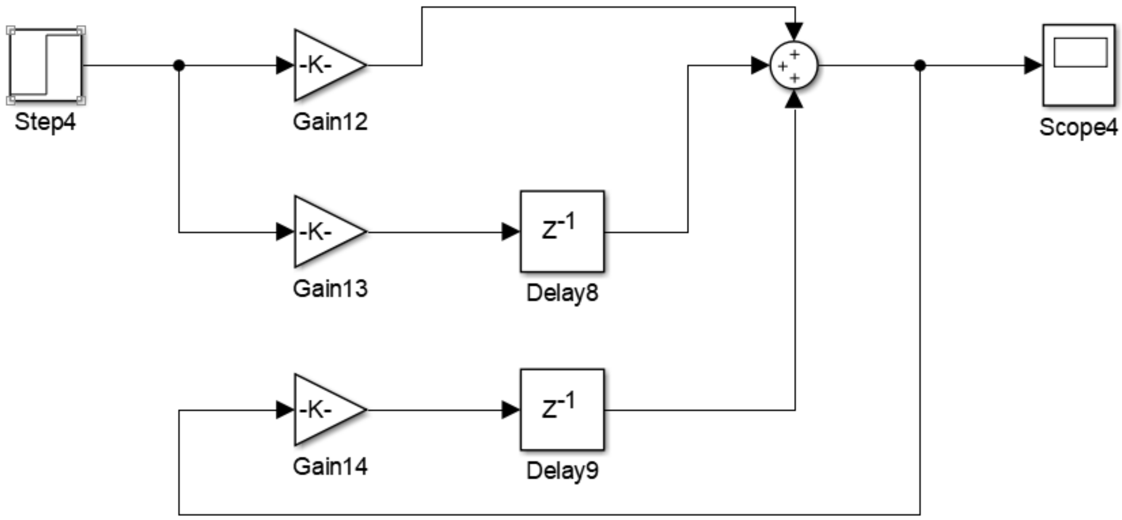
Inverse Z-transform

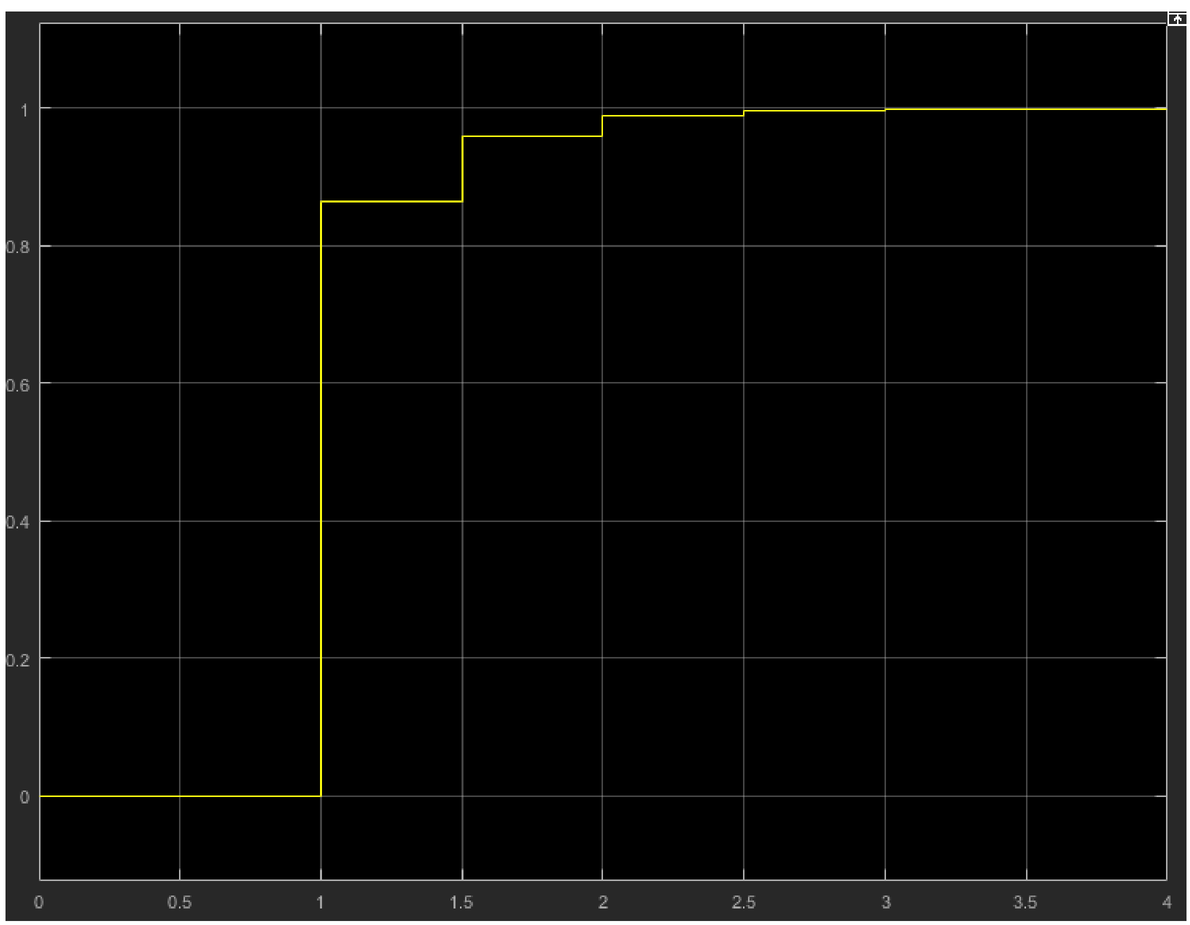
$$Y(k) - Y(k)0.2912(k - 1) = U(k)0.8643 - U(k)0.1555(k - 1)$$

Isolate Y(k)

$$\boldsymbol{Y(k) = U(k)0.8643 - U(k)0.1555(k - 1) + Y(k)0.2912(k - 1)}$$

Discrete Lag Comp using Tustin @ 500 ms sampling time
Difference Equation





Discrete lag compensator using 100 ms sampling time:

```
tustin_sampling_100_ms =

  0.8106 z - 0.6128
  -----------------
     z - 0.8021
```

Difference equation calculation:

$$\frac{Y(z)}{U(z)} = \frac{0.8106z - 0.6128}{z - 0.8021}$$

Multiply the numerator and denominator by z^{-1}

$$\frac{Y(z)}{U(z)} = \frac{0.8106 - 0.6128z^{-1}}{1 - 0.8021z^{-1}}$$

Cross multiply

$$Y(z)(1 - 0.8021z^{-1}) = U(z)(0.8106 - 0.6128z^{-1})$$

$$Y(z) - Y(z)0.8021z^{-1} = U(z)0.8106 - U(z)0.6128z^{-1}$$

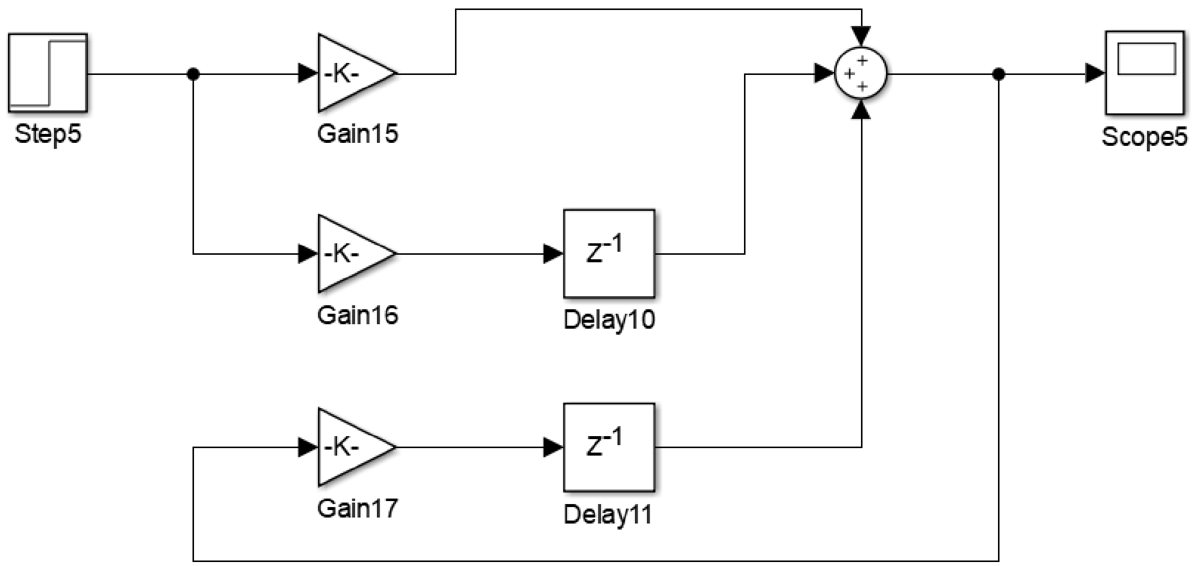
Inverse Z-transform

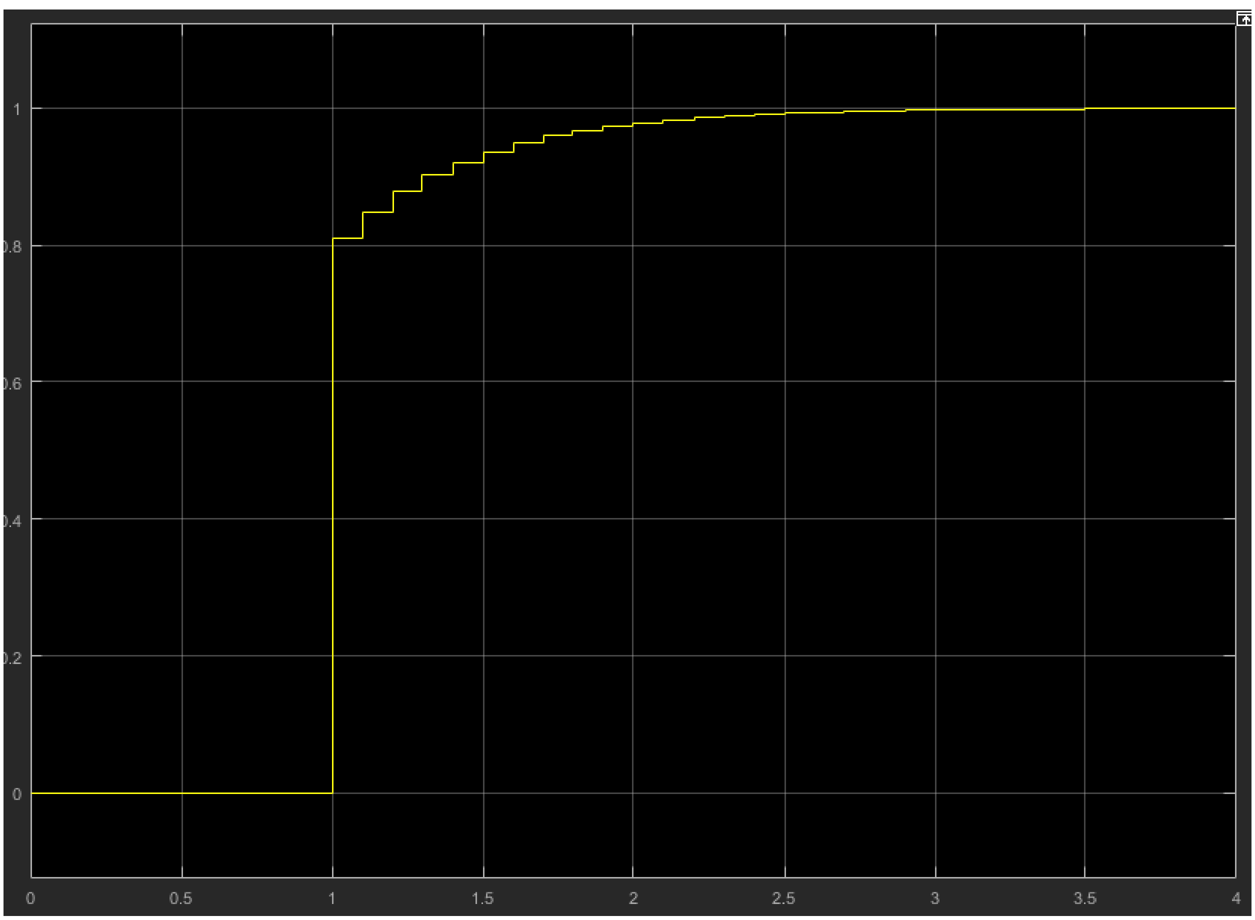
$$Y(k) - Y(k)0.8021(k - 1) = U(k)0.8106 - U(k)0.6128(k - 1)$$

Isolate Y(k)

$$\boldsymbol{Y(k) = U(k)0.8106 - U(k)0.6128(k - 1) + Y(k)0.8021(k - 1)}$$

Discrete Lag Comp using Tustin @ 100 ms sampling time
Difference Equation





Conclusion

Matlab and Simulink have been used to model a DC motor, design a lag compensator for a continuous system and design a discrete lag compensator for a discrete system. The DC motor model has been developed using several different equations pertaining to different aspects of the DC motor.

In part 1 of the project, a field control transfer function, a armature control transfer function and a number of different contents have been derived from equations listed in session #3 of the lab manual. The resulting model when combining all the blocks together yielded the motor response which also represents the plant in general terms for a system. The step input represents an input voltage being used to control the motor. The output of the motor is represented by the shaft rotational velocity which can be integrated to find the shaft displacement. An open loop plant cannot be controlled, and it was observed when a step input signal was applied to the input of the motor the position output did not stabilize to the input value. This is to be expected because the open loop plant is incapable of receiving position feedback which can be used to correct position errors.

In part 2 of the project, a unity feedback was implemented for the motor which aided in stabilizing the shaft position and ensuring the output of the system corresponds to the desired input value. It is observed that when the unity feedback was added to the system, the response of the motor is highly under damped and the overshoot of the system was 42.13%. The output of the system converged to the desired input value, but the overshoot of the system was too high. When looking at the Bode diagram for the closed loop system, the gain margin was 41.9 dB and the phase margin was 43.7 degrees which occurred at the gain crossover frequency of 6.03 rad/s. It has been observed that as the phase margin decreases, the system response becomes increasingly underdamped and the oscillations increase in the response meaning the damping ratio decreases. When the phase margin becomes negative, the system become unstable.

In part 3 of the project, a lag compensator controller was designed which allowed the system response to reach an overshoot of 20%. This was achieved by examining the Bode diagram and adjusting different parameters which include the gain margin, crossover frequency and the phase margin. As mentioned previously, increasing the phase margin will increase the damping ratio of the system which will aid in achieving the desired overshoot value. Given the desired overshoot value, it is possible to calculate the damping ratio and relating it to the required phase margin. In part 3, the damping ratio was calculated to be 0.4559 and the phase margin related to the damping ratio was calculated to be 48.14 degrees. To achieve a larger phase margin value, the cross over frequency will have to be shifted to the left by reducing the gain of the system at the location of the desired phase margin frequency. The lag compensator was successfully designed and then the controller was implemented into the closed loop system, the overshoot of the system did not fulfill the design requirements. In order to reach the overshoot of 20%, a constant was added to the system which help reduce the overshoot further until the desired overshoot was met. The lagging characteristic of the controller is noticeable when comparing the closed loop motor response with the closed loop motor response including the controller. The peak of the controlled response occurs later when compared to the motor response without the controller. A comparison between the bode diagrams was made and it is clear that the controller shifted the crossover frequency to the left which in turn increased the phase margin of the system. The controller was successfully designed because the overshoot requirement was successfully achieved.

In part 4 of the project, the controller designed in part 3 was converted from a continuous controller to a discrete controller. This was achieved using Matlab's c2d function which converted the continuous controller into a discrete controller using different sampling times. The discrete controller was used to drive a continuous plant and the output of the plant was observed for the different sampling times. An observation which can be made when comparing the different sampling times. It is noticeable that as the sampling time decreases, the response of the system follows the continuous response more closely and the errors in the response are less pronounced. For example, when observing the motor response to a controller with a sampling time of 1000 ms, it is possible to notice that the overshoot of the system is near 80% and the shape of the response is oddly shaped when compared to a continuous controller. When the sampling time was decreased to 100 ms, the response more closely followed the original response.

In part 5 of the project, the discrete controllers which have been found for part 4 have been converted into difference equations and implemented in Simulink. It was observed that the discrete transfer functions and the difference equations when implemented in Simulink give the same response.

In part 6 of the project, a discrete controller must be designed for a discrete plant using the frequency domain design method. It is noted in section 13.6 of the textbook that using the frequency domain method as seen in part 3 of the project is not directly appliable for discrete systems. In order to fulfill the requirements, the frequency response for the continuous response was compared to the zoh discretization method. It has been observed that the frequency response at higher frequencies does not follow the continuous response. The Tustin method is another discretization method which has been used. It has been observed that the frequency response for a controller which has been discretized using the Tustin method followed the continuous frequency response more closely that the zoh method. In both cases, the lower the sampling time used, the closer the response behaved like the continuous response. Given the Tustin method produced a discrete controller that behaves closer to the continuous response, the plant has also been converted to discrete using the Tustin method. When the discrete system's response has been measured, it is observable that the smaller sample time followed the continuous response more closely than the larger sample times.

In part 7 of the project, the discrete controllers which have been found for part 6 using the Tustin method have been converted into difference equations and implemented in Simulink. It was observed that the discrete transfer functions and the difference equations when implemented in

Simulink give the same response. It was observed the controller responses differed slightly from the zoh method controllers which have been observed in part 5.

Reference

1. *MECH 473 Control System Design Project Manual*, Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Fall 2020.

2. Chen, C. (2006). *Analog and digital control system design: Transfer-function, state-space, and algebraic methods*. Place of publication not identified: Oxford University Press.