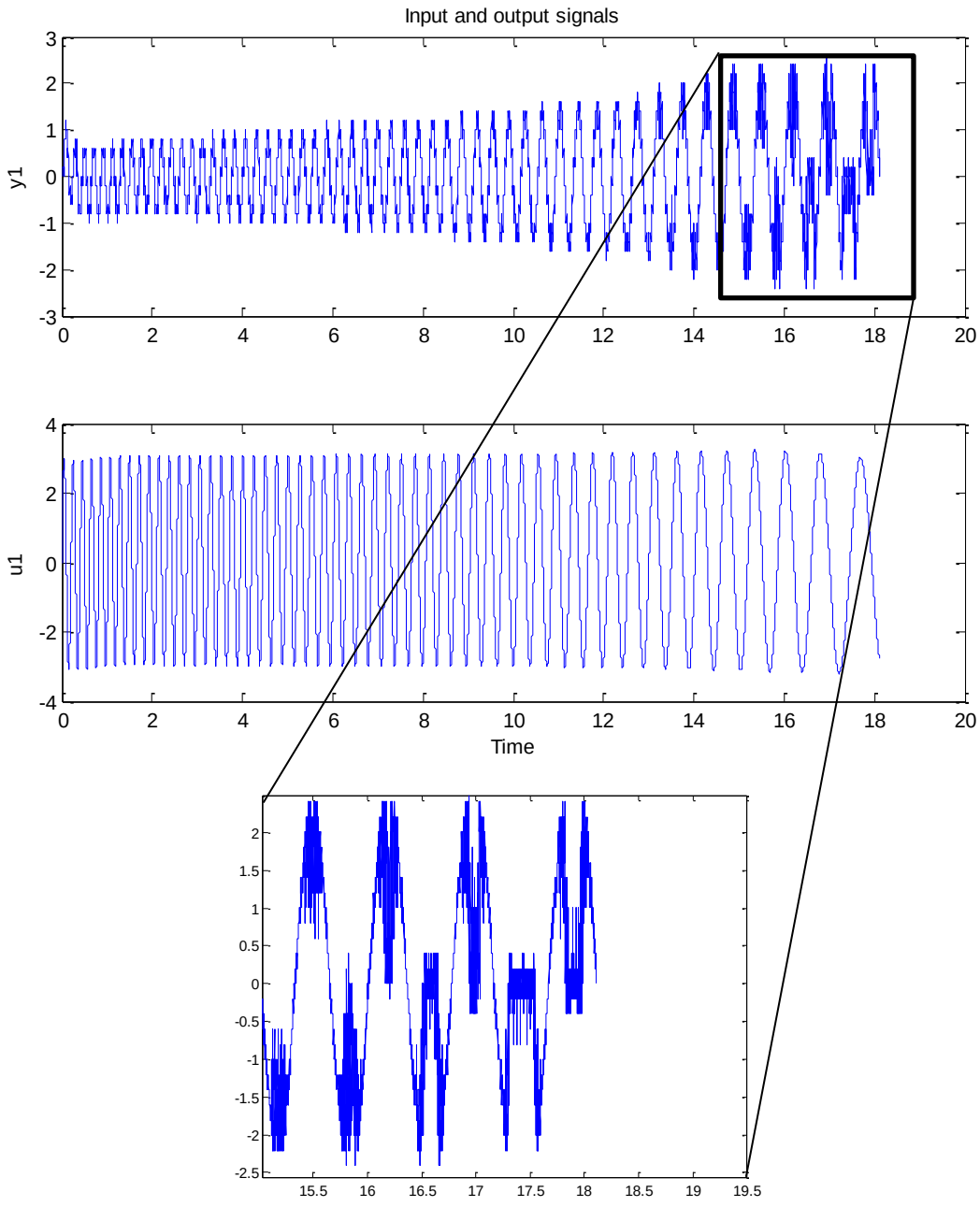
Robotics Homework 4

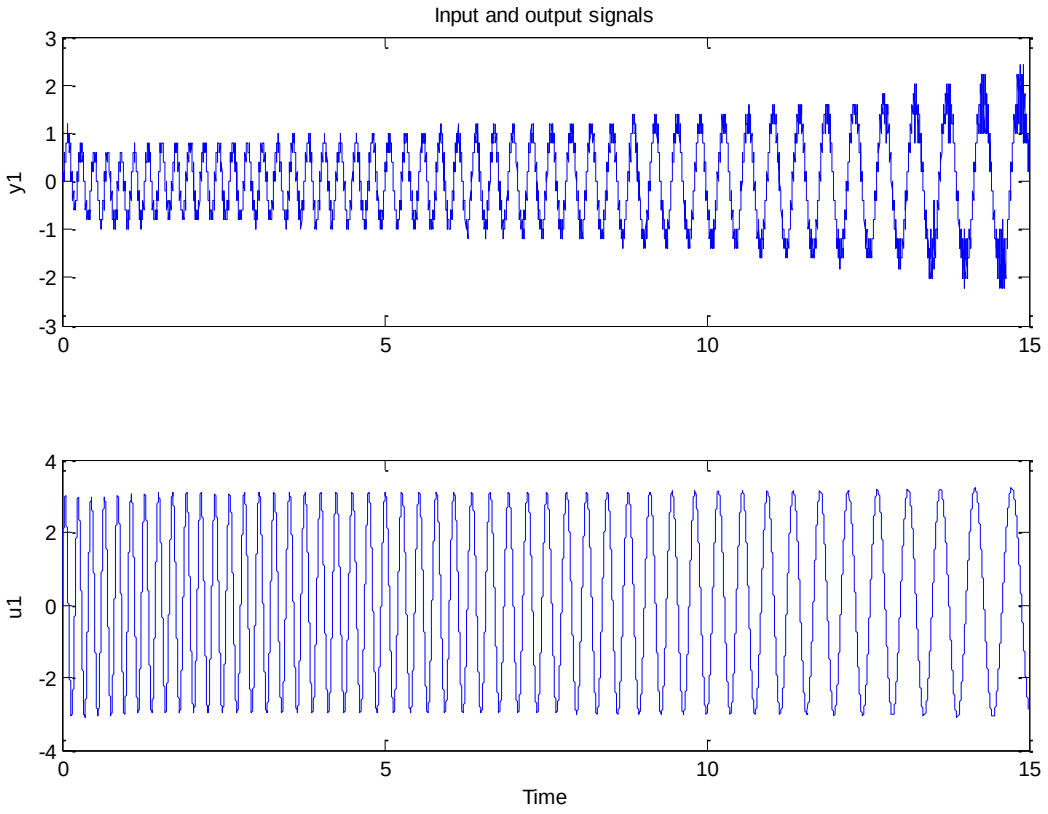
Poya Khalaf [redacted]

Obtain an estimated transfer function from amplifier input voltage to joint velocity using the data collected during class and Matlab's System Identification Toolbox.

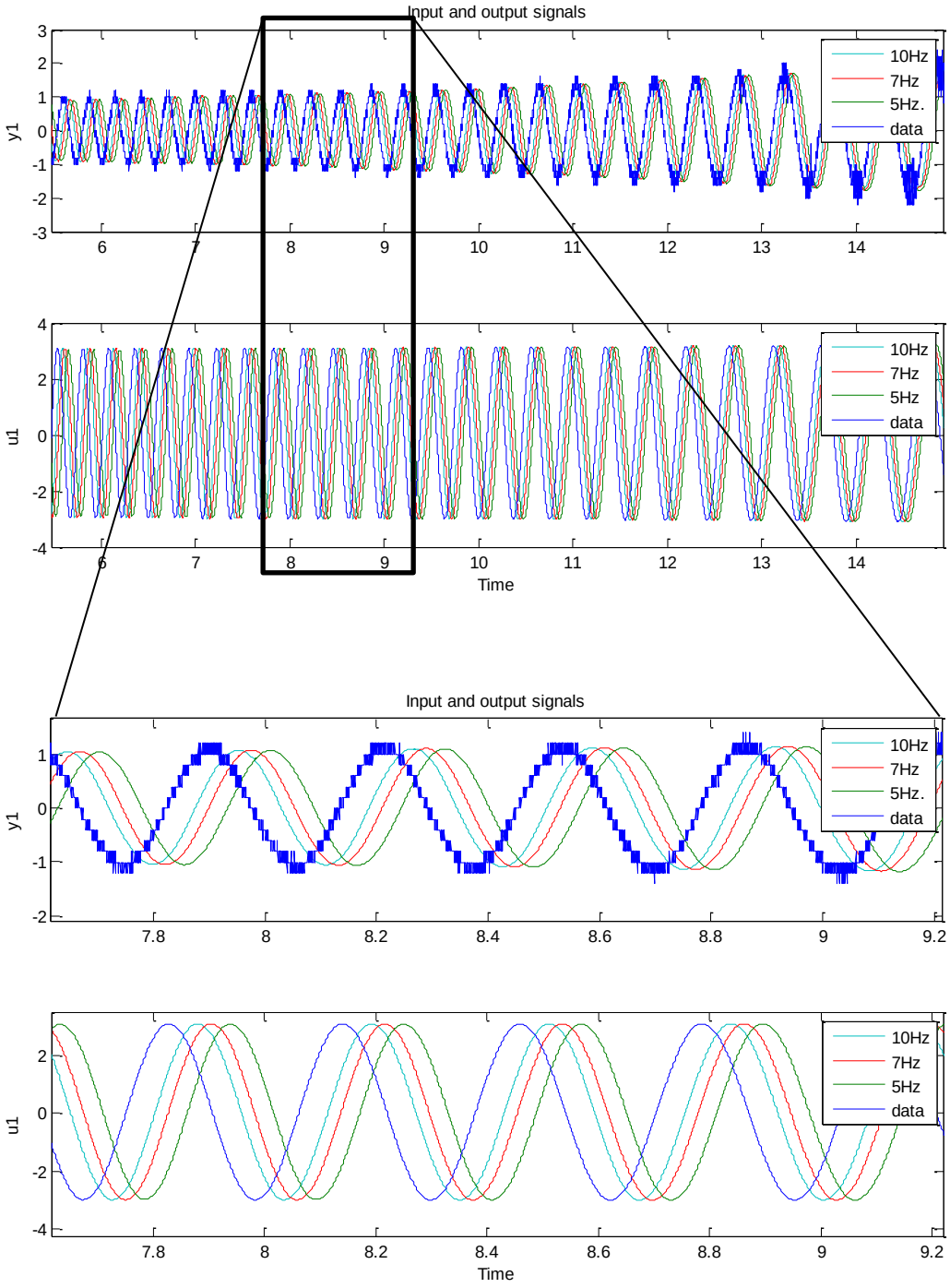
After loading the chirp data and running the system identification tool box, we import the data into the system identification tool box. The input data is u and the output data is q1dot.we also set the sampling time to Ts. The figure below plots the input and output data.



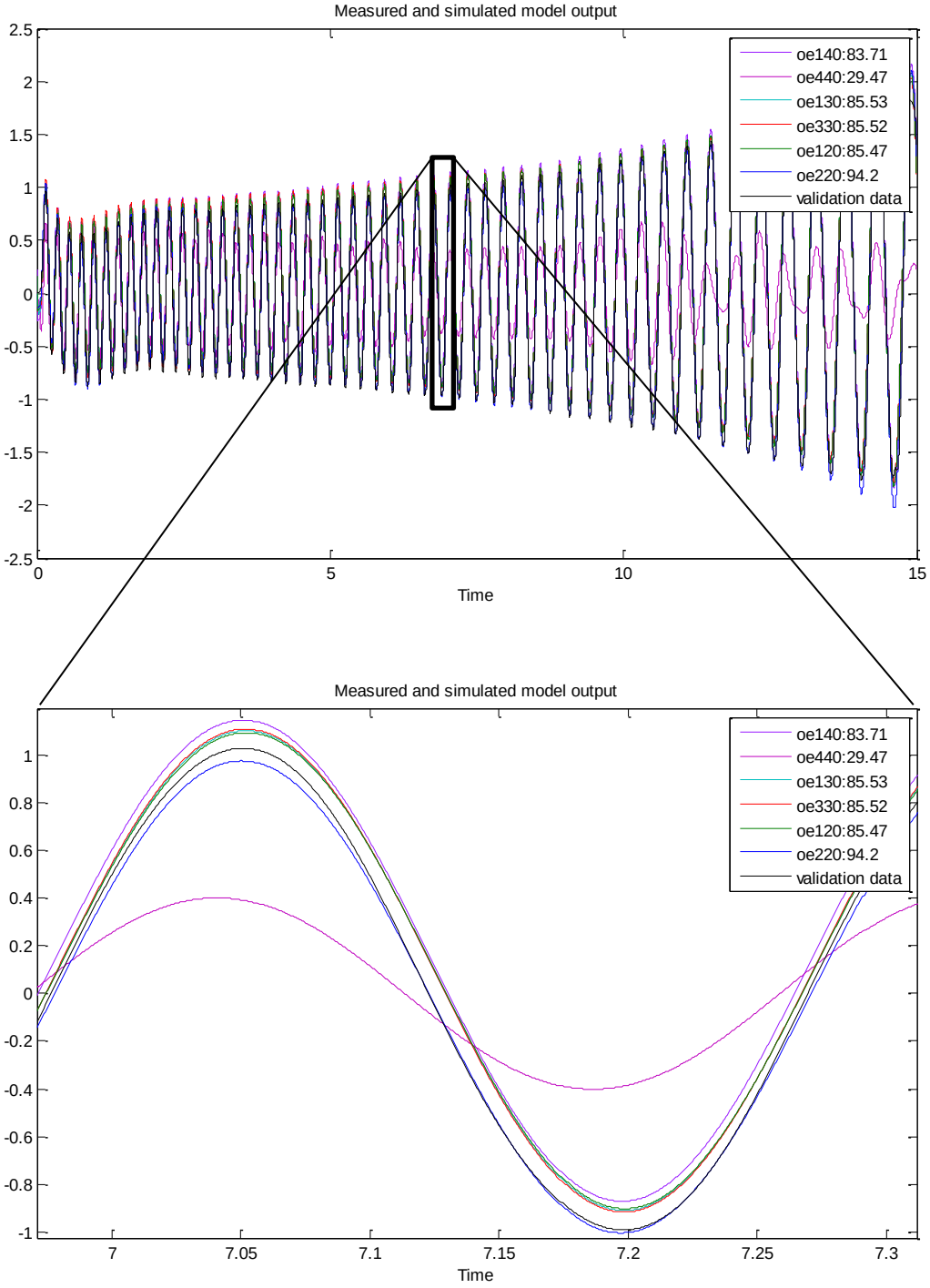
A carful look at the end of the data shows some behavior that even if it is the actual behavior of the system it cannot be captured by a transfer function model that we aim to identify. For this reason we truncate the end of the data. The figure below shows the truncated data that we will use for the system identification.



In the next step we filter the data to remove unnecessary noise. For this purpose we low pass filter the data with a cutoff frequency of 5, 7 and 10 Hz. the figure below plots the results of filtering the data. From the figure below we choose a cutoff frequency of 10Hz.


Input and output signals
y1
u1
Time
10Hz
7Hz
5Hz.
5Hz
data
Input and output signals
y1
u1
Time
10Hz
7Hz
5Hz.
5Hz
data

In the next step we attempt to fit a transfer function model with different number of poles and zeros using the output error structure. The figure below shows these results.



The OE220 model has the highest score however by converting this model from discrete to continuous we see that it has a right half zero hand two complex conjugate poles. This does not seem to be the right model for the motor.

```
>> zpk(d2c(oe220))

ans =

  From input "u1" to output "y1":
  0.0034689 (s+2001) (s-0.04857)
  ------------------------------
     (s^2 + 2.41s + 3.634)

Continuous-time zero/pole/gain model.
```

The next highest score is for the OE130 model. Converting this model to continuous time we have:

```
>> zpk(d2c(oe130))

ans =

  From input "u1" to output "y1":
  1.3142e-05 (s+1743) (s^2 + 1996s + 4.705e06)
  --------------------------------------------
     (s+4.517) (s^2 + 93.75s + 1.523e04)

Continuous-time zero/pole/gain model.
```

This model has two fast complex conjugate poles which can be neglected. Also the model has three fast zeros that can be neglected. After these reductions and fixing the dc gain the model becomes:

```
oe130r =

    7.077
  ---------
  s + 4.517

Continuous-time transfer function.
```

The next model is OE330.The transfer function of this model in continuous time is as below. This model also has right half zeros which is not compatible with our system.

```
>> zpk(d2c(oe330))

ans =

  From input "u1" to output "y1":
  0.0013018 (s+2048) (s^2 - 88.79s + 1.757e04)
  -------------------------------------------
       (s+4.411) (s^2 + 1.799s + 6771)

Continuous-time zero/pole/gain model.
```

The next model is OE120. The transfer function of this model in continuous time is as below.

```
>> zpk(d2c(oe120))

ans =

  From input "u1" to output "y1":
  0.00026833 (s^2 + 3107s + 4.161e06)
  ----------------------------------
         (s+154) (s+4.632)

Continuous-time zero/pole/gain model.
```

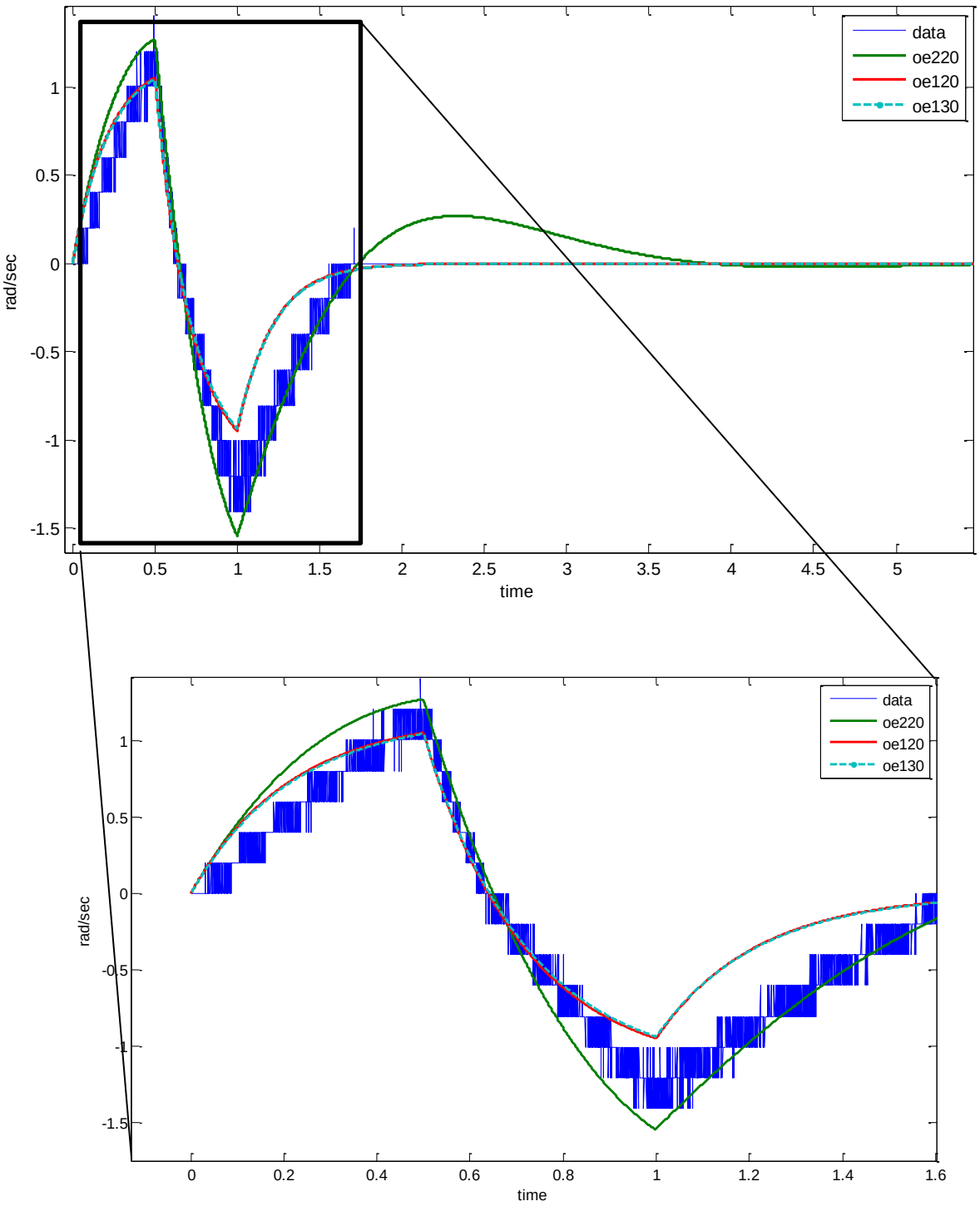
This model has one fast pole and two fast zeros that can be neglected. The reduced model is:

```
oe120r =

    7.25
  --------
  s + 4.63

Continuous-time transfer function.
```

The figure below compares the pulse response of the OE220, OE130, OE120 models.



It can be seen from the above figure that the OE220 model matches the data very good in the pulse phase but has an undesirable over shoot. Also it is seen that the OE120 model and the OE130 model have very close responses. We will choose the OE130 model for the remaining of this HW.

Design a controller (classical compensator) for your estimated transfer function for the following

specifications relative to a reference step input of 45 degrees:

1. Zero (or almost zero) overshoot.

2. Zero steady-state error.

3. The fastest settling time that can be obtained under the restriction that |u| < 10 V.

The open loop transfer function from voltage to velocity is:

```
oe130r =

   7.077
 ---------
 s + 4.517

Continuous-time transfer function.
```

The open loop transfer function from voltage to becomes:

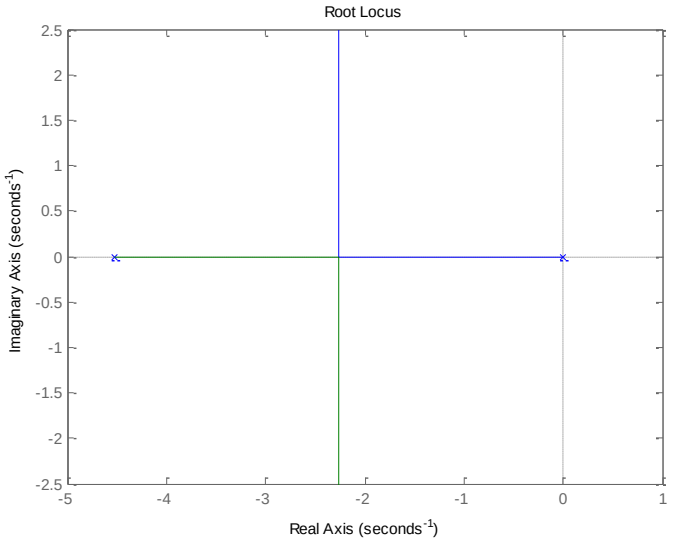
```
>> G=tf([7.077],[1 4.517 0])

G =

    7.077
 -------------
 s^2 + 4.517 s

Continuous-time transfer function.
```

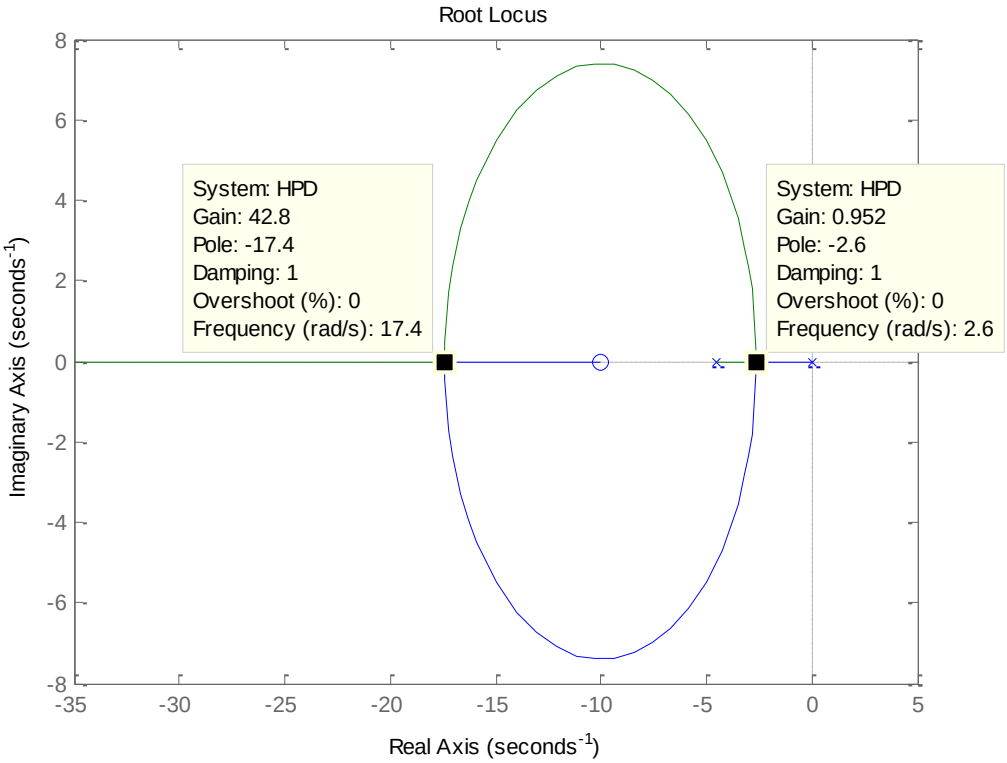
The root locus of this system is plotted below.



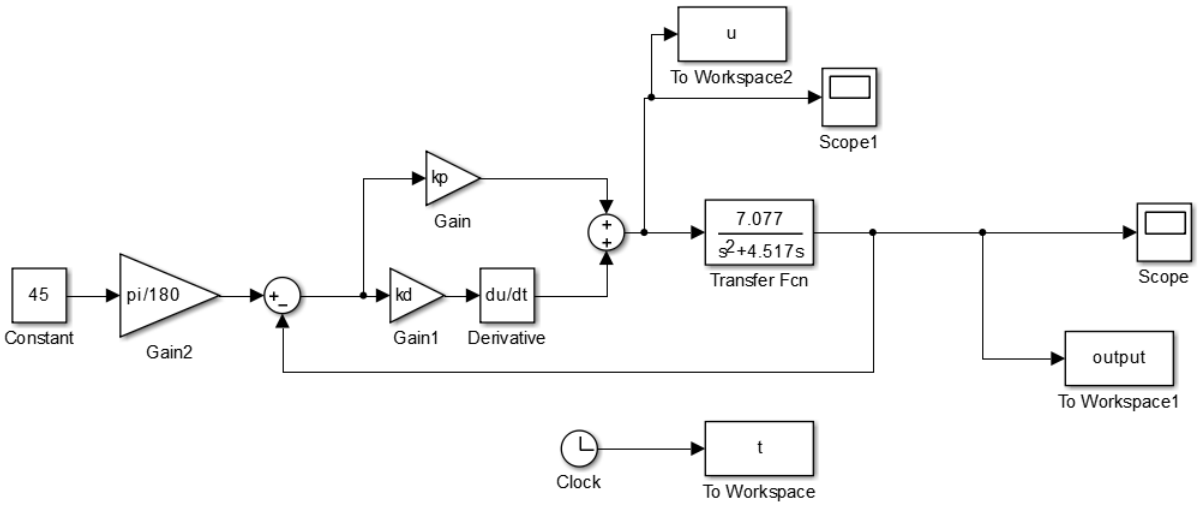
Adding a PD controller the open loop transfer function becomes.

$$G_{PD} = \frac{k_p 7.077(1+\frac{k_d}{k_p}s)}{s^2 + 4.517s}$$

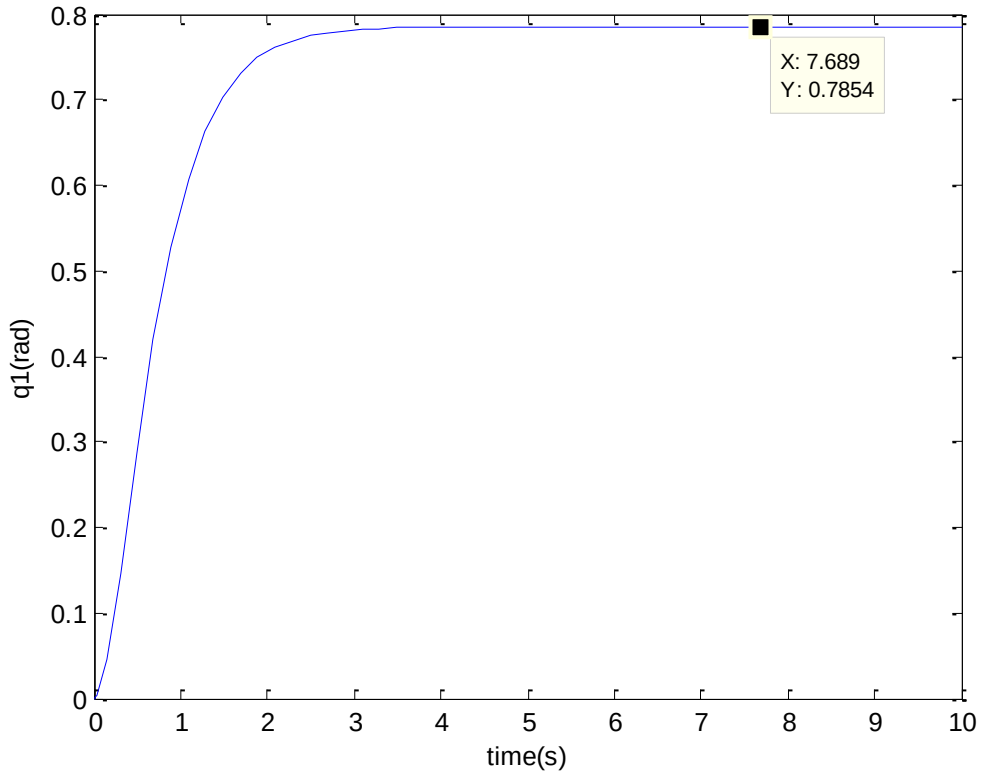
Selecting $\frac{k_d}{k_p} = 0.1$ the root locus of this system becomes:



There are two gains that produce zero overshoot. Also this system with the PD controller produces zero steady state error. This can be seen from the closed loop transfer function.in conclusion we check these two gains to see which one has the least settling time. For this purpose a Simulink model is crated as below:



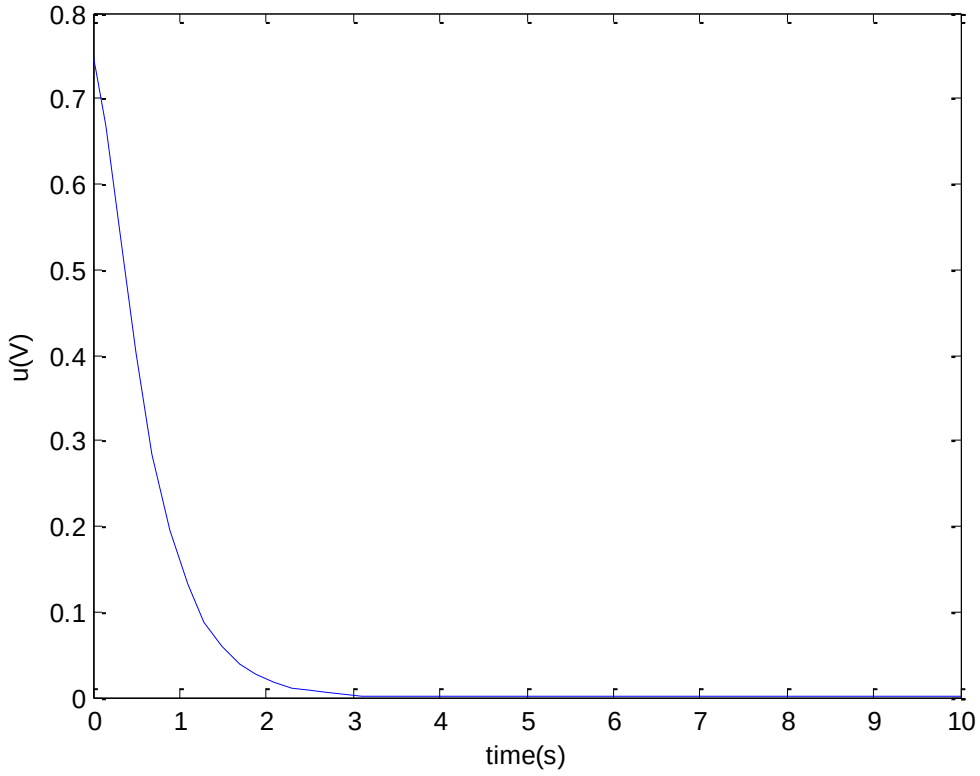
Selecting kp=0.952 and kd=0.1kp the step response of the system is obtained.



It is seen that the response has zero over shoot and zero steady state error.to check the settling time we do as below:

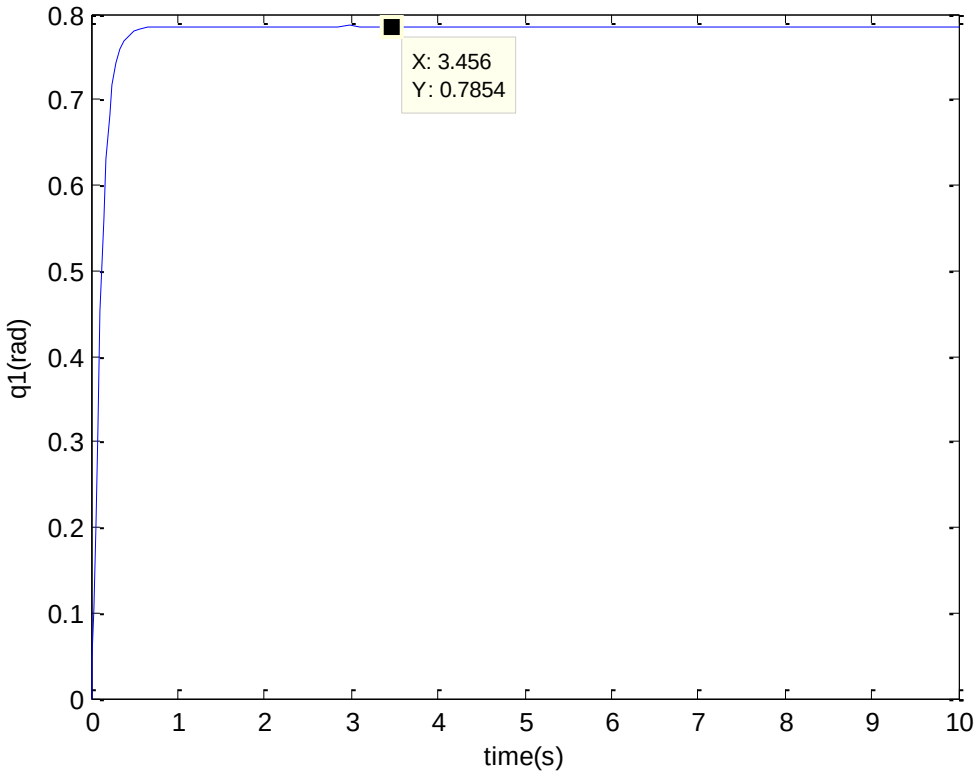
```
>> stepinfo(output,t,0.7854)
ans =
       RiseTime: 1.3155
   SettlingTime: 2.2950
    SettlingMin: 0.7306
    SettlingMax: 0.7854
      Overshoot: 0
     Undershoot: 0
           Peak: 0.7854
       PeakTime: 10
```

So the settling time is equal to 2.29 sec. Now we check the controllers output to see if it has satisfied the required constraint.



We see that setting kp=0.952 and kd=0.0952 satisfies the zero over shoot and zero steady state error constraints. Also is satisfies the maximum voltage constraint and it produces a settling time of 2.29 sec.

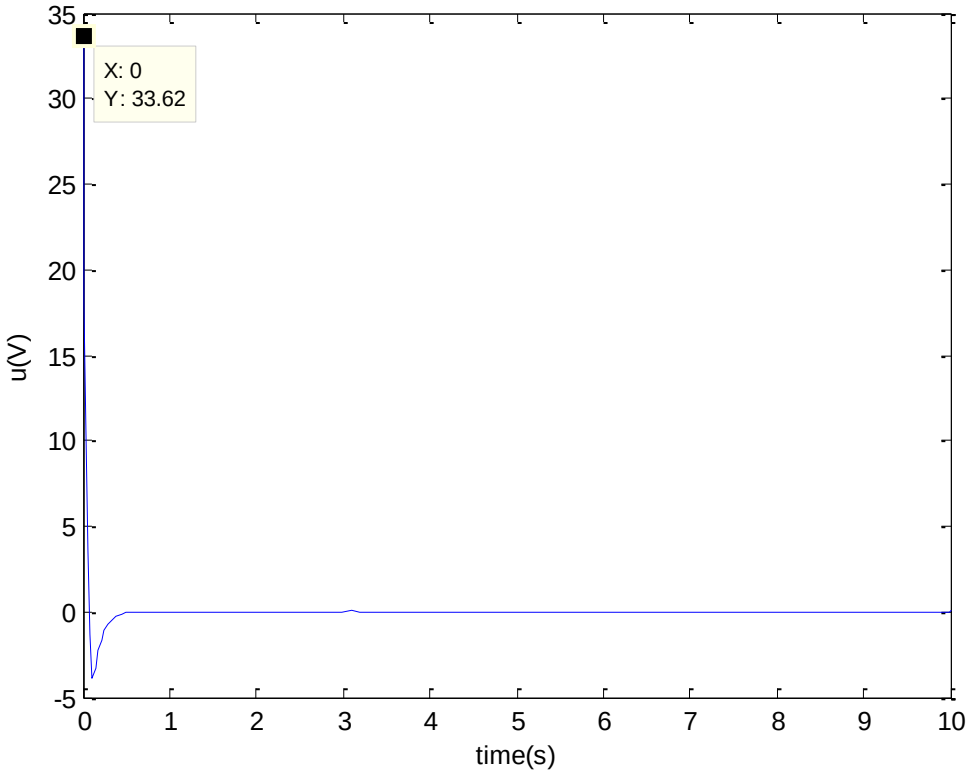
In the next step we set kp=42.8 and kd=0.1kp.the step response is obtained as below



We see that the over shoot and steady state error are zero. We check the settling time:

```
>> stepinfo(output,t,0.7854)
ans =
       RiseTime: 0.2126
   SettlingTime: 0.3903
    SettlingMin: 0.7172
    SettlingMax: 0.7860
     Overshoot: 0.0813
    Undershoot: 0
          Peak: 0.7860
      PeakTime: 2.9812
```

We see that the settling time is 0.39 sec. We check the controller output:



We see that it does not satisfy our constraint. Therefore the best settling time achieved was 2.29 sec.

3: Find a state-space representation of the estimated transfer function having q1 and q˙1 as states (find matrices A, B, C, D). Then:

1. Design a linear state feedback controller (u = Pr−Kx) using linear quadratic optimal design for K. Choose Q and R so that the response time is similar to what you obtained with the classical compensator. The control input remains between -10 and 10. (Use example4.mdl as an initial template and iterate with simulations.). Calculate the required value of P and verify that the output settles at 45 degrees without offset.

The open loop transfer function of the estimated system is :

$$\frac{q(s)}{u(s)} = \frac{7.077}{s^2 + 4.157s}$$

This corresponds to the following differential equation

$$\ddot{q} + 4.157\dot{q} = 7.077u$$

The state space representation for this system is:

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -4.157 \end{bmatrix} x + \begin{bmatrix} 0 \\ 7.077 \end{bmatrix} u$$

$$y = x_1$$

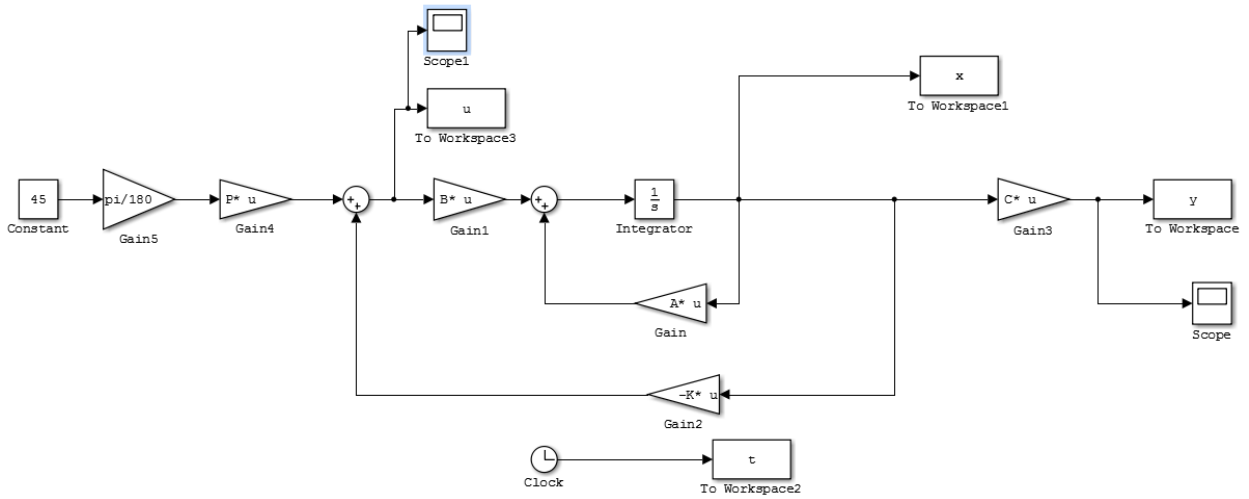
$$A = \begin{bmatrix} 0 & 1 \\ 0 & -4.157 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 7.077 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$D = 0$$

We modify the example4.mdl file as below:



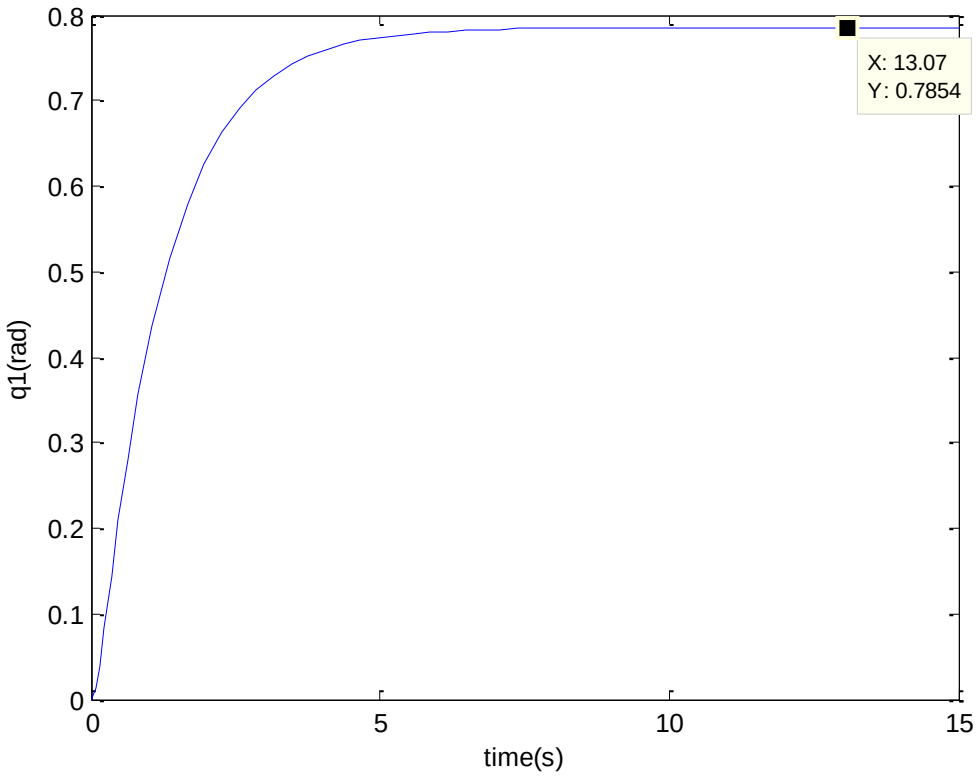
By setting the matrices Q and R we can use the LQR command to design the state feedback:

```
>> Q=eye(2)
Q =
    1    0
    0    1
>> R=1
R =
    1
>> K=lqr(A,B,Q,R)
K =
   1.0000    0.6884
```

After designing the state feedback we can calculate P:

```
>> P=1/(D-(C-D*K)*((A-B*K)^-1)*B)
P =
   1.0000
```

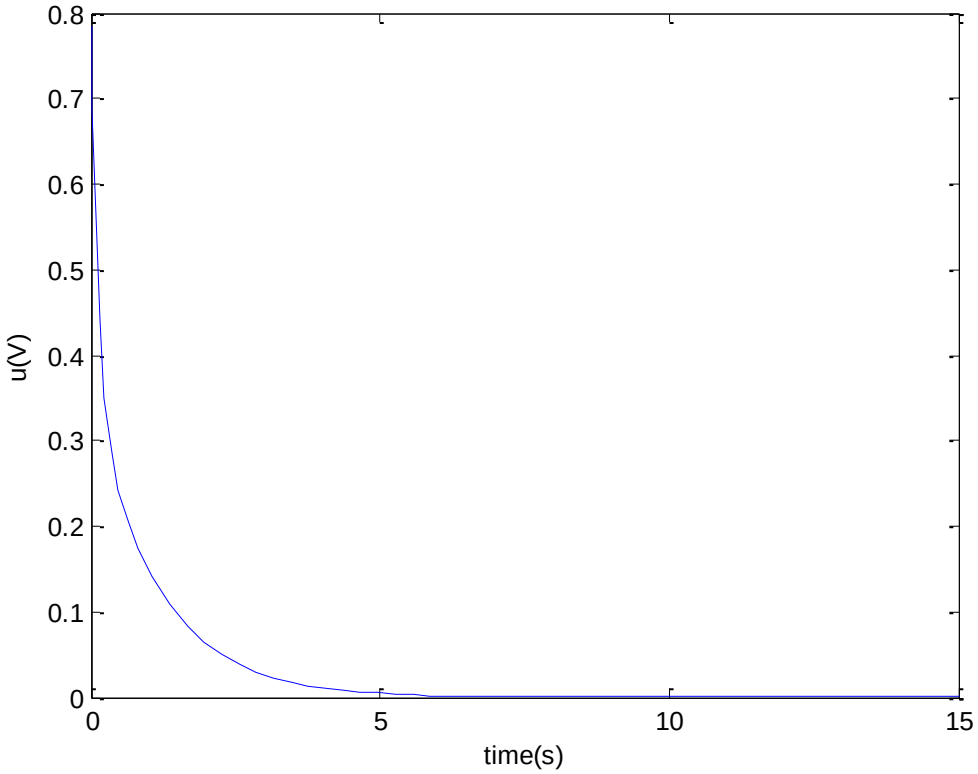
Running the Simulink simulation file the step response of the system with a state feedback controller is obtained:



We see that the over shoot and the steady state error are zero. We check the settling time:

```
>> stepinfo(y,t,0.7854)
ans =
       RiseTime: 2.5657
   SettlingTime: 4.6441
    SettlingMin: 0.7122
    SettlingMax: 0.7854
      Overshoot: 0
     Undershoot: 0
           Peak: 0.7854
       PeakTime: 15
```

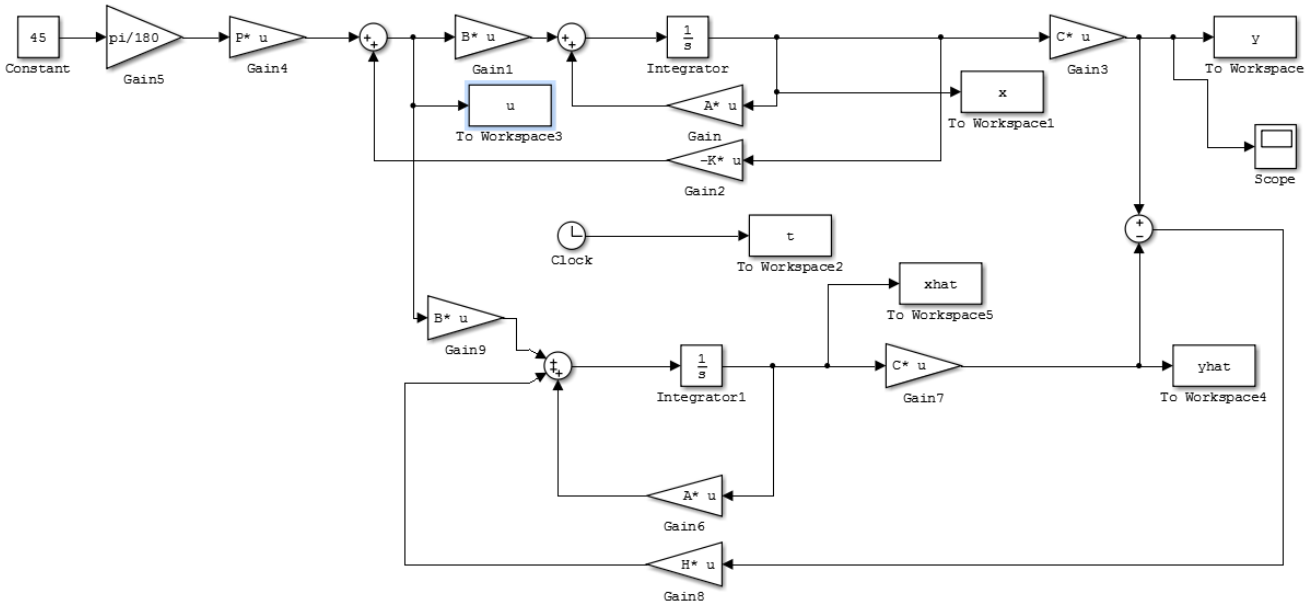
The settling time is 4.6 sec which is higher than what was obtained from the PD controller.we che the controller output:



We see that the voltage remains within the constraint.

2. Design and tune a linear state estimator (Luenberger observer). Add it to the simulation and check that the actual states are being tracked by the estimator.

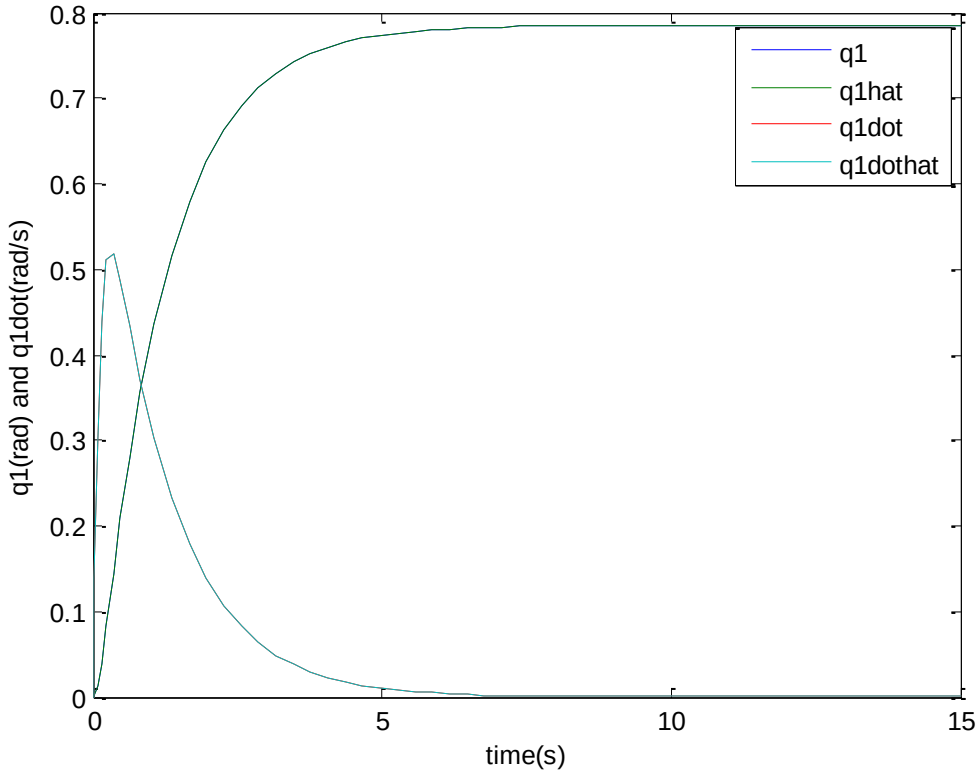
In the next step we design an observer for this system. We modify the Simulink file as below:



We design the observer as below:

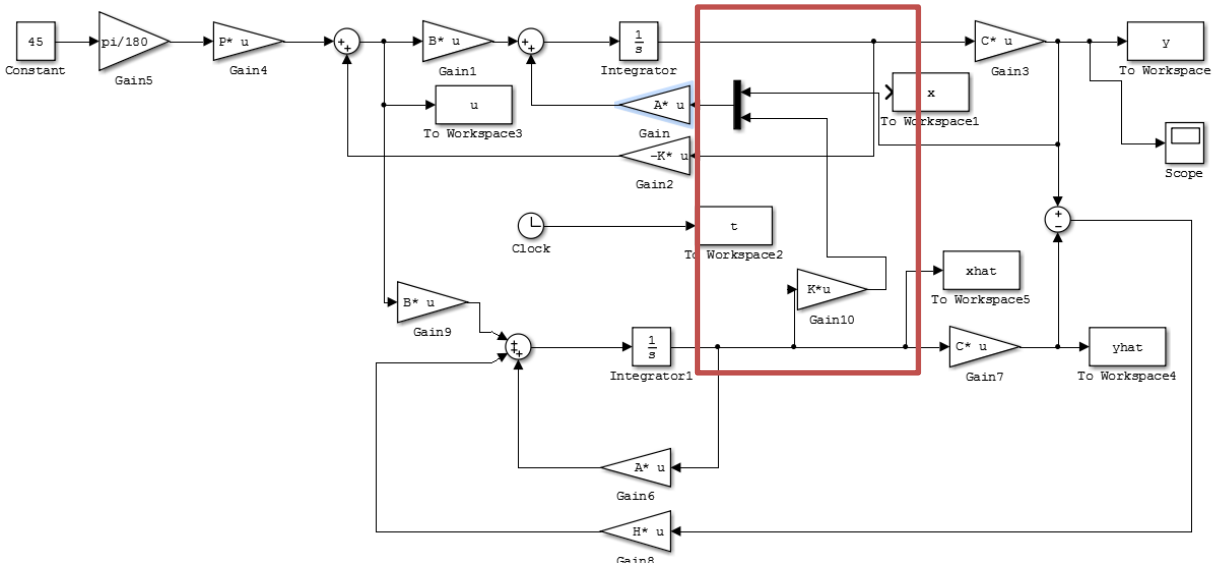
```
>> Q=2*eye(2)
Q =
    2    0
    0    2
>> R=1
R =
    1
>> H=lqr(A',C',Q,R)
H =
   1.4442    0.0429
>> H=H'
H =
   1.4442
   0.0429
>>
```

The figure below shows that the stats are being tracked correctly by the observer.



3. Use the velocity estimate in place of the actual velocity for calculating the control. Re-tune controller and observer if necessary, to regain the original performance.

The Simulink file is modified so that the position is directly fed back to the controller but the velocity is estimated by the observer and fed into the controller:



The output of the system in this situation is as below: