Assignment 2

Runmotorsim.mlx

Alex Curtis EENG350 2/6/22

This script runs a simulation of a motor with bearing friction and plots the results.

Required file: motorsim.slx

Motor Parameters

```
Ra = 1;     % armature resistance   [Ohms]
Kt = 0.5;   % motor torque constant [Nm/A]
Ke = 0.5;   % back emf constant     [Vs/rad]
J = 0.05;   % Load inertia          [Nm^2]
b = 0.5;    % damping               [Nm/s]
```

Run Simulation

This simulation applies a rectifed sinusoidal voltage to a DC motor model, with the output as the angular position in radians.

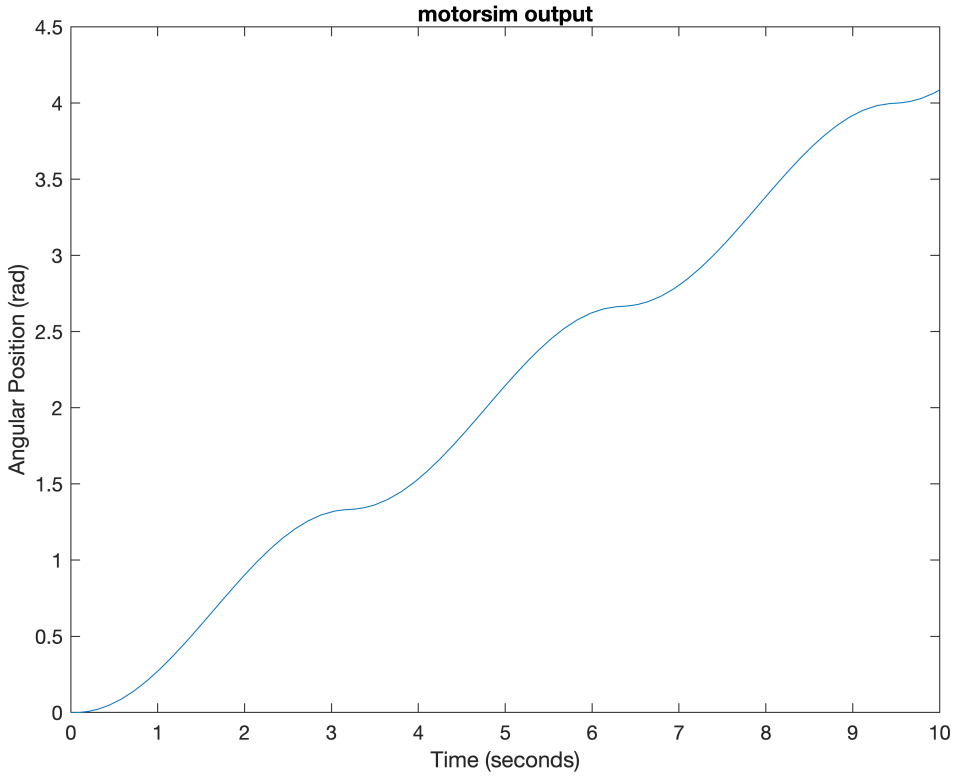
Open the block diagram so it appears in the documentation when published. *Make sure the block diagram is closed before running the publish function*

A Plot of the results

We see that the motor rotates in the positive direction, with some oscillations due to the varying input

Plot of motorsim output

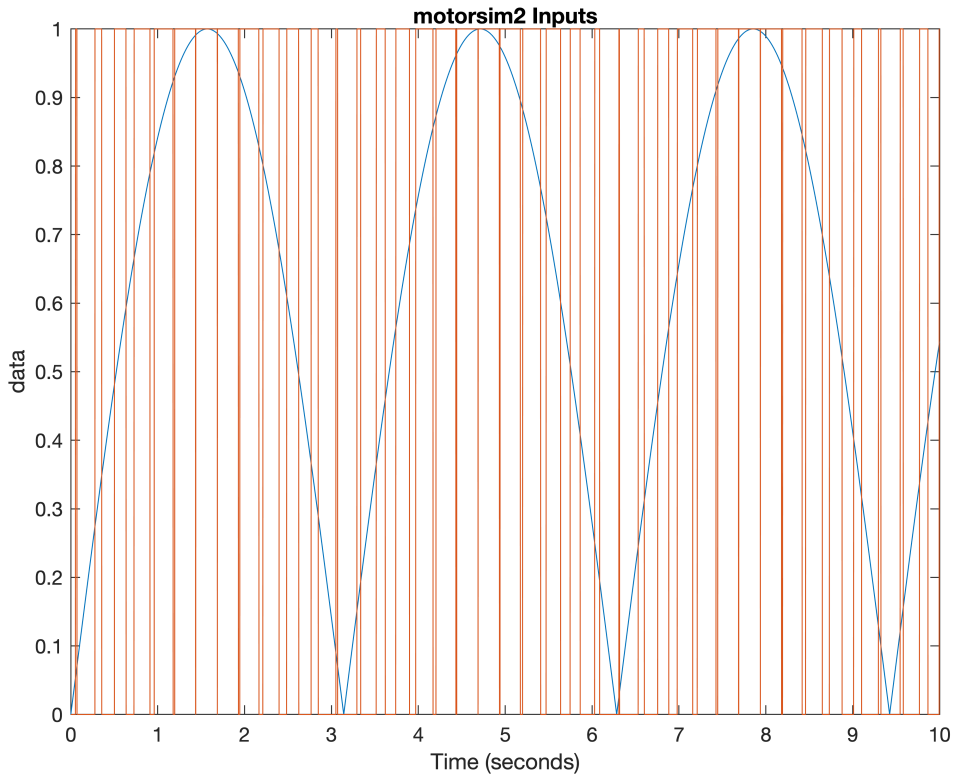
```
open_system('motorsim')
out1 = sim('motorsim');
figure
plot(out1.simout)
title('motorsim output')
ylabel('Angular Position (rad)')
```



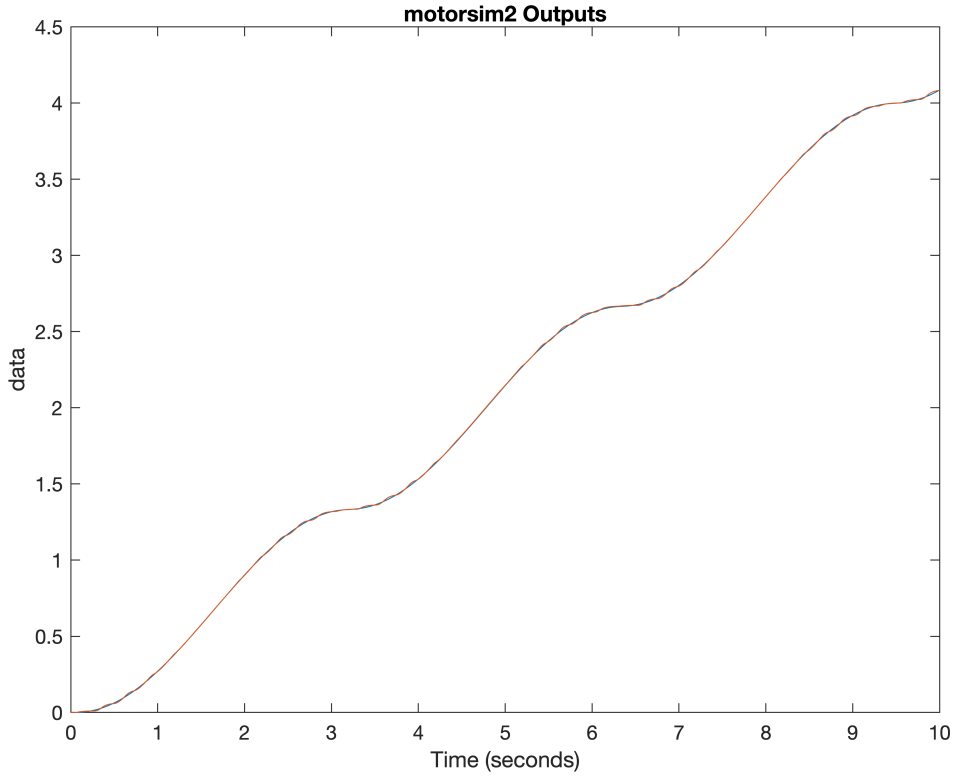
Plot of motorsim2 inputs and outputs

Shows the effect of a continuous PWM input

```
open_system('motorsim2')
out2 = sim('motorsim2');
figure
plot(out2.inputs)
title('motorsim2 Inputs')
```



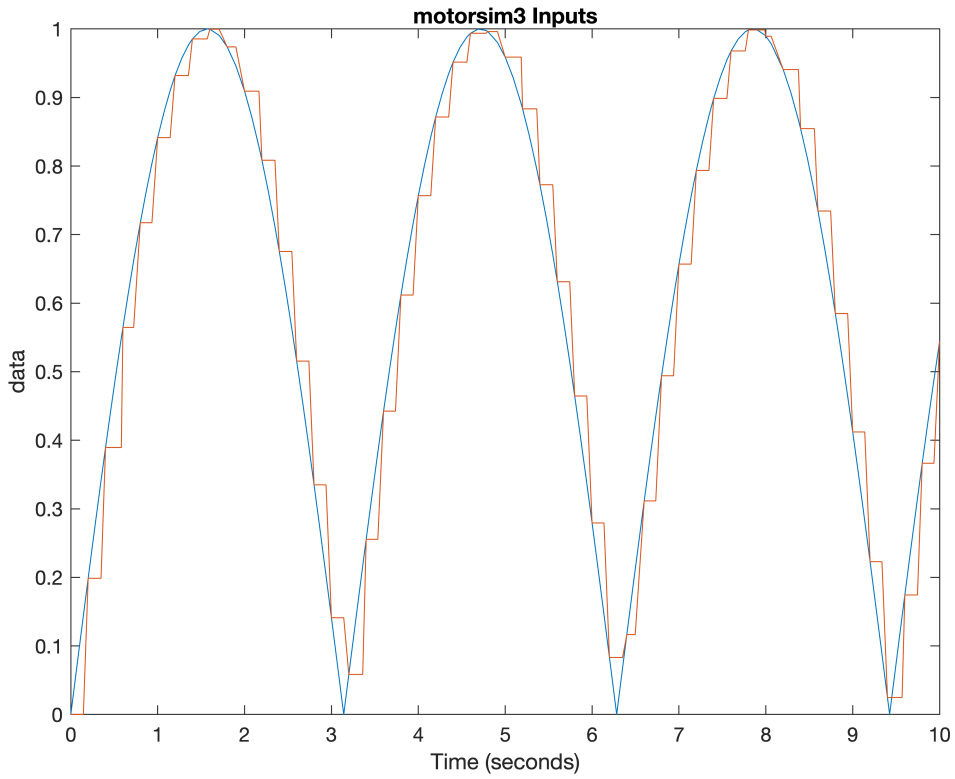

```
figure
plot(out2.outputs)
title('motorsim2 Outputs')
```



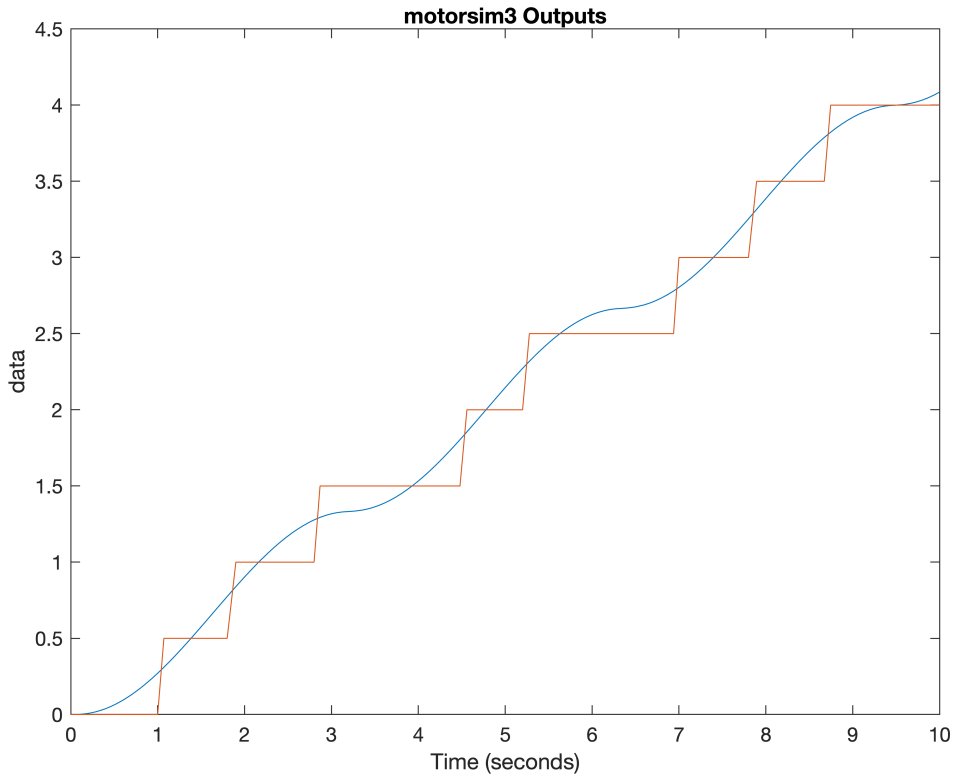
Plot of motorsim3 inputs and outputs

Shows the effect of a quantized input

```
open_system('motorsim3')
out3 = sim('motorsim3');
figure
plot(out3.inputs)
title('motorsim3 Inputs')
```




```
figure
plot(out3.outputs)
title('motorsim3 Outputs')
```



Finding the transfer function

Using slLinearizer and getIOTransfer, MATLAB will find the closed loop transfer function of the motor for me!

motorsim5 is the "Motor with inertial load" subsystem

I was using the tf for position, not for velocity

```
open_system('motorsim5')
sllin = slLinearizer('motorsim5');
addPoint(sllin,{'Va','pos','vel'});

posTf = getIOTransfer(sllin,'Va','pos');
velTf = getIOTransfer(sllin,'Va','vel');

pos = tf(posTf)
```

```
pos =

  From input "Va" to output "pos":
      10
  ----------
  s^2 + 15 s

Continuous-time transfer function.
```

```
posNum = tf(posTf).Numerator;
posNum = posNum{1};
posDen = tf(posTf).Denominator;
posDen = posDen{1};
```

```
tf(velTf)
```

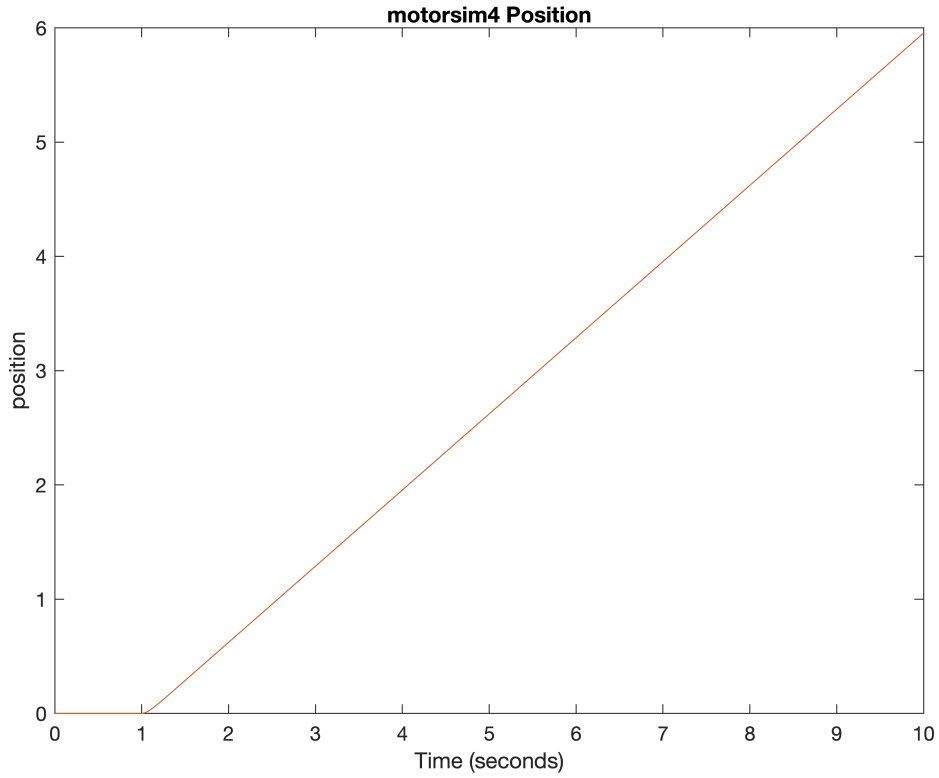
```
ans =

  From input "Va" to output "vel":
    10
  ------
  s + 15

Continuous-time transfer function.
```

```
velNum = tf(velTf).Numerator;
velNum = velNum{1};
velDen = tf(velTf).Denominator;
velDen = velDen{1};
```

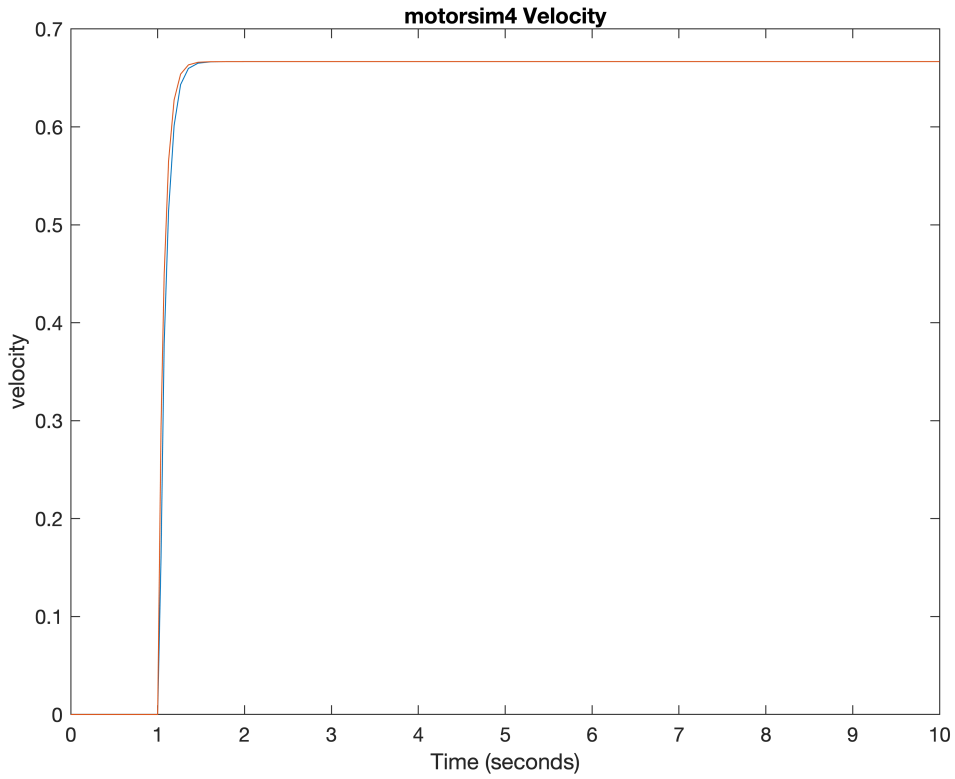
Plot of motorsim4 outputs

Red is the transfer function response, blue is the motor response

```
open_system('motorsim4')
out4 = sim('motorsim4');
figure
plot(out4.position)
title('motorsim4 Position')
```

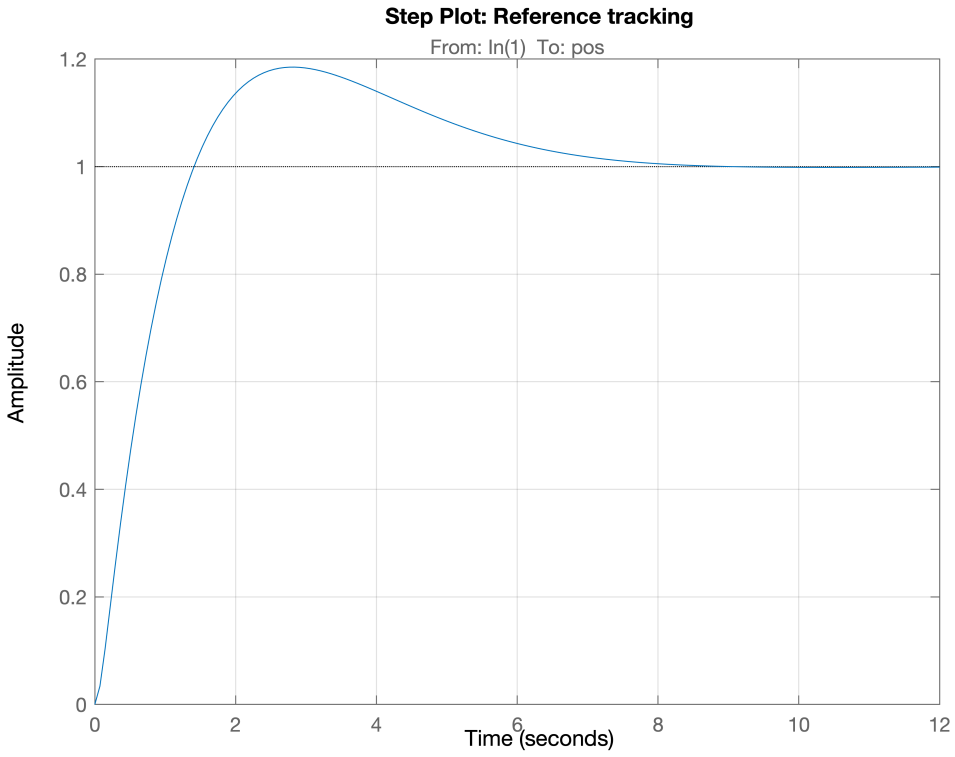



```
figure
plot(out4.velocity)
title('motorsim4 Velocity')
```



Creating a PI controller

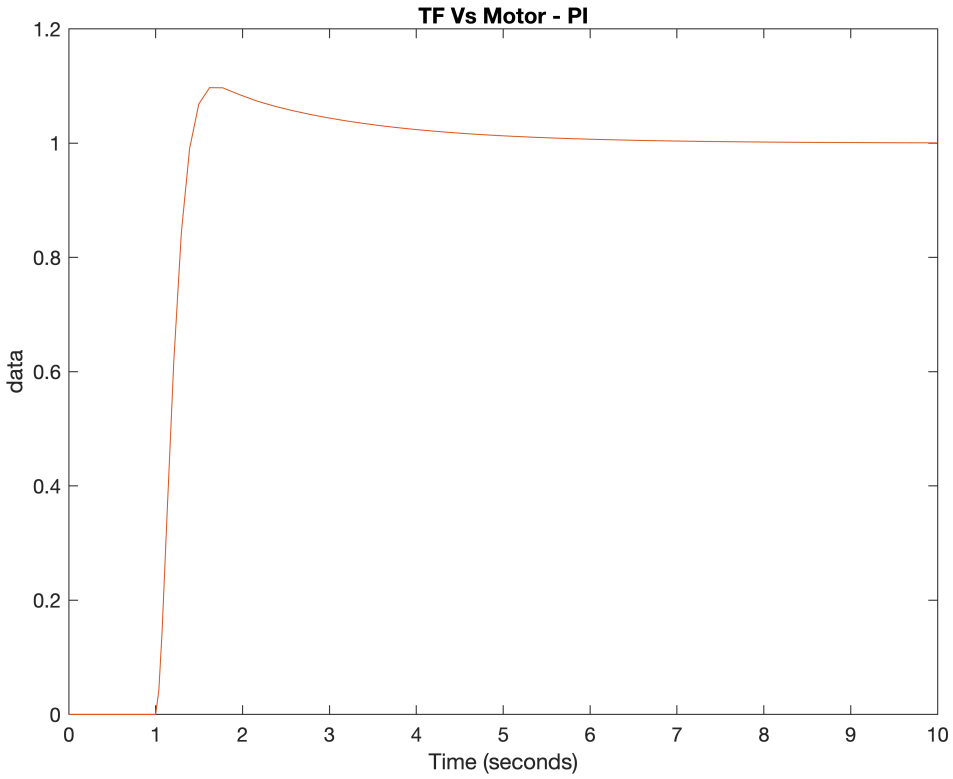
```
% Convert Response Time to Bandwidth
% Bandwidth is equivalent to 2 divided by the Response Time
wc = 2/1.58545;

% Convert Transient Behavior to Phase Margin
% Phase Margin is equivalent to the Transient Behavior multiplied by 100
PM = 100*0.6617;

% Define options for pidtune command
opts = pidtuneOptions('PhaseMargin',PM);

% PID tuning algorithm for linear plant model
[C,pidInfo] = pidtune(posTf,'PI',wc,opts);

% Clear Temporary Variables
clear wc PM opts

% Get desired loop response
Response = getPIDLoopResponse(C,posTf,'closed-loop');

% Plot the result
stepplot(Response)
title('Step Plot: Reference tracking')
grid on
```




```
% Display system response characteristics
disp(stepinfo(Response))
```

```
        RiseTime: 1.0150
    TransientTime: 6.8958
     SettlingTime: 6.8958
      SettlingMin: 0.9034
      SettlingMax: 1.1851
        Overshoot: 18.5054
       Undershoot: 0
             Peak: 1.1851
         PeakTime: 2.8380
```

```
% Clear Temporary Variables
clear Response
open_system('motorsim6')
open_system('motorsim7')
tfOut = sim('motorsim6');
motorOut = sim('motorsim7');
figure
plot(tfOut.position)
hold on
plot(motorOut.position)
hold off
title('TF Vs Motor - PI')
```


TF Vs Motor - PI
1.2
1
0.8
0.6
0.4
0.2
0
data
0
1
2
3
4
5
6
7
8
9
10
Time (seconds)