Hacettepe University - Computer Engineering

Advanced Robust Control MMÜ 749

Homework-5
H∞ Controller Design

Student Name Ayça KULA
Student ID: 202237285

Spring 2022

1 System description

The model is simple where a force u(t) is acting on a mass m. So the the plant is basically described as

$$\ddot{x} = u(t)/m$$

The measurement will be either position or speed. Therefore, in order to have a measurement of speed I have to add a derivative term. For the two problems if I have an measurement of position transfer function is defined as in algorithm 1. However, for speed measurement I have defined a new plant as in 2.

```
% Define Plant
num = 1;
den = [1 1e-3 1e-6];
% Nominal plant (Transfer function with no uncertainty)
G0 = tf(num,den);
```

Algorithm 1: Plant for measurement of Position

```
% Define Plant
num = 1;
den = [1 1e-3];
% Nominal plant (Transfer function with no uncertainty)
G0 = tf(num,den);
```

Algorithm 2: Plant for measurement of Speed

2 H∞ controller design with unstructured uncertainty approach

2.1 Mass uncertainty

Each uncertain parameter is bounded within some region. Therefore the mass m is defined within a known interval:

$$m_{\min} \leq m_p \leq m_{\max}$$

where m_p is an uncertain gain. We have parameter sets of the form

$$m_p = \bar{m}\left(1 + r_m \Delta\right), \quad \bar{m} \triangleq \frac{m_{\min} + m_{\max}}{2}, \quad r_m \triangleq \frac{\left(m_{\max} - m_{\min}\right)/2}{\bar{m}},$$

where r_m is the relative magnitude of the gain uncertainty and $\bar{m}$ is the average gain. Moreover, r_m could also be defined as:

$$r_m = \frac{m_{\max} - m_{\min}}{m_{\min} + m_{\max}}$$

And this equation can also be rewritten as multiplicative uncertainty:

$$G_p(s) = \underbrace{\bar{m}G_0(s)}_{G(s)}\left(1 + r_m\Delta\right), \quad |\Delta| \leq 1$$

where Δ is a real scalar and G(s) is the nominal plant. Therefore, I take the Δ value as 1.

```
% Parametric uncertainty
% Upper and lower bound for mass
m_upperBar = 1.2;
m_lowerBar = 0.8;

% Relative magnitude of the gain uncertainty
rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);

% Any stable transfer function which at each frequency is less
% than or equal to one in magnitude.
delta = 1;

% Plant with multiplicative uncertainty
Gp = G0*(1+rm*delta);
```

Algorithm 3: Parametric Uncertainty for Mass

2.2 Time constant and time delay uncertainty for actuator

$U(t)$ is a force produced by an actuator. I have selected a delay-free nominal actuator model U_0 as given in the book:

$$U_0 = \frac{2.5}{2.5s + 1}$$

If there is an unknown time constant τ and unknown actuator time delay θ as:

$$U(s) = U_0(s)\frac{e^{-\theta s}}{1 + \tau s} \text{ where } \underline{\tau} < \tau < \bar{\tau} \text{ and } \underline{\theta} < \theta < \bar{\theta}$$

I prefer to lump the uncertainty into a multiplicative uncertainty of the form

$$\Pi_I: \quad G_p(s) = G(s)\left(1 + w_I(s)\Delta_I(s)\right); \underbrace{|\Delta_I(j\omega)| \leq 1 \forall\omega}_{\|\Delta_I\|_\infty \leq 1}$$

Here $\Delta_I(s)$ is any stable transfer function which at each frequency is less than or equal to one in magnitude. Some examples of allowable $\Delta_I(s)$'s with $\mathcal{H}_\infty$ norm less than one, $\|\Delta_I\|_\infty \leq 1$, are

$$\frac{s-z}{s+z}, \quad \frac{1}{\tau s + 1}, \quad \frac{1}{(5s+1)^3}, \quad \frac{0.1}{s^2 + 0.1s + 1}$$

Therefore, I have chosen $\Delta_I(s)$ value as:

$$\Delta_I(s) = \frac{1}{(5s+1)^3}$$

To derive $w_I(s)$ we first try a simple first-order weight that matches this limiting behaviour as given in "Skogestad's" book.

$$w_{I1}(s) = \frac{Ts + 0.2}{(T/2.5)s + 1}, \quad T = 4$$

And finally we use:

$$w_I(s) = \omega_{I1}(s)\frac{s^2 + 1.6s + 1}{s^2 + 1.4s + 1}$$

```
% Actuator
U0 = 2.5/(2.5*s+1);
% Simple first-order weight
T = 4;
wI1 = (T*s + 0.2)/((T/2.5)*s+1);
deltaI = 1/(5*s+1)^3;
% Multiply with correction factor to lift the gain
wI = wI1*tf([1 1.6 1],[1 1.4 1]);
Up = U0*(1+wI*deltaI);

% Plant with actuator
G = Up*Gp;
```

Algorithm 4: Parametric Uncertainty for Actuator

2.3 Obtaining generalized plant

The vector output for a servo model which is shown in figure 1, can be defined as:

$$z_1 = W_1(s)T(s)V(s)\omega$$
$$z_2 = W_2(s)S(s)V(s)\omega$$
$$z_3 = W_3(s)U(s)V(s)\omega$$

where $U(s) = K(s)S(s)$.

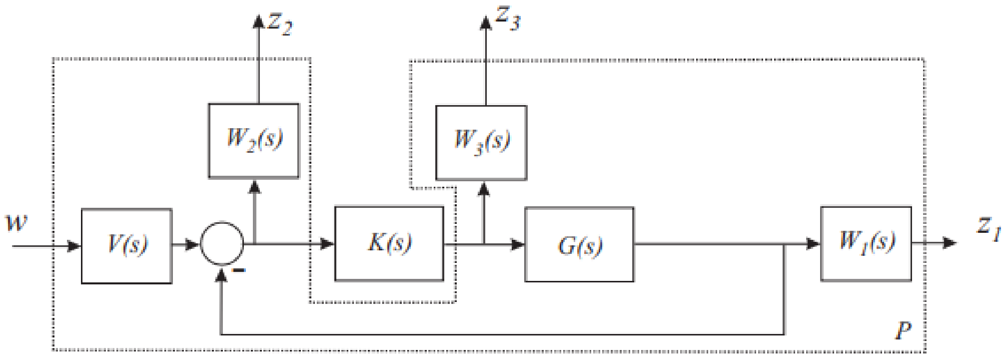

Figure 1: Servo problem

W_1 is large in frequency band where accurate tracking or disturbance rejection is required. W_2 is constant. W_3 is large in frequency band were the model is uncertain.Therefore, for both servo and regulator problems the weighting functions are taken as (Copied from Hinf pdf):

$$W_1 = \frac{4e06}{1e06s^2 + 1400s + 1}$$
$$W_2 = 0.01$$
$$W_3 = \frac{0.05s^2 + 7e - 05s + 5e - 08}{2.5e - 05s^2 + 0.007s + 1}$$

As seen in algorithm 5, the weights obtained is implemented into the "augw" function in Matlab and therefore the generalized plant is built. While implementing the weights into Matlab be careful that W1,W2,W3 are the weights on S, $K * S$, and T. For the servo model the command is used as in 5.

```
% Selecting weighting functions
% W1 is large frequency band where accurate tarcking/disturbance rejection
% is required
wn=1e-3;
num_W1=2000^2;
den_W1=[1/wn^2 2*0.7/wn 1];
W1=tf(num_W1,den_W1);

% W2 is constant
num_W2=1e-3;
den_W2=1;
W2=tf(num_W2,den_W2);

% W3 is large frequency band where model is uncertain
num_W3=5e-8*[1/wn^2 2*0.7/wn 1];
den_W3=[1/200^2 2*0.7/200 1];
W3=tf(num_W3,den_W3);

% Calculation of augmented plant:
P = augw(G,W1,W2,W3);
```

Algorithm 5: Obtaining generalized plan for servot

For a regulator model,shown in figure 2, the vector output can be defined as:

$$z_1 = W_1(s)S(s)V(s)\omega$$
$$z_2 = -W_2(s)U(s)V(s)\omega$$
$$z_3 = -W_3(s)T(s)V(s)\omega$$

where $U(s) = K(s)S(s)$.

Here you have to modify the weights in "augw" command because now the weights W1,-W2,-W3 are the weights on S, $K * S$, and T as seen in **??**. The frequency response for weights can be seen in figure 3.The γ value for H_{inf} controller is 0.9350.

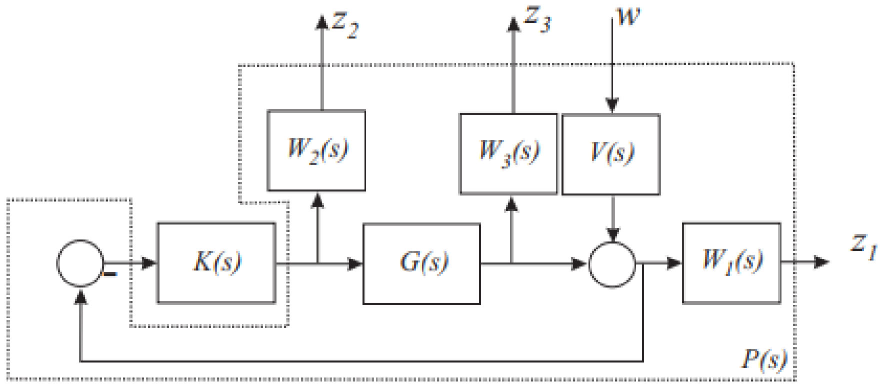

Figure 2: Regulation problem

```
% Selecting weighting functions
% W1 is large frequency band where accurate tarcking/disturbance rejection
% is required
wn = 1e-3;
num_W1 = 2000^2;
den_W1 = [1/wn^2 2*0.7/wn 1];
W1 = tf(num_W1,den_W1);

% W2 is constant
num_W2 = 1e-3;
den_W2 = 1;
W2 = tf(num_W2,den_W2);

% W3 is large frequency band where model is uncertain
num_W3 = 5e-8*[1/wn^2 2*0.7/wn 1];
den_W3 = [1/200^2 2*0.7/200 1];
W3 = tf(num_W3,den_W3);

% Calculation of augmented plant:
P = augw(G,W1,-W2,-W3);
```

Algorithm 6: Frequency response for weights

2.4 Calculation of Optimal Controller

In order to calculate the controller gain we use "hinfsyn" command in Matlab. The inputs to this command is the number of inputs of the controller($nmeas$), number of control inputs of the system P and the generalized plant P.

```
% Calculation of optimal controller
nmeas=1; %  number of inputs of the controller
ncon=1;  %  number of control inputs of the system P
[Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);
```

Algorithm 7: Calculation of Optimal Controller

2.5 Regulation problem 1: Disturbance (sinusoidal) is acting on the plant and Measurement is position

The figure 5 shows the regulation response where the measurement is position for two different sinus values. Also, you can see that the controller can handle the disturbance. And in figure 6 the response is shown using the uncertainty.

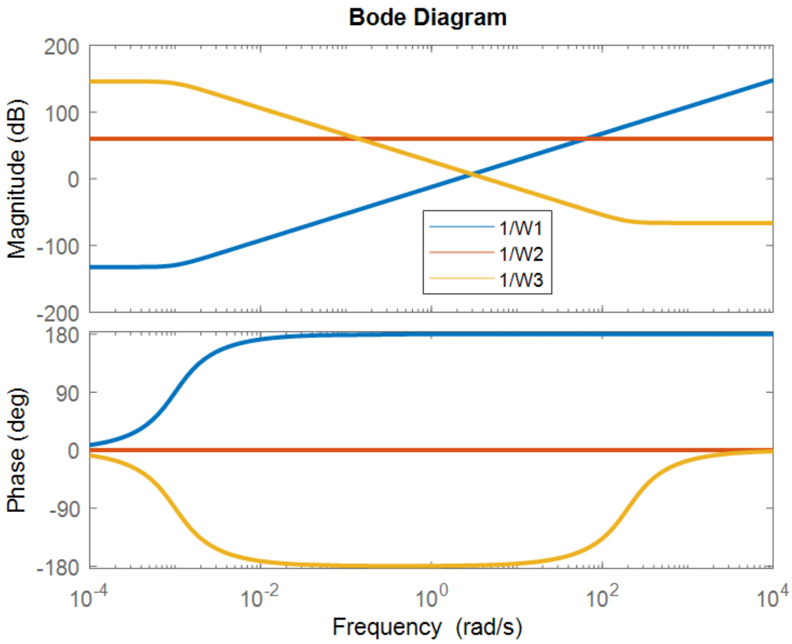

Figure 3: Frequency response for weights

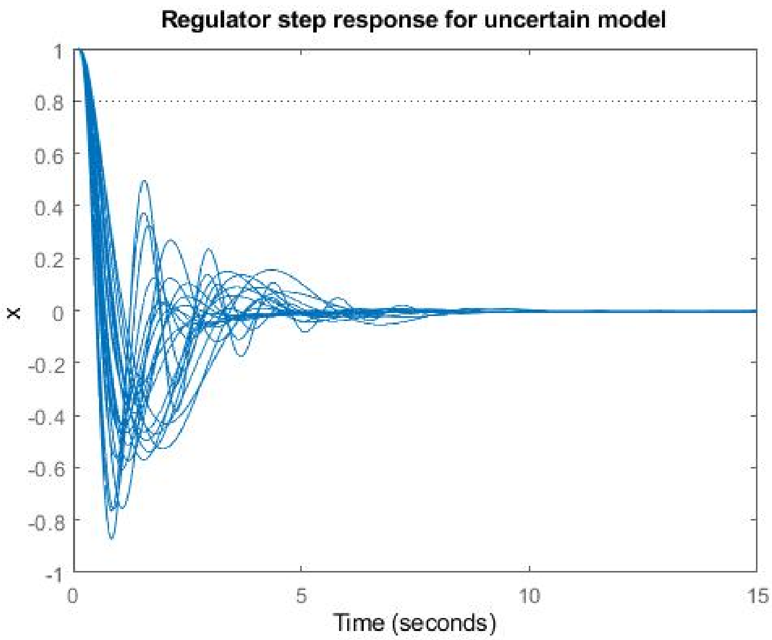

Figure 4: Step response for regulation problem

2.6 Regulation problem 2: Disturbance (sinusoidal) is acting on the plant and Measurement is speed

The figure 8 show the regulation problem where the measurement is speed. And in figure 9 the response is shown using the uncertainty.Also, you can see that the

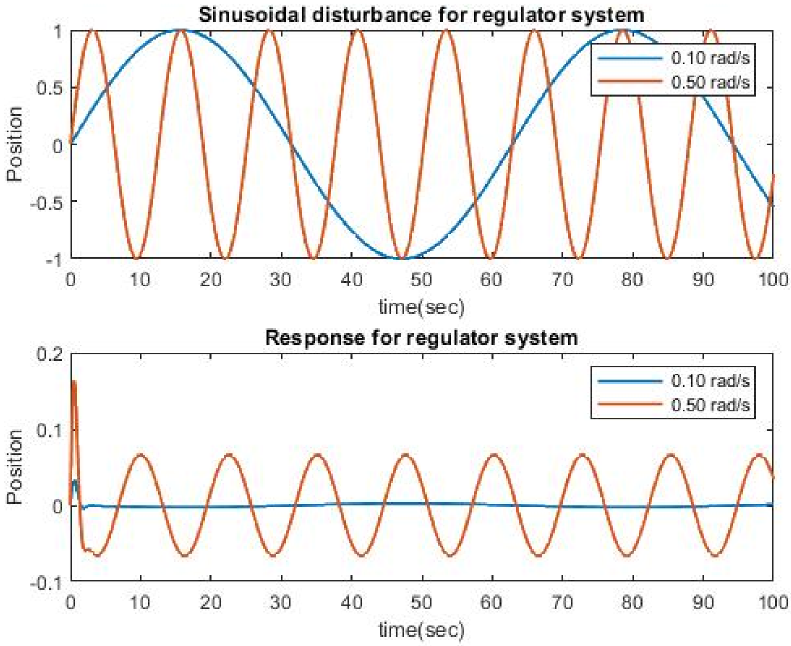

Figure 5: Regulation problem for position measurement

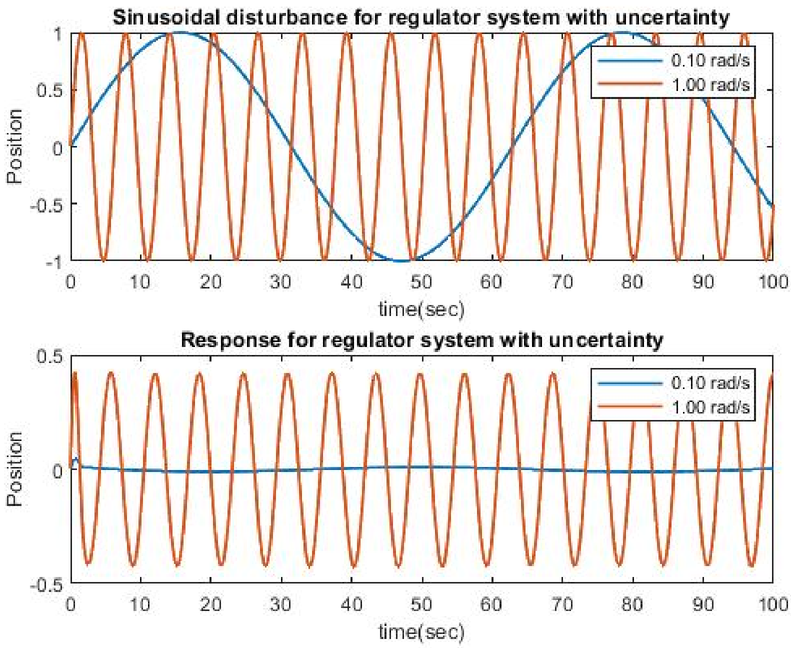

Figure 6: Regulation problem with Uncertainty for position measurement

controller can handle the disturbance.

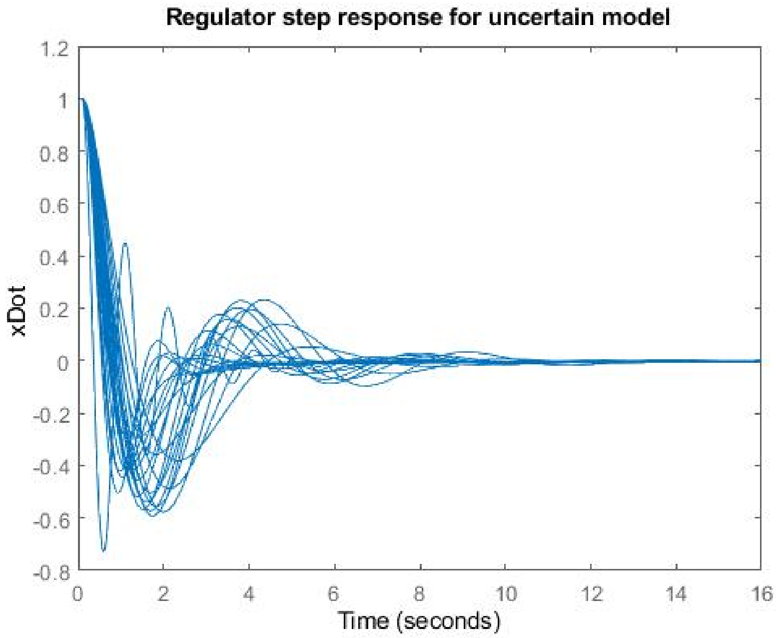

Figure 7: Step response for regulation problem

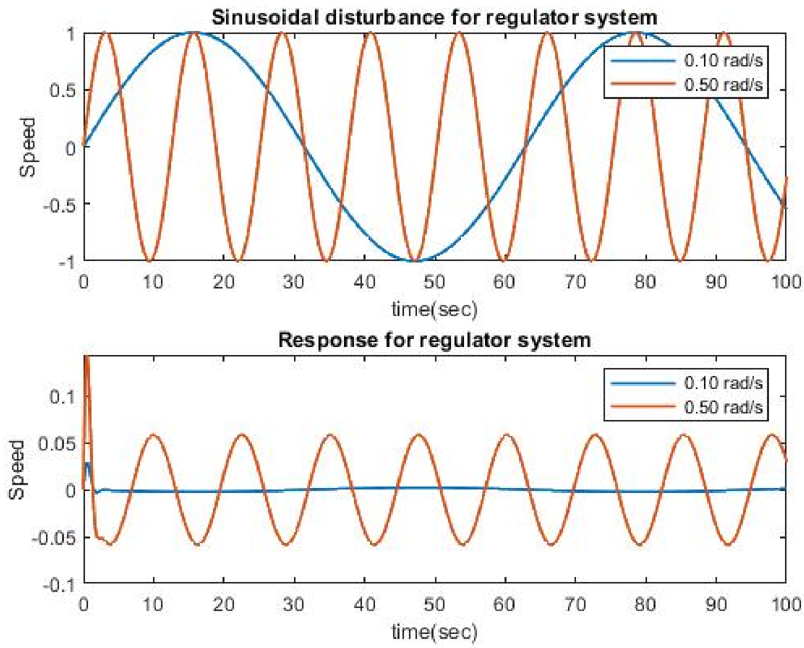

Figure 8: Regulation problem for speed measurement

2.7 Servo problem 1: Reference (sinusoidal) is position and Measurement is position

The figure 11 show the servo problem where the measurement is position. And in figure 12 the response is shown using the uncertainty.Also, you can see that the

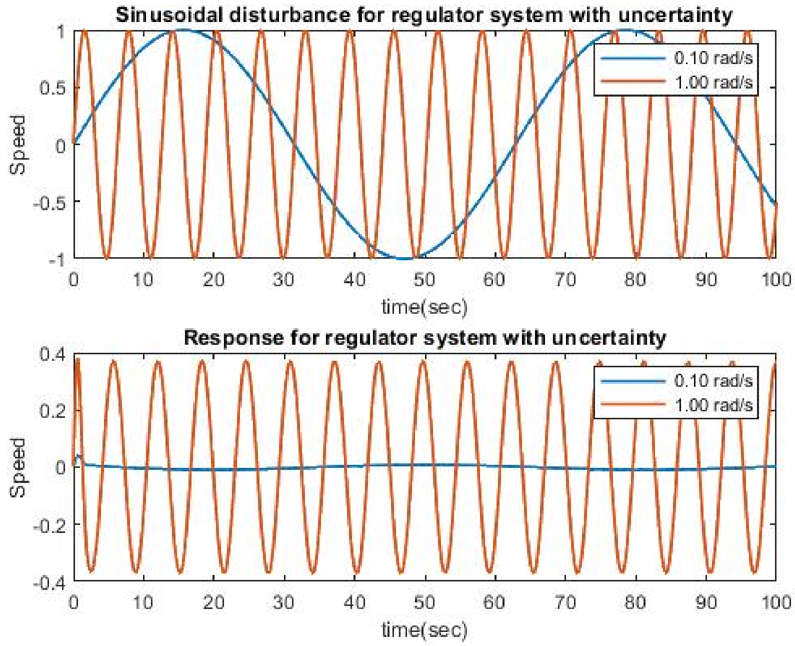

Figure 9: Regulation problem with Uncertainty for speed measurement

controller can track the reference.

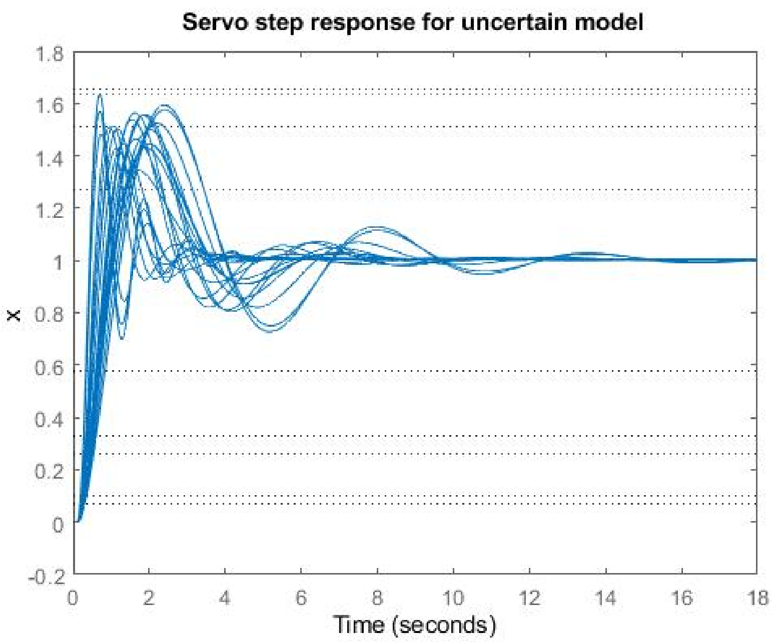

Figure 10: Step response for servo problem

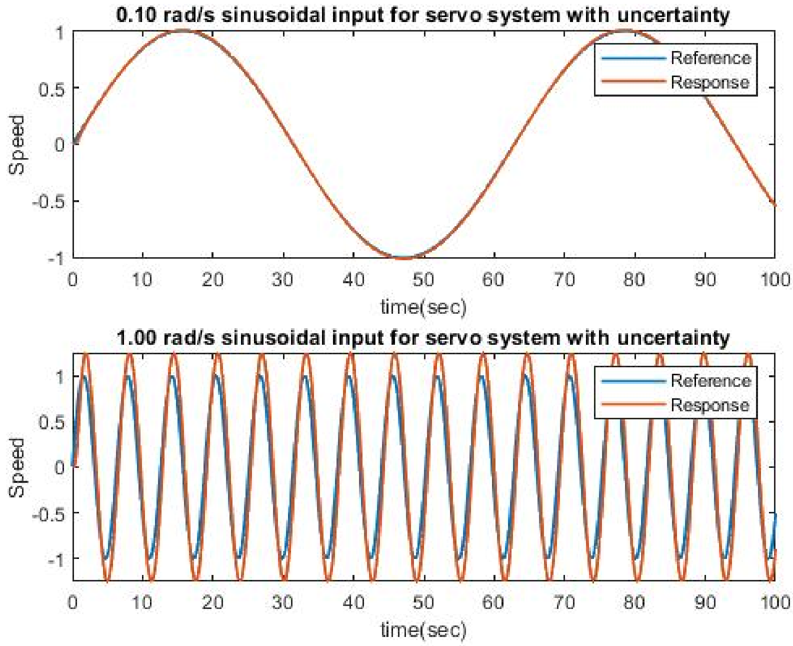

Figure 11: Servo problem for position measurement

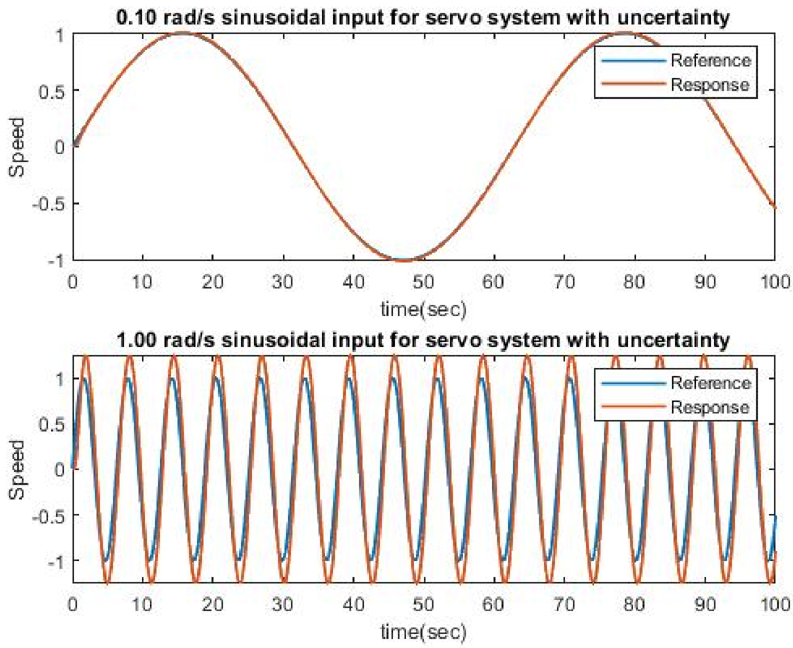

Figure 12: Servo problem with Uncertainty for position measurement

2.8 Servo problem 2: Reference (sinusoidal) is position and Measurement is speed (How?)

Figure 13 shows the response for the servo problem where the measurement is speed. The controller can track the reference.

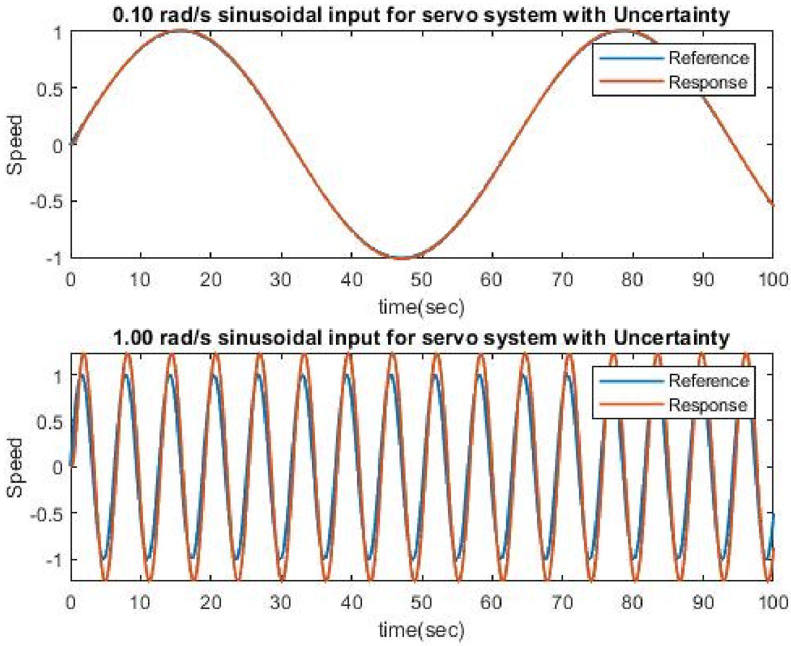

Figure 13: Servo problem with Uncertainty for speed measurement

2.9 Mixed scenarios: both reference and disturbance inputs are applied

Both reference and disturbance are added for plotting figures 14 and 15. When the control input is higher(high frequency), the controller can not control the system.

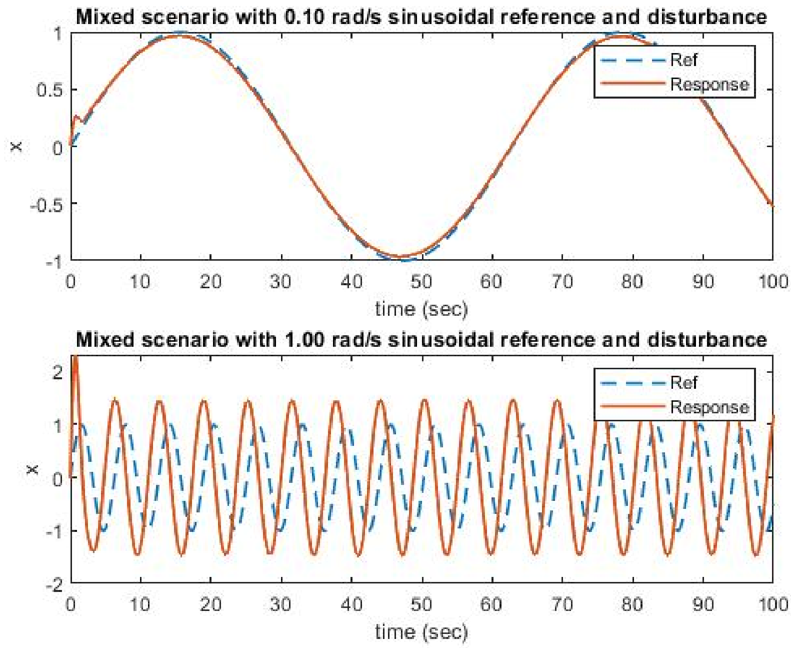

Figure 14: Mixed scenarios for position measurement

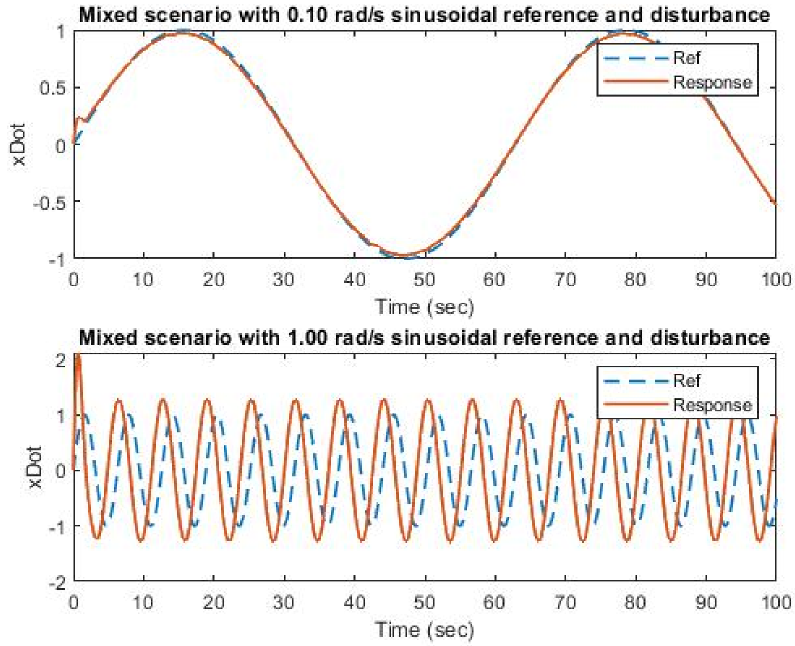

Figure 15: Mixed scenarios for speed measurement

2.10 Discussion for robust stability

Is the system stable for all different plants with different masses/ actuator time constants?
The step responses for regulator are shown in figure 7, 4 and for the servo in figure 10. Therefore, I can say that with uncertain mass and time constant H_{inf} controller can handle problems.

Is the system able to follow sinusoidal references while rejecting sinusoidal disturbances?
Yes, the system is able to follow sinusoidal references while rejecting sinusoidal disturbances as seen from previous figures.

3 Parametric uncertainty Modelling

3.1 Position Measurement

Lets model our system:

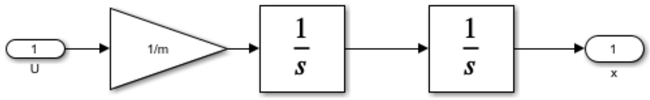

Figure 16: Simple Mass System

And now, we have to model our actuator system. However, the question says that we do not have to include the time delay uncertainty. Therefore, we define our actuator as:

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

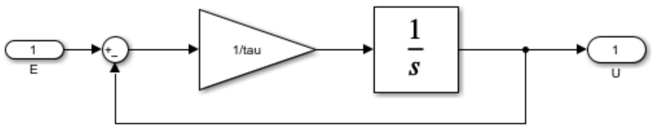

Figure 17: Actuator System

In a realistic system, the two physical parameters m and τ are not known exactly. However, it can be assumed that their values are within certain, known intervals as:

$$m = \bar{m}\left(1 + p_m \delta_m\right), \tau = \bar{\tau}\left(1 + p_\tau \delta_\tau\right)$$

where $\bar{m} = 3, \bar{\tau} = 1$ are the so-called nominal values of m and τ. p_m, p_τ, δ_m and δ_τ represent the possible (relative) perturbations on these two parameters.

We can represent a linear fractional transformation (LFT) for mass and τ repectively as:

$$M_{mi} = \begin{bmatrix} -p_m & \frac{1}{\bar{m}} \\ -p_m & \frac{1}{\bar{m}} \end{bmatrix}$$

$$M_{\tau i} = \begin{bmatrix} -p_\tau & \frac{1}{\bar{\tau}} \\ -p_\tau & \frac{1}{\bar{\tau}} \end{bmatrix}$$

All these LFTs are depicted by block diagrams in 18.

The equations relating all "inputs" to corresponding "outputs" around these perturbed parameters can now be obtained as:

$$\begin{bmatrix} y_m \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} -p_m & \frac{1}{\bar{m}} \\ -p_m & \frac{1}{\bar{m}} \end{bmatrix} \begin{bmatrix} u_m \\ U \end{bmatrix}$$

$$\begin{bmatrix} y_\tau \\ \dot{u} \end{bmatrix} = \begin{bmatrix} -p_\tau & \frac{1}{\bar{\tau}} \\ -p_\tau & \frac{1}{\bar{\tau}} \end{bmatrix} \begin{bmatrix} u_\tau \\ E - U \end{bmatrix}$$

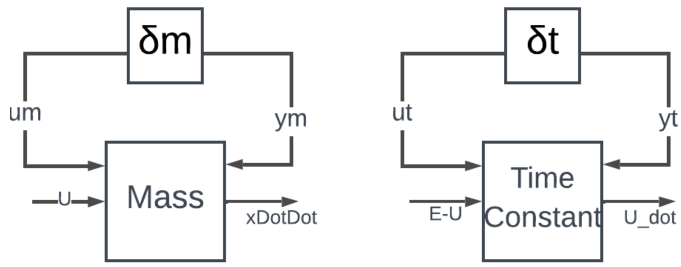

Figure 18: Representation of uncertain parameters as LFTs

$$
\begin{aligned}
u_m &= \delta_m y_m \\
u_\tau &= \delta_\tau y_\tau
\end{aligned}
$$

Now let us set

$$x_1 = x, x_2 = \dot{x} = \dot{x}_1, y = x_1$$

such that

$$\dot{x}_2 = \ddot{x} = \ddot{x}_1$$

As a result, we obtain the following equations

$$
\begin{aligned}
&\dot{x}_1 = x_2 \\
&\dot{x}_2 = -p_m u_m + \frac{1}{\bar{m}}(U) \\
&y_m = -p_m u_m + \frac{1}{\bar{m}}(U) \\
&y_\tau = -p_\tau u_\tau + \frac{1}{\bar{\tau}}(E - U) \\
&\dot{u} = -p_\tau u_\tau + \frac{1}{\bar{\tau}}(E - U) \\
&y = x_1 \\
&u_m = \delta_m y_m \\
&u_\tau = \delta_\tau y_\tau
\end{aligned}
$$

The equations governing the system dynamic behaviour are given by:

$$
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{u} \\ y_m \\ y_\tau \\ y \end{bmatrix}
=
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{m} & -p_m & 0 & 0 \\
0 & 0 & -\frac{1}{\tau} & 0 & -p_\tau & \frac{1}{\tau} \\
0 & 0 & \frac{1}{m} & -p_m & 0 & 0 \\
0 & 0 & -\frac{1}{\tau} & 0 & -p_\tau & \frac{1}{\tau} \\
1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ U \\ u_m \\ u_\tau \\ E \end{bmatrix}
$$

$$
\begin{bmatrix} u_m \\ u_\tau \end{bmatrix}
=
\begin{bmatrix} \delta_m & 0 \\ 0 & \delta_\tau \end{bmatrix}
\begin{bmatrix} y_m \\ y_\tau \end{bmatrix}
$$

The state space representation of G_{mds} is

$$G_{\text{mds}} = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{1}{m} \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix}, B_1 = \begin{bmatrix} 0 & 0 \\ -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix}$$
$$C_1 = \begin{bmatrix} 0 & 0 & \frac{1}{m} \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix}, D_{11} = \begin{bmatrix} -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, D_{12} = \begin{bmatrix} 0 \\ \frac{1}{\tau} \end{bmatrix}$$
$$C_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, D_{21} = \begin{bmatrix} 0 & 0 \end{bmatrix}, D_{22} = 0$$

In order to compute the system matrix G_{mds}, I have used the code given in 9.

```
%% System Model
% Mass
m = 3;
% Time constant
tau = 1;
% Perturbation for mass
pm = 0.4;
% Perturbation for tau
ptau = 0.3;
%% State-space representation for G_mds
A = [ 0 1 0; 0 0 1/m ; 0 0 -1/tau];
B1 = [ 0 0; -pm 0 ;0 -ptau];
B2 = [ 0; 0; 1/tau];
C1 = [0 0 1/m; 0 0 -1\tau];
C2 = [ 1 0 0];
D11 = [-pm 0; 0 -ptau];
D12 = [0; 1\tau];
D21 = [0 0];
D22 = 0;

G = pck(A,[B1,B2],[C1;C2],[D11 D12;D21 D22]);

% This part is implemented to not see a error
% Unpack G and see the eignevalues
[a,b,c,d] = unpck(G);
eig(a)
% We do not want zero valued eigenvalue
% Zero eigenvalue leads to error
% Therefore, we add a very small number to change the zero eigenvalue
% situation
a(2,1) = 0.0001;
eig(a)
G = pck(a,b,c,d);
```

Algorithm 8: Compute system matrix

3.1.1 Frequency Analysis of Uncertain System

The frequency responses of the perturbed open-loop system may be computed by using the command "starp" at a few different values of the perturbation parameters δ_m, δ_τ.In figure 19, two values of each perturbation are chosen, the corresponding open-loop transfer function matrices generated and frequency responses calculated and plotted.

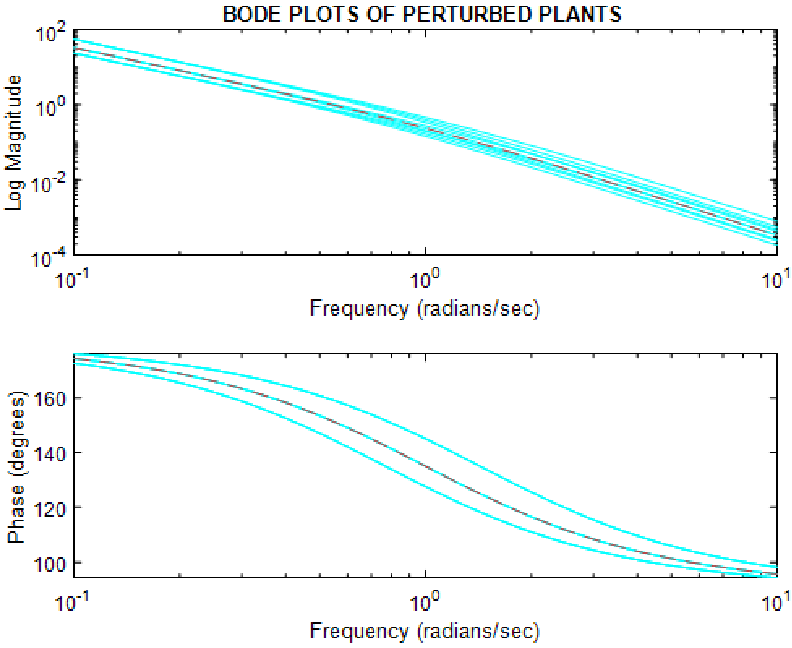

Figure 19: Frequency responses of the perturbed plants

```
%% Frequency responses of the perturbed plants
omega = logspace(-1,1,100);
[delta1,delta2] = ndgrid([-1 0 1],[-1 0 1]);
for j = 1:9
    delta = diag([delta1(j),delta2(j)]);
    olp = starp(delta,G);
    olp_ic = sel(olp,1,1);
    olp_g = frsp(olp_ic,omega);
    figure(1)
    vplot('bode',olp_g,'c-')
    subplot(2,1,1)
    hold on
    subplot(2,1,2)
    hold on
end
subplot(2,1,1)
olp_ic = sel(G,3,3);
olp_g = frsp(olp_ic,omega);
vplot('bode',olp_g,'r--')
subplot(2,1,1)
title('BODE PLOTS OF PERTURBED PLANTS')
hold off
subplot(2,1,2)
hold off
```

Algorithm 9: Frequency responses of the perturbed plants

3.1.2 Design Requirements of Closed-loop System

In the given case, the performance weighting function is a scalar function $W_p(s) = \omega_p(s)$ and chosen as:

$$W_p(s) = 0.95\frac{s^2 + 1.8s + 10}{s^2 + 8 + 0.01}$$

which ensures, apart from good disturbance attenuation, good transient response.The control weighting function W_u is chosen as 50^{-3}. The singular values of $1/\omega_p$ over frequency range $[10^{-4}, 10^4]$. To define the weighting functions and to calculate the inverse weighting function in Matlab, the code 10 is used. It is seen in figure 20 that from the frequency 1 rad/s the disturbance is no longer to be "attenuated".

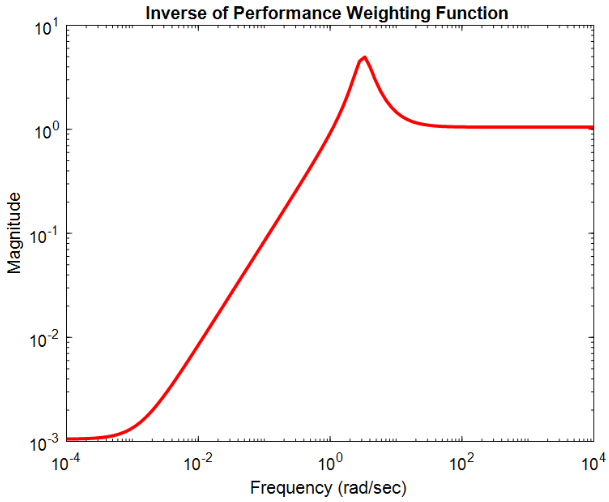

Figure 20: Singular values of $\frac{1}{\omega_p(s)}$

```
%%  Design Requirements of Closed-loop System
% Robust performance:
% Define the chosen weighting functions
nuWp = [1 1.8 10];
dnWp = [1 8 0.01];
gainWp = 0.95;
Wp = nd2sys(nuWp,dnWp,gainWp);
nuWu = 1;
dnWu = 1;
gainWu = 50^(-3);
Wu = nd2sys(nuWu,dnWu,gainWu);
% Calculate the inverse weighting function
omega = logspace(-4,4,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);
figure
vplot('liv,lm',Wpi_g)
title('Inverse of Performance Weighting Function')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')
```

Algorithm 10: Define weighting functions and calculate the inverse weighting function

3.1.3 System Interconnections

The variables pertin and pertout have two elements as seen in 11.The command sysic can be used to create the structure of open-loop systems in Matlab.

```
%% System Interconnections
systemnames = ' G Wp Wu ';
inputvar = '[ pert{2}; dist; control ]';
outputvar = '[ G(1:2); Wp; -Wu; -G(3)-dist ]';
input_to_G = '[ pert; control ]';
input_to_Wp = '[ G(3)+dist ]';
input_to_Wu = '[ control ]';
sysoutname = 'sys_ic';
cleanupsysic = 'no';
% create the structure of open-loop systems
sysic

% To analyse the open-loop system, the following commands can be used.
minfo(sys_ic)
spoles(sys_ic)
spoles(Wp)

% The model of the open-loop system with uncertainties is set
systemnames = ' G ';
inputvar = '[ pert{2}; ref; dist; control ]';
outputvar = '[ G(1:2); G(3)+dist; ref - G(3) - dist ]';
input_to_G = '[ pert; control ]';
sysoutname = 'sim_ic';
cleanupsysic = 'yes';
sysic
```

Algorithm 11: Define system interconnections

3.1.4 Suboptimal H_∞ Controller Design

This controller minimizes the infinite-norm of $F_L(P, K)$ over all stabilising controllers K. We first extract from "sys_ic" the corresponding transfer function matrix using the "sel" command as seen in 12.

```
%% 8.5 Suboptimal Hinf Controller Design
% number of measurements
nmeas = 1;
% number of controls
ncon = 1;
% lower bound of bisection
gmin = 1;
% upper bound of bisection
gmax = 10;
% absolute tolerance for the bisection method
tol = 0.001;
% open-loop interconnection is saved in the variable "hin_ic"
hin_ic = sel(sys_ic,3:5,3:4);
% K_hin: controller (matrix of type SYSTEM)
% clp: closed-loop system (matrix of type SYSTEM)
[K_hin,clp] = hinfsyn(hin_ic,nmeas,ncon,gmin,gmax,tol);
```

Algorithm 12: Compute a suboptimal H_∞ controller

The results can be seen in figure 21.

3.1.5 Analysis of Closed-loop System with K_{hin}

Figure 22 shows the singular values of the closed-loop system clp.

It is seen in figure 23 that in the low-frequency range the sensitivity function lies below $\frac{1}{\omega_p}$.

The frequency responses of the upper and lower bounds of μ are shown in Figure 24. It is clear from the figure that the closed-loop system with K_{hin} achieves robust stability.

The transient responses to the reference input and to the disturbance input are shown in 25 and 26.The transient responses are relatively slow and have slight overshoots.

Test bounds: 1.0000 < gamma <= 10.0000

gamma	hamx_eig	xinf_eig	hamy_eig	yinf_eig	nrho_xy	p/f
10.000	7.2e-01	4.5e-04	1.3e-03	-1.7e-16	0.0001	p
5.500	7.2e-01	4.5e-04	1.3e-03	-6.9e-16	0.0003	p
3.250	7.1e-01	4.5e-04	1.3e-03	-2.8e-15	0.0010	p
2.125	7.1e-01	4.6e-04	1.3e-03	-4.9e-50	0.0025	p
1.562	7.1e-01	4.7e-04	1.3e-03	-1.3e-23	0.0052	p
1.281	7.0e-01	4.8e-04	1.3e-03	-1.7e-52	0.0091	p
1.141	7.0e-01	4.9e-04	1.3e-03	-3.1e-50	0.0135	p
1.070	6.9e-01	5.0e-04	1.3e-03	-1.7e-50	0.0174	p
1.035	6.9e-01	5.1e-04	1.3e-03	-1.2e-51	0.0202	p
1.018	6.9e-01	5.1e-04	1.3e-03	0.0e+00	0.0220	p
1.009	6.9e-01	5.2e-04	1.3e-03	0.0e+00	0.0230	p
1.004	6.9e-01	5.2e-04	1.3e-03	-6.3e-50	0.0236	p
1.002	6.9e-01	5.2e-04	1.3e-03	-2.8e-50	0.0239	p
1.001	6.9e-01	5.2e-04	1.3e-03	-1.7e-49	0.0240	p
1.001	6.9e-01	5.2e-04	1.3e-03	-1.0e-51	0.0241	p

Gamma value achieved: 1.0005

Figure 21: Display of results for suboptimal H_∞ controller

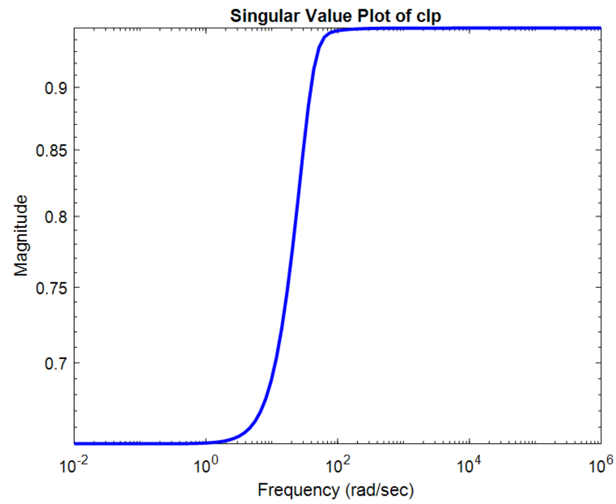

Figure 22: Singular values of the closed-loop system with K_{hin}

In order to plot the previous figures in this section use algorithm 13.

3.1.6 Regulation Problem

From figure 27, it can be seen that the controller can handle the disturbance. However, the unstructured uncertainty approach can handle the disturbance in a more better way.

3.1.7 Servo Problem

For the servo problem, this method allows the tracking to be more accurate. Therefore, the tracking is better in parametric uncertainty approach as seen in figure 28.

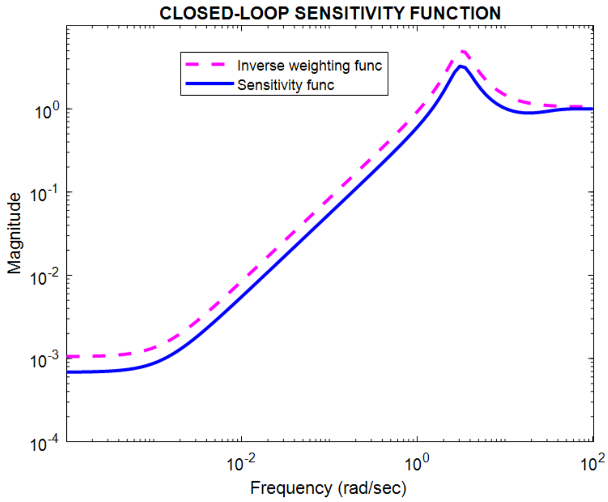

Figure 23: Sensitivity function with K_{hin}

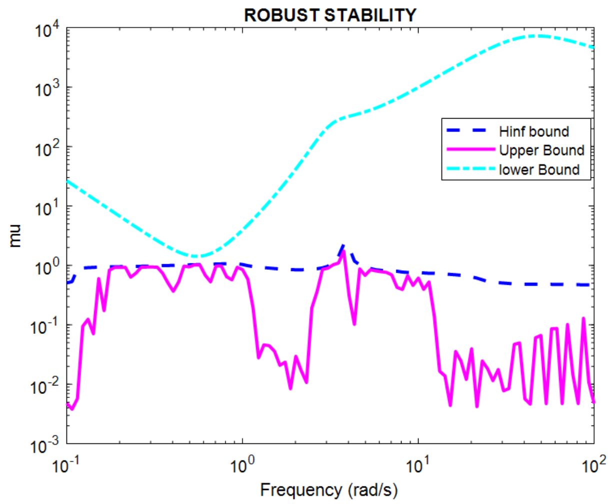

Figure 24: Robust stability analysis of K_{hin}

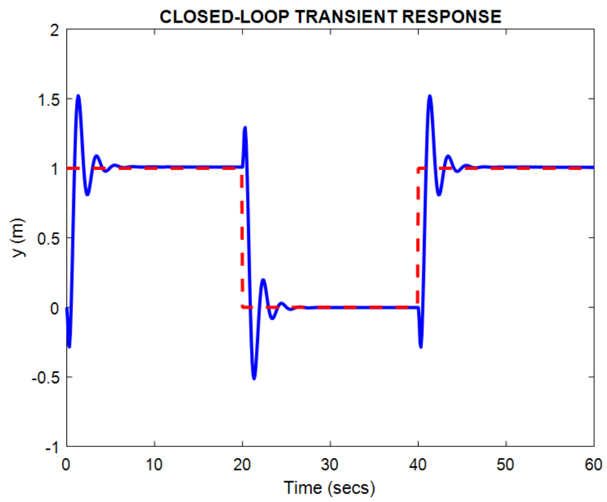

Figure 25: Transient response to reference input (K_{hin})

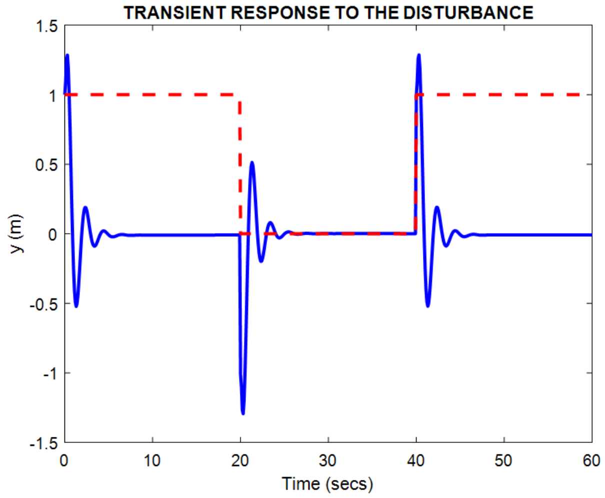

Figure 26: Transient response to disturbance input (K_{hin})

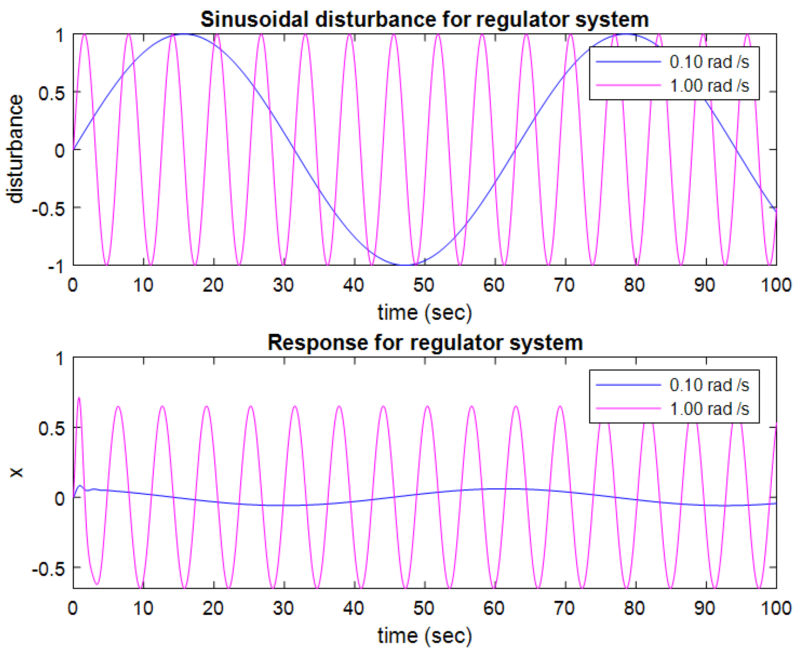

Figure 27: Sinusoidal disturbance for Regulator system for position measurement

```
% Analysis of Closed-loop System with Khin
% Singular values of the closed-loop system with Khin
minfo(K_hin)
spoles(K_hin)
omega = logspace(-2,6,100);
clp_g = frsp(clp,omega);
vplot('liv,lm',vsvd(clp_g))
title('Singular Value Plot of clp')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Sensitivity function with Khin plot
K = K_hin;
clp = starp(sim_ic,K);

% inverse performance weighting function
omega = logspace(-4,2,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);

% sensitivity function
sen_loop = sel(clp,3,4);
sen_g = frsp(sen_loop,omega);
vplot('liv,lm',Wpi_g,'m--',vnorm(sen_g),'y-')
title('CLOSED-LOOP SENSITIVITY FUNCTION')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Robust stability analysis of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1;-1 1];
rob_stab = sel(clp_g,[1:3],[1:3]);
pdim = ynum(rob_stab);
fixl = [eye(pdim); 0.1*eye(pdim)]; % 1% Complex
fixr = fixl';
blkrs = [blkrsR; abs(blkrsR)];
clp_mix = mmult(fixl,rob_stab,fixr);
[rbnds,rowd,sens,rowp,rowg] = mu(clp_mix,blkrs);
disp(' ')
disp(['mu-robust stability: ' ...
num2str(pkvnorm(sel(rbnds,1,1)))])
disp(' ')
vplot('liv,lm',sel(rbnds,1,1),'y--',sel(rbnds,1,2),'m-', ...
vnorm(rob_stab),'c-.')
title('ROBUST STABILITY')
xlabel('Frequency (rad/s)')
ylabel('mu')

% Nominal and robust performance of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1];

% Nominal performance
nom_perf = sel(clp_g,3,3);

% Robust performance
rob_perf = clp_g;
blkrp = [blkrsR;[1 2]];
bndsrp = mu(rob_perf,blkrp);
vplot('liv,lm',vnorm(nom_perf),'y-',sel(bndsrp,1,1),'m--',...
sel(bndsrp,1,2),'c--')
tmp1 = 'NOMINAL PERFORMANCE (solid) and';
tmp2 = ' ROBUST PERFORMANCE (dashed)';
title([tmp1 tmp2])
xlabel('Frequency (rad/s)')
disp(' ')
disp(['mu-robust performance: ' ...
num2str(pkvnorm(sel(bndsrp,1,1)))])
disp(' ')

% Transient response to reference/disturbance input
clp = starp(sim_ic,K_hin);
timedata = [0 20 40];
stepdata = [1 0 1];
dist = 0;
ref = step_tr(timedata,stepdata,0.1,60);
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',ref,'r--')
title('CLOSED-LOOP TRANSIENT RESPONSE')
xlabel('Time (secs)')
ylabel('y (m)')

% Response to the disturbance
timedata = [0 20 40];
stepdata = [1 0 1];
dist = step_tr(timedata,stepdata,0.1,60);
ref = 0;
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',dist,'r--')
title('TRANSIENT RESPONSE TO THE DISTURBANCE')
xlabel('Time (secs)')
ylabel('y (m)')
```


Algorithm 13: Plot figures for Analysis

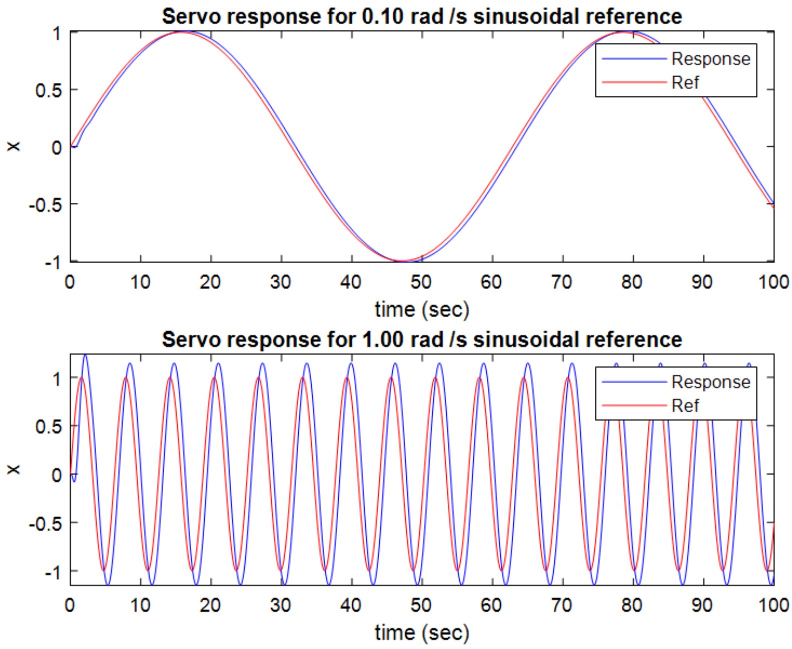

Figure 28: Servo response for different disturbances for position measurement

3.2 Speed Measurement

In order to use speed as measurement, we need to design the system as $1/s$.In this part, we only change the system matrix and we do the same calculations.

As a result, we obtain the following equations

$$
\begin{aligned}
\dot{x} &= -p_m u_m + \frac{1}{\bar{m}}(U) \\
y_m &= -p_m u_m + \frac{1}{\bar{m}}(U) \\
y_\tau &= -p_\tau u_\tau + \frac{1}{\bar{\tau}}(E-U) \\
\dot{u} &= -p_\tau u_\tau + \frac{1}{\bar{\tau}}(E-U) \\
y &= x \\
u_m &= \delta_m y_m \\
u_\tau &= \delta_\tau y_\tau
\end{aligned}
$$

The equations governing the system dynamic behaviour are given by:

$$
\begin{bmatrix} \dot{x} \\ \dot{u} \\ y_m \\ y_\tau \\ y \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{m} & -p_m & 0 & 0 \\ 0 & -\frac{1}{\tau} & 0 & -p_\tau & \frac{1}{\tau} \\ 0 & \frac{1}{m} & -p_m & 0 & 0 \\ 0 & -\frac{1}{\tau} & 0 & -p_\tau & \frac{1}{\tau} \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ U \\ u_m \\ u_\tau \\ E \end{bmatrix}
$$

$$
\begin{bmatrix} u_m \\ u_\tau \end{bmatrix} = \begin{bmatrix} \delta_m & 0 \\ 0 & \delta_\tau \end{bmatrix} \begin{bmatrix} y_m \\ y_\tau \end{bmatrix}
$$

The state space representation of G_{mds} is

$$
G_{\mathrm{mds}} = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array} \right]
$$

where

$$
\begin{aligned}
A &= \begin{bmatrix} 0 & \frac{1}{m} \\ 0 & -\frac{1}{\tau} \end{bmatrix}, B_1 = \begin{bmatrix} -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ \frac{1}{\tau} \end{bmatrix} \\
C_1 &= \begin{bmatrix} 0 & \frac{1}{m} \\ 0 & -\frac{1}{\tau} \end{bmatrix}, D_{11} = \begin{bmatrix} -p_m & 0 \\ 0 & -p_\tau \end{bmatrix}, D_{12} = \begin{bmatrix} 0 \\ \frac{1}{\tau} \end{bmatrix} \\
C_2 &= \begin{bmatrix} 1 & 0 \end{bmatrix}, D_{21} = \begin{bmatrix} 0 & 0 \end{bmatrix}, D_{22} = 0
\end{aligned}
$$

3.2.1 Frequency Analysis of Uncertain System

The frequency responses of the perturbed open-loop system may be computed by using the command "starp" at a few different values of the perturbation parameters δ_m, δ_τ.In figure 29, two values of each perturbation are chosen, the corresponding open-loop transfer function matrices generated and frequency responses calculated and plotted.

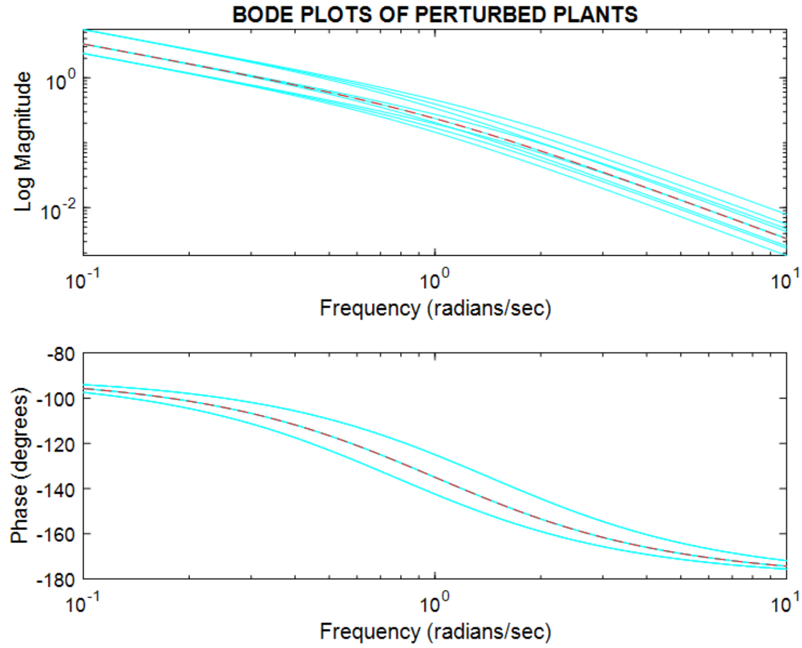

Figure 29: Frequency responses of the perturbed plants

3.2.2 Design Requirements of Closed-loop System

In the given case, the performance weighting function is a scalar function $W_p(s) = \omega_p(s)$ and chosen as:

$$W_p(s) = 0.95\frac{s^2 + 1.8s + 10}{s^2 + 8 + 0.01}$$

which ensures, apart from good disturbance attenuation, good transient response.The control weighting function W_u is chosen as 50^{-3}. The singular values of $1/\omega_p$ over frequency range $[10^{-4}, 10^4]$. It is seen in figure 30 that from the frequency 1 rad/s the disturbance is no longer to be "attenuated".

3.2.3 System Interconnections

The command sysic can be used to create the structure of open-loop systems in Matlab.

3.2.4 Suboptimal H_∞ Controller Design

This controller minimizes the infinite-norm of $F_L(P, K)$ over all stabilising controllers K. We first extract from "sys_ic" the corresponding transfer function matrix using the "sel" command.

3.2.5 Analysis of Closed-loop System with K_{hin}

Figure 31 shows the singular values of the closed-loop system clp.

It is seen in figure 32 that in the low-frequency range the sensitivity function lies below $\frac{1}{\omega_p}$.

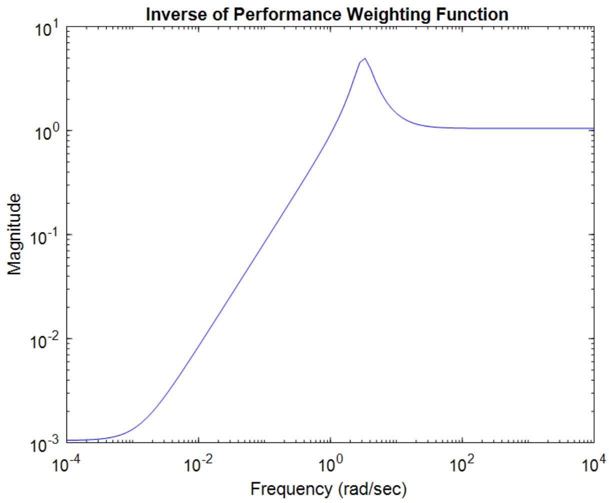

Figure 30: Singular values of $\frac{1}{\omega_p(s)}$

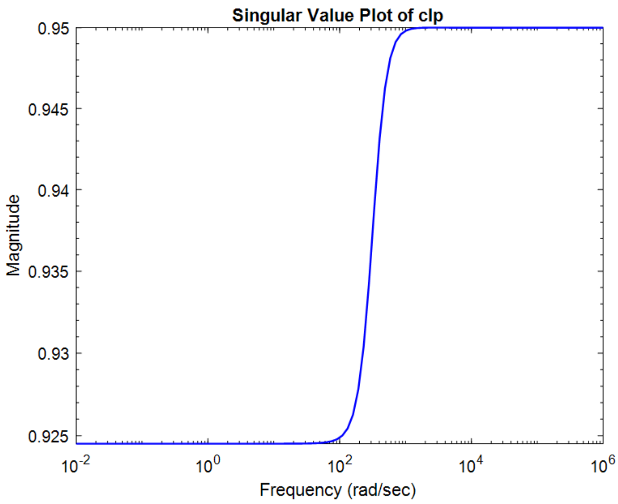

Figure 31: Singular values of the closed-loop system with K_{hin}

The frequency responses of the upper and lower bounds of μ are shown in Figure 34. It is clear from the figure that the closed-loop system with K_{hin} achieves robust stability.

The transient responses to the reference input and to the disturbance input are shown in 35 and 36.The transient responses are relatively slow and have slight overshoots.

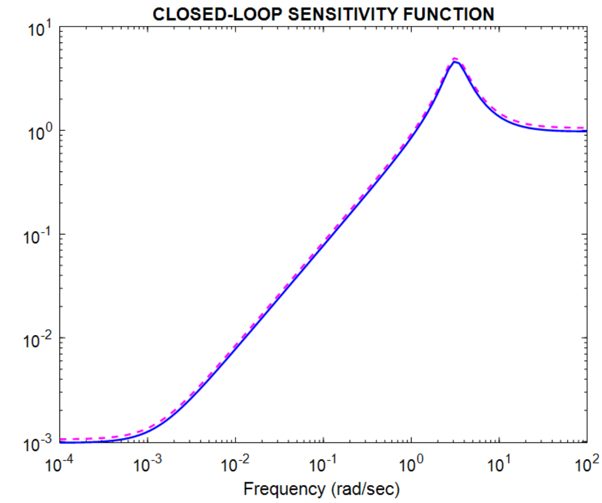

Figure 32: Sensitivity function with K_{hin}

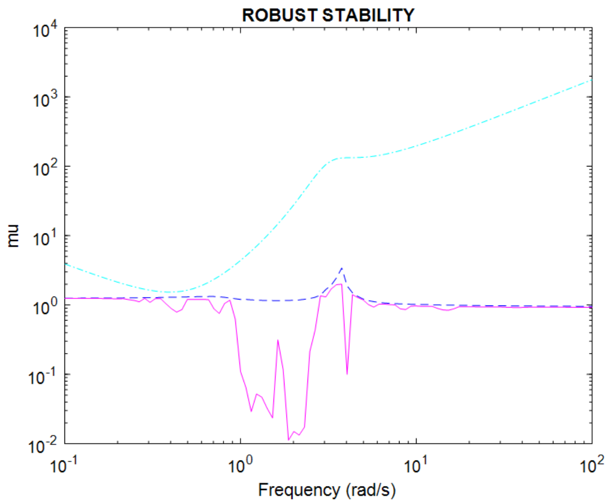

Figure 33: Robust stability analysis of K_{hin}

3.2.6 Regulation Problem

From figure 37, it can be seen that the controller can handle the disturbance. However, the unstructured uncertainty approach can handle the disturbance in a more better way.

3.2.7 Servo Problem

For the servo problem, this method allows the tracking to be more accurate. Therefore, the tracking is better in parametric uncertainty approach as seen in figure 38.

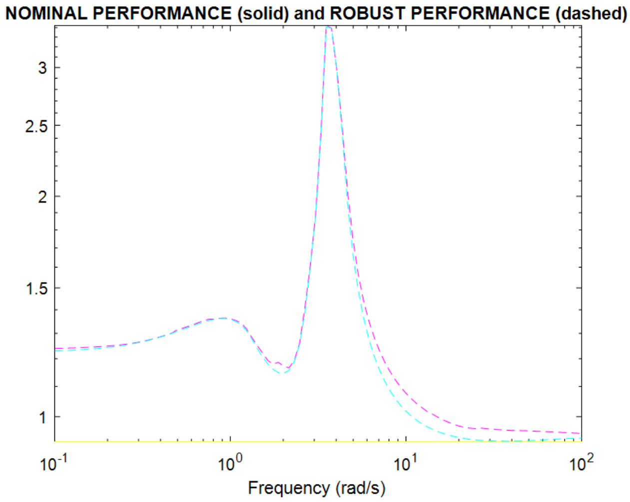

Figure 34: Nominal and robust performance

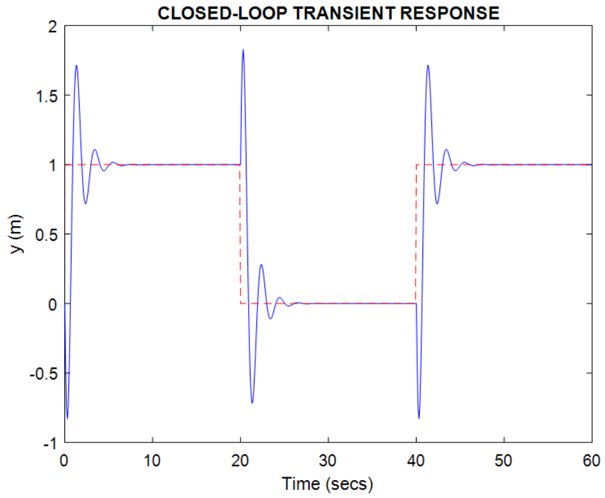

Figure 35: Transient response to reference input (K_{hin})

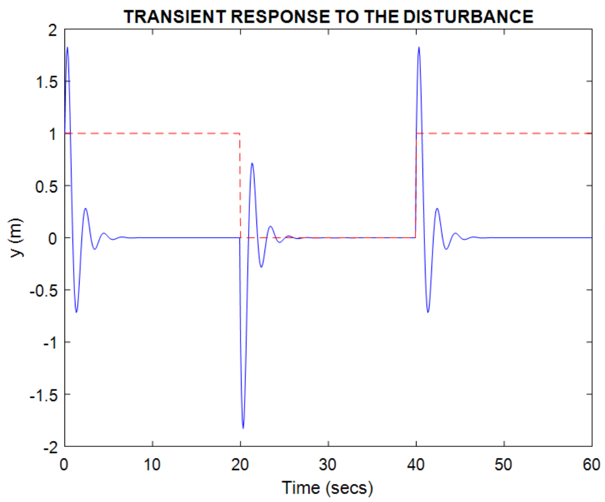

Figure 36: Transient response to disturbance input (K_{hin})

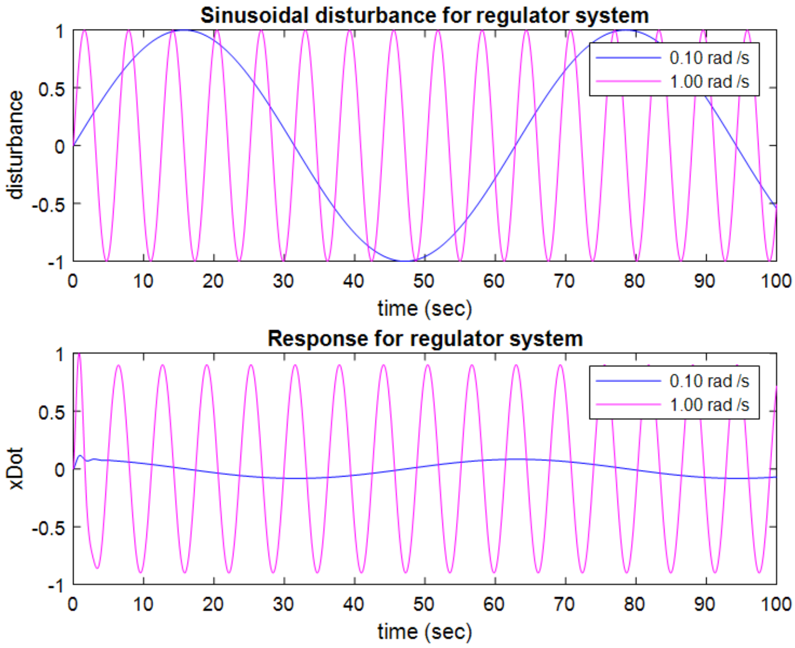

Figure 37: Sinusoidal disturbance for Regulator system for speed measurement

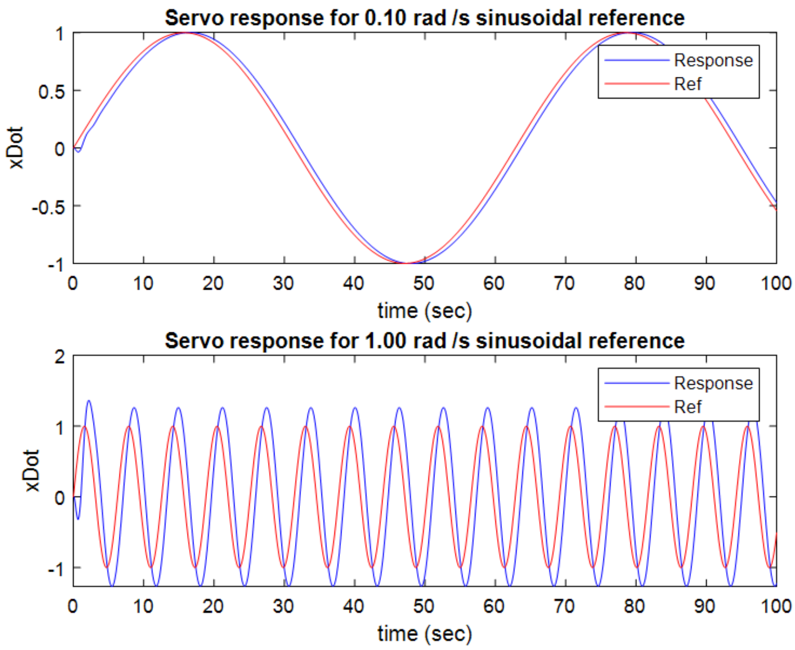

Figure 38: Servo response for different disturbances for speed measurement

Appendix

Part-A Regulator System

```
clc;clear all;close all;
%% Hinf controller design with unstructured uncertainty approach
% Part A) Regulation Problem with uncertainty -- Measurement is position
% Part B) Regulation Problem with uncertainty -- Measurement is speed
%% Part A) Regulation Problem with uncertainty -- Measurement is position
% Define Plant
num=1;
den=[1 1e-3 1e-6];
% Nominal plant (Transfer function with no uncertainty)
G0=tf(num,den);

s = tf('s');

% Parametric uncertainty
% Upper and lower bound for mass
m_upperBar = 1.2;
m_lowerBar = 0.8;

% Relative magnitude of the gain uncertainty
rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);

% Any stable transfer function which at each frequency is less
% than or equal to one in magnitude.
delta = 1;

% Plant with multiplicative uncertainty
Gp = G0*(1+rm*delta);

% Add Actuator
% Delay-free nominal mode
U0 = 2.5/(2.5*s+1);
T = 4;
% Simple first-order weight
wI1 = (T*s + 0.2)/((T/2.5)*s+1);
deltaI = 1/(5*s+1)^3;
% deltaI = 0.1 / (s^2 + 0.1*s + 1);


% Multiply with correction factor to lift the gain
wI = wI1*tf([1 1.6 1],[1 1.4 1]);
Up = U0*(1+wI*deltaI);

% Plant with actuator
G = Up*Gp;

% Selecting weighting functions
% W1 is large frequency band where accurate tarcking/disturbance rejection
```

```
% is required
wn=1e-3;
num_W1=2000^2;
den_W1=[1/wn^2 2*0.7/wn 1];
W1=tf(num_W1,den_W1);

% W2 is constant
num_W2=1e-3;
den_W2=1;
W2=tf(num_W2,den_W2);

% W3 is large frequency band where model is uncertain
num_W3=5e-8*[1/wn^2 2*0.7/wn 1];
den_W3=[1/200^2 2*0.7/200 1];
W3=tf(num_W3,den_W3);

% Calculation of augmented plant:
P = augw(G,W1,-W2,-W3);

% Calculation of optimal controller
nmeas=1; %  number of inputs of the controller
ncon=1;  %  number of control inputs of the system P
[Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);

L = G * Kinf;
% Sensitivity function
S = 1/(1 + L);
% Comp. Sensitivity function
T = L/(1 + L);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
% Bode Plot
figure
bode(1/W1)
hold on;
bode(1/W2)
hold on;
bode(1/W3)
hold on;

%%%%%%%%%%%%%%%
Gd = 1;
%
d = 1;
r = 0;

y = T*r + S*Gd*d;

% time variables
t = 0:0.1:100;
w = [0.1 0.5];

figure;
subplot(2,1,1)
% Plot Disturbance
for i = 1:length(w)
```

```
    u_in = sin(w(i).*t);
    [y_out,t_out,x] = lsim(y, u_in, t);
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Position');
    title('Sinusoidal disturbance for regulator system');
end
legend(legendStr);

hold on;
subplot(2,1,2)
% Plot Responses
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out,t_out,x] = lsim(y, u_in, t);
    plot(t_out,y_out,'LineWidth',1.25);
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Position');
    hold on;
    title('Response for regulator system');
end
legend(legendStr);

% Uncertainty
% Ureal for mass
m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
% Ureal for actuator gain
U0_ureal = ureal('uo_ureal',1,'Range',[0.5 2]);
% Ureal for time constant
tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2]);
% Time delay value
theta_ureal = 0.1;

% Plant
G_ureal = 1/(m_ureal*s^2);
% Actuator
U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);

L_ureal = (U_ureal*G_ureal) * Kinf;
% Sensitivity function
S_ureal = 1 / (1 + L_ureal);
% Comp. Sensitivity function
T_ureal = L_ureal / (1 + L_ureal);

% plot the uncertain reponse of the controller
figure;
y_ureal = T_ureal * r + S_ureal * d;
step(y_ureal)
xlabel('Time');
ylabel('x');
title('Regulator step response for uncertain model');

% time variables
```

```
t = 0:0.1:100;
w = [0.1 1];

figure;
subplot(2,1,1)
% Plot Disturbance
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Position');
    title('Sinusoidal disturbance for regulator system with
   uncertainty');
end
legend(legendStr);

hold on;
subplot(2,1,2)
% Plot Responses
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
    plot(t_out,y_out_ureal,'LineWidth',1.25);
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Position');
    hold on;
    title('Response for regulator system with uncertainty for
   position measurement');
end
legend(legendStr);

%% Plot mixed scenarios for position
r = 1;
d = 5;
y = T_ureal * r + S_ureal * d;
figure;
for i = 1:length(w)
    uin = sin(w(i).*t);
    subplot(2,1,i)
    plot(t,uin,'--','LineWidth',1.25);
    hold on;
    [y_Out,t_Out,x] = lsim(y, uin, t);
    plot(t_Out,y_Out,'LineWidth',1.25);
    legend('Ref','Response')
    xlabel('time (sec)');
    ylabel('x');
    title(sprintf('Mixed scenario with %.2f rad/s sinusoidal
   reference and disturbance',w(i)));
end

%% Part B) Regulation Problem with uncertainty -- Measurement is
   speed
```

```
% Define Plant
num = 1;
den = [1 1e-3];
% Nominal plant (Transfer function with no uncertainty)
G0 = tf(num,den);

s = tf('s');

% Parametric uncertainty
% Upper and lower bound for mass
m_upperBar = 1.2;
m_lowerBar = 0.8;

% Relative magnitude of the gain uncertainty
rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);

% Any stable transfer function which at each frequency is less
% than or equal to one in magnitude.
delta = 1;

% Plant with multiplicative uncertainty
Gp = G0*(1+rm*delta);

% Add Actuator
% Delay-free nominal mode
U0 = 2.5/(2.5*s+1);
T = 4;
% Simple first-order weight
wI1 = (T*s + 0.2)/((T/2.5)*s+1);
deltaI = 1/(5*s+1)^3;
% deltaI = 0.1 / (s^2 + 0.1*s + 1);

% Multiply with correction factor to lift the gain
wI = wI1*tf([1 1.6 1],[1 1.4 1]);
Up = U0*(1+wI*deltaI);

% Plant with actuator
G = Up*Gp;

% Selecting weighting functions
% W1 is large frequency band where accurate tarcking/disturbance
    rejection
% is required
wn = 1e-3;
num_W1 = 2000^2;
den_W1 = [1/wn^2 2*0.7/wn 1];
W1 = tf(num_W1,den_W1);

% W2 is constant
num_W2 = 1e-3;
den_W2 = 1;
W2 = tf(num_W2,den_W2);

% W3 is large frequency band where model is uncertain
num_W3 = 5e-8*[1/wn^2 2*0.7/wn 1];
den_W3 = [1/200^2 2*0.7/200 1];
```

```
W3 = tf(num_W3,den_W3);

% Calculation of augmented plant:
P = augw(G,W1,-W2,-W3);

% Calculation of optimal controller
nmeas = 1; %  number of inputs of the controller
ncon = 1;  %  number of control inputs of the system P
[Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);

L = G * Kinf;
% Sensitivity function
S = 1/(1 + L);
% Comp. Sensitivity function
T = L/(1 + L);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
% % Bode Plot
% figure
% bode(1/W1)
% hold on;
% bode(1/W2)
% hold on;
% bode(1/W3)
% hold on;

%%%%%%%%%%%%%%
Gd = 1;
d = 1;
r = 0;

y = T*r + S*Gd*d;

% time variables
t = 0:0.1:100;
w = [0.1 0.5];

figure;
subplot(2,1,1)
% Plot Disturbance
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out,t_out,x] = lsim(y, u_in, t);
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Speed');
    title('Sinusoidal disturbance for regulator system');
end
legend(legendStr);

hold on;
subplot(2,1,2)
% Plot Responses
for i = 1:length(w)
```

```
    u_in = sin(w(i).*t);
    [y_out,t_out,x] = lsim(y, u_in, t);
    plot(t_out,y_out,'LineWidth',1.25);
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Speed');
    hold on;
    title('Response for regulator system');
end
legend(legendStr);

% Uncertainty
% Ureal for mass
m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
% Ureal for actuator gain
U0_ureal = ureal('uo_ureal',1,'Range',[0.5 1.5]);
% Ureal for time constant
tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2.5]);
% Time delay value
theta_ureal = 0.1;

% Plant
G_ureal = 1/(m_ureal*s);
% Actuator
U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);

L_ureal = (U_ureal*G_ureal) * Kinf;
% Sensitivity function
S_ureal = 1 / (1 + L_ureal);
% Comp. Sensitivity function
T_ureal = L_ureal / (1 + L_ureal);

% plot the uncertain reponse of the controller
figure;
y_ureal = T_ureal * r + S_ureal * d;
step(y_ureal)
xlabel('Time');
ylabel('xDot');
title('Regulator step response for uncertain model');

% time variables
t = 0:0.1:100;
w = [0.1 1];

figure;
subplot(2,1,1)
% Plot Disturbance
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Speed');
```

```
    title('Sinusoidal disturbance for regulator system with
    uncertainty');
end
legend(legendStr);

hold on;
subplot(2,1,2)
% Plot Responses
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
    plot(t_out,y_out_ureal,'LineWidth',1.25);
    legendStr(i) = sprintf(" %.2f rad/s", w(i));
    xlabel('time(sec)');
    ylabel('Speed');
    hold on;
    title('Response for regulator system with uncertainty');
end
legend(legendStr);


%% Mixed scenarios
r = 1;
d = 5;
y = T_ureal * r + S_ureal * d;
figure;
for i = 1:length(w)
    uin = sin(w(i).*t);
    subplot(2,1,i)
    plot(t,uin,'--','LineWidth',1.25);
    hold on;
    [y_Out,t_Out,x] = lsim(y, uin, t);
    plot(t_Out,y_Out,'LineWidth',1.25);
    legend('Ref','Response')
    xlabel('Time (sec)');
    ylabel('xDot');
    title(sprintf('Mixed scenario with %.2f rad/s sinusoidal
    reference and disturbance',w(i)));
end
```

Part-A Servo System

```
clc;clear all;close all;
%% Hinf controller design with unstructured uncertainty approach
% Part A) Servo Problem with uncertainty -- Measurement is position
% Part B) Servo Problem with uncertainty -- Measurement is speed
%% Part A) Servo Problem with uncertainty -- Measurement is
    position
% Define Plant
num=1;
den=[1 1e-3 1e-6];
G0=tf(num,den);

s = tf('s');

% Parametric uncertainty
```

```
% Upper and lower bound for mass
m_upperBar = 1.2;
m_lowerBar = 0.8;

% Relative magnitude of the gain uncertainty
rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);

% Any stable transfer function which at each frequency is less
% than or equal to one in magnitude.
delta = 1;
Gp = G0*(1+rm*delta);

% Actuator
U0 = 2.5/(2.5*s+1);
% Simple first-order weight
T = 4;
wI1 = (T*s + 0.2)/((T/2.5)*s+1);
deltaI = 1/(5*s+1)^3;
% deltaI = 0.1 / (s^2 + 0.1*s + 1);

% Multiply with correction factor to lift the gain
wI = wI1*tf([1 1.6 1],[1 1.4 1]);
Up = U0*(1+wI*deltaI);

% Plant with actuator
G = Up*Gp;

% Selecting weighting functions
% W1 is large frequency band where accurate tarcking/disturbance rejection
% is required
wn = 1e-3;
num_W1 = 2000^2;
den_W1 = [1/wn^2 2*0.7/wn 1];
W1 = tf(num_W1,den_W1);

% W2 is constant
num_W2 = 1e-3;
den_W2 = 1;
W2 = tf(num_W2,den_W2);

% W3 is large frequency band where model is uncertain
num_W3 = 5e-8*[1/wn^2 2*0.7/wn 1];
den_W3 = [1/200^2 2*0.7/200 1];
W3 = tf(num_W3,den_W3);

% Calculation of augmented plant:
P = augw(G,W1,W2,W3);

% Calculation of optimal controller
nmeas = 1;
ncon = 1;
[Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);

L = G*Kinf;
% Sensitivity function
```

```
S = 1/(1 + L);
% Comp. Sensitivity function
T = L/(1 + L);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
% Bode Plot
figure
bode(1/W1)
hold on;
bode(1/W2)
hold on;
bode(1/W3)
hold on;

%%%%%%%%%%%%%%%
Gd = 1;
% disturbance with no reference (Servo)
d = 0;
r = 1;

y = T*r + S*Gd*d;

% time variables
t = 0:0.1:100;
w = [0.1 1];

figure;
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out,t_out,x] = lsim(y, u_in, t);
    subplot(2,1,i)
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    plot(t_out,y_out,'LineWidth',1.25);
    hold on;
    legend('Reference','Response')
    xlabel('time(sec)');
    ylabel('Position');
    title(sprintf('%.2f rad/s sinusoidal input for servo system',w(
   i)));
end

% Uncertainty
% Ureal for mass
m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
% Ureal for actuator gain
U0_ureal = ureal('uo_ureal',1,'Range',[0.5 1.5]);
% Ureal for time constant
tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2.5]);
% Time delay value
theta_ureal = 0.1;

% Plant
G_ureal = 1/(m_ureal*s^2);
% Actuator
U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);

```

```
L_ureal = (U_ureal*G_ureal) * Kinf;
% Sensitivity function
S_ureal = 1 / (1 + L_ureal);
% Comp. Sensitivity function
T_ureal = L_ureal / (1 + L_ureal);

% plot the uncertain reponse of the controller
figure;

y_ureal = T_ureal * r + S_ureal * d;
step(y_ureal)
xlabel('Time');
ylabel('x');
title('Servo step response for uncertain model');

% time variables
t = 0:0.1:100;
w = [0.1 1];

figure;
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
    subplot(2,1,i)
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    plot(t_out,y_out_ureal,'LineWidth',1.25);
    hold on;
    legend('Reference','Response')
    xlabel('time(sec)');
    ylabel('Speed');
    title(sprintf('%.2f rad/s sinusoidal input for servo system
   with uncertainty',w(i)));

end

%% Part B) Servo Problem with uncertainty -- Measurement is speed
% Define Plant
num=1;
den=[1 1e-3];
G0=tf(num,den);

s = tf('s');

% Parametric uncertainty
% Upper and lower bound for mass
m_upperBar = 1.2;
m_lowerBar = 0.8;

% Relative magnitude of the gain uncertainty
rm = (m_upperBar-m_lowerBar)/(m_upperBar+m_lowerBar);

% Any stable transfer function which at each frequency is less
% than or equal to one in magnitude.
delta = 1;
Gp = G0*(1+rm*delta);
```

```
% Actuator
U0 = 2.5/(2.5*s+1);
% Simple first-order weight
T = 4;
wI1 = (T*s + 0.2)/((T/2.5)*s+1);
deltaI = 1/(5*s+1)^3;
% Multiply with correction factor to lift the gain
wI = wI1*tf([1 1.6 1],[1 1.4 1]);
Up = U0*(1+wI*deltaI);

% Plant with actuator
G = Up*Gp;

% Selecting weighting functions
% W1 is large frequency band where accurate tarcking/disturbance
    rejection
% is required
wn = 1e-3;
num_W1 = 2000^2;
den_W1 = [1/wn^2 2*0.7/wn 1];
W1 = tf(num_W1,den_W1);

% W2 is constant
num_W2 = 1e-3;
den_W2 = 1;
W2 = tf(num_W2,den_W2);

% W3 is large frequency band where model is uncertain
num_W3 = 5e-8*[1/wn^2 2*0.7/wn 1];
den_W3 = [1/200^2 2*0.7/200 1];
W3 = tf(num_W3,den_W3);

% Calculation of augmented plant:
P = augw(G,W1,W2,W3);

% Calculation of optimal controller
nmeas=1;
ncon=1;
[Kinf,Pcl,gamma_opt] = hinfsyn(P,nmeas,ncon);

L = G*Kinf;
% Sensitivity function
S = 1/(1 + L);
% Comp. Sensitivity function
T = L/(1 + L);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Figures
% Bode Plot
bode(1/W1)
hold on;
bode(1/W2)
hold on;
bode(1/W3)
hold on;

%%%%%%%%%%%%%%%
```

```
Gd = 1;
% disturbance with no reference (Servo)
d = 0;
r = 1;

y = T*r + S*Gd*d;

% time variables
t = 0:0.1:100;
w = [0.1 1];

figure;
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out,t_out,x] = lsim(y, u_in, t);
    subplot(2,1,i)
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    plot(t_out,y_out,'LineWidth',1.25);
    hold on;
    legend('Reference','Response')
    xlabel('time(sec)');
    ylabel('Speed');
    title(sprintf('%.2f rad/s sinusoidal input for servo system',w(
   i)));

end


% Uncertainty
% Ureal for mass
m_ureal = ureal('m_ureal',1,'Range',[0.8 1.2]);
% Ureal for actuator gain
U0_ureal = ureal('uo_ureal',1,'Range',[0.5 1.5]);
% Ureal for time constant
tau_ureal = ureal('tau_ureal',1,'Range',[0.5 2.5]);
% Time delay value
theta_ureal = 0.1;

% Plant
G_ureal = 1/(m_ureal*s);
% Actuator
U_ureal = U0_ureal * exp(-theta_ureal*s)/(tau_ureal*s + 1);

L_ureal = (U_ureal*G_ureal) * Kinf;
% Sensitivity function
S_ureal = 1 / (1 + L_ureal);
% Comp. Sensitivity function
T_ureal = L_ureal / (1 + L_ureal);

% plot the uncertain reponse of the controller
figure;

y_ureal = T_ureal * r + S_ureal * d;
step(y_ureal)
xlabel('Time');
```

```
ylabel('xDot');
title('Servo step response for uncertain model');

% time variables
t = 0:0.1:100;
w = [0.1 1];

figure;
for i = 1:length(w)
    u_in = sin(w(i).*t);
    [y_out_ureal,t_out,x] = lsim(y_ureal, u_in, t);
    subplot(2,1,i)
    plot(t_out, u_in, 'LineWidth',1.25)
    hold on;
    plot(t_out,y_out_ureal,'LineWidth',1.25);
    hold on;
    legend('Reference','Response')
    xlabel('time(sec)');
    ylabel('Speed');
    title(sprintf('%.2f rad/s sinusoidal input for servo system with Uncertainty',w(i)));

end
```

Part-B Code for Position Measurement

```
clc;clear; close all;
%% System Model
% Mass
m = 3;
% Time constant
tau = 1;
% Perturbation for mass
pm = 0.4;
% Perturbation for tau
ptau = 0.3;
%% State-space representation for G_mds
A = [ 0 1 0; 0 0 1/m ; 0 0 -1/tau];
B1 = [ 0 0; -pm 0 ;0 -ptau];
B2 = [ 0; 0; 1/tau];
C1 = [0 0 1/m; 0 0 -1\tau];
C2 = [ 1 0 0];
D11 = [-pm 0; 0 -ptau];
D12 = [0; 1\tau];
D21 = [0 0];
D22 = 0;

G = pck(A,[B1,B2],[C1;C2],[D11 D12;D21 D22]);

% This part is implemented to not see a error!!
% Unpack G and see the eignevalues
[a,b,c,d] = unpck(G);
eig(a)
% We do not want zero valued eigenvalue!!
% Zero eigenvalue leads to error
% Therefore, we add a very small number to change the zero
```

```
    eigenvalue
% situation
a(2,1)= 0.0001;
eig(a)
G = pck(a,b,c,d);

%% Frequency responses of the perturbed plants
omega = logspace(-1,1,100);
[delta1,delta2] = ndgrid([-1 0 1],[-1 0 1]);
for j = 1:9
    delta = diag([delta1(j),delta2(j)]);
    olp = starp(delta,G);
    olp_ic = sel(olp,1,1);
    olp_g = frsp(olp_ic,omega);
    figure(1)
    vplot('bode',olp_g,'c-')
    subplot(2,1,1)
    hold on
    subplot(2,1,2)
    hold on
end
subplot(2,1,1)
olp_ic = sel(G,3,3);
olp_g = frsp(olp_ic,omega);
vplot('bode',olp_g,'r--')
subplot(2,1,1)
title('BODE PLOTS OF PERTURBED PLANTS')
hold off
subplot(2,1,2)
hold off

%%  Design Requirements of Closed-loop System
% Robust performance:
% Define the chosen weighting functions
nuWp = [1 1.8 10];
dnWp = [1 8 0.01];
gainWp = 0.95;
Wp = nd2sys(nuWp,dnWp,gainWp);
nuWu = 1;
dnWu = 1;
gainWu = 50^(-3);
Wu = nd2sys(nuWu,dnWu,gainWu);
% Calculate the inverse weighting function
omega = logspace(-4,4,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);
figure
vplot('liv,lm',Wpi_g)
title('Inverse of Performance Weighting Function')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

%% System Interconnections
systemnames = ' G Wp Wu ';
inputvar = '[ pert{2}; dist; control ]';
outputvar = '[ G(1:2); Wp; -Wu; -G(3)-dist ]';
```

```
input_to_G = '[ pert; control ]';
input_to_Wp = '[ G(3)+dist ]';
input_to_Wu = '[ control ]';
sysoutname = 'sys_ic';
cleanupsysic = 'no';
% create the structure of open-loop systems
sysic

% To analyse the open-loop system, the following commands can be used.
minfo(sys_ic)
spoles(sys_ic)
spoles(Wp)

% The model of the open-loop system with uncertainties is set
systemnames = ' G ';
inputvar = '[ pert{2}; ref; dist; control ]';
outputvar = '[ G(1:2); G(3)+dist; ref - G(3) - dist ]';
input_to_G = '[ pert; control ]';
sysoutname = 'sim_ic';
cleanupsysic = 'yes';
sysic

%% 8.5 Suboptimal H  Controller Design
% number of measurements
nmeas = 1;
% number of controls
ncon = 1;
% lower bound of bisection
gmin = 1;
% upper bound of bisection
gmax = 10;
% absolute tolerance for the bisection method
tol = 0.001;
% open-loop interconnection is saved in the variable "hin_ic"
hin_ic = sel(sys_ic,3:5,3:4);
% K_hin: controller (matrix of type SYSTEM)
% clp: closed-loop system (matrix of type SYSTEM)
[K_hin,clp] = hinfsyn(hin_ic,nmeas,ncon,gmin,gmax,tol);

%% Analysis of Closed-loop System with Khin
% Singular values of the closed-loop system with Khin
minfo(K_hin)
spoles(K_hin)
omega = logspace(-2,6,100);
clp_g = frsp(clp,omega);
vplot('liv,lm',vsvd(clp_g))
title('Singular Value Plot of clp')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Sensitivity function with Khin plot
K = K_hin;
clp = starp(sim_ic,K);

% inverse performance weighting function
```

```
omega = logspace(-4,2,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);

% sensitivity function
sen_loop = sel(clp,3,4);
sen_g = frsp(sen_loop,omega);
vplot('liv,lm',Wpi_g,'m--',vnorm(sen_g),'y-')
title('CLOSED-LOOP SENSITIVITY FUNCTION')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Robust stability analysis of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1;-1 1];
rob_stab = sel(clp_g,[1:3],[1:3]);
pdim = ynum(rob_stab);
fixl = [eye(pdim); 0.1*eye(pdim)]; % 1% Complex
fixr = fixl';
blkrs = [blkrsR; abs(blkrsR)];
clp_mix = mmult(fixl,rob_stab,fixr);
[rbnds,rowd,sens,rowp,rowg] = mu(clp_mix,blkrs);
disp(' ')
disp(['mu-robust stability: ' ...
num2str(pkvnorm(sel(rbnds,1,1)))])
disp(' ')
vplot('liv,lm',sel(rbnds,1,1),'y--',sel(rbnds,1,2),'m-', ...
vnorm(rob_stab),'c-.')
title('ROBUST STABILITY')
xlabel('Frequency (rad/s)')
ylabel('mu')

% Nominal and robust performance of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1];

% Nominal performance
nom_perf = sel(clp_g,3,3);

% Robust performance
rob_perf = clp_g;
blkrp = [blkrsR;[1 2]];
bndsrp = mu(rob_perf,blkrp);
vplot('liv,lm',vnorm(nom_perf),'y-',sel(bndsrp,1,1),'m--',...
sel(bndsrp,1,2),'c--')
tmp1 = 'NOMINAL PERFORMANCE (solid) and';
tmp2 = ' ROBUST PERFORMANCE (dashed)';
title([tmp1 tmp2])
xlabel('Frequency (rad/s)')
disp(' ')
disp(['mu-robust performance: ' ...
num2str(pkvnorm(sel(bndsrp,1,1)))])
```

```
disp(' ')

% Transient response to reference/disturbance input
clp = starp(sim_ic,K_hin);
timedata = [0 20 40];
stepdata = [1 0 1];
dist = 0;
ref = step_tr(timedata,stepdata,0.1,60);
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',ref,'r--')
title('CLOSED-LOOP TRANSIENT RESPONSE')
xlabel('Time (secs)')
ylabel('y (m)')

% Response to the disturbance
timedata = [0 20 40];
stepdata = [1 0 1];
dist = step_tr(timedata,stepdata,0.1,60);
ref = 0;
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',dist,'r--')
title('TRANSIENT RESPONSE TO THE DISTURBANCE')
xlabel('Time (secs)')
ylabel('y (m)')

%% Servo Problem measurement speed
clp = starp(sim_ic,K_hin);
w = [0.1, 1];
dist = 0;
figure;
for i = 1:length(w)
    freq = w(i);
    ref = sin_tr (freq ,1 ,0.1 ,100) ;
    u = abv(0,0 ,ref , dist );
    y = trsp(clp ,u ,100 ,0.1) ;
    subplot (2,1,i);
    vplot ( sel(y,3,1),'b',ref ,'r')
    xlabel ('time (sec)');
    ylabel ('x');
    legend ('Response','Ref ')
    title (sprintf('Servo response for %.2f rad /s sinusoidal
   reference',w(i)));
end

%% Regulator Problem
clp = starp(sim_ic ,K_hin);
w = [0.1, 1];
ref = 0;
figure ;
for i = 1: length(w)
    freq = w(i);
    dist = sin_tr(freq ,1 ,0.1 ,100) ;
```

```
    u = abv (0,0 ,ref , dist );
    y = trsp (clp ,u ,100 ,0.1) ;

    % Sinusoidal disturbance for regulator system
    subplot (2 ,1 ,1);
    vplot (dist);
    hold on;
    xlabel ('time (sec)');
    ylabel ('disturbance');
    title ('Sinusoidal disturbance for regulator system');

    % Response for regulator system
    subplot (2 ,1 ,2)
    vplot (sel(y ,3 ,1))
    hold on;
    legendStr (i) = sprintf (" %.2f rad /s", w(i));
    xlabel ('time (sec)');
    ylabel ('x');
    title ('Response for regulator system');
end
subplot (2 ,1 ,1)
legend (legendStr);
subplot (2 ,1 ,2)
legend (legendStr);
```

Part-B Code for Speed Measurement

```
clc;clear; close all;
%% System Model
% Mass
m = 3;
% Time constant
tau = 1;
% Perturbation for mass
pm = 0.4;
% Perturbation for tau
ptau = 0.3;
%% State-space representation for G_mds
% System matrices
A = [0  1/m; 0 -1/tau];
B1 = [-pm 0 ;0 -ptau];
B2 = [0; 1/tau];
C1 = [0 1/m;0 -1\tau];
C2 = [1 0];
D11 = [-pm 0; 0 -ptau];
D12 = [0; 1\tau];
D21 = [0 0];
D22 = 0;

G = pck(A,[B1,B2],[C1;C2],[D11 D12;D21 D22]);

% This part is implemented to not see a error!!
% Unpack G and see the eignevalues
[a,b,c,d] = unpck(G);
eig(a)
% We do not want zero valued eigenvalue!!
```

```
% Zero eigenvalue leads to error
% Therefore, we add a very small number to change the zero
    eigenvalue
% situation
a(2,1)= 0.0001;
eig(a)
G = pck(a,b,c,d);

%% Frequency responses of the perturbed plants
omega = logspace(-1,1,100);
[delta1,delta2] = ndgrid([-1 0 1],[-1 0 1]);
for j = 1:9
    delta = diag([delta1(j),delta2(j)]);
    olp = starp(delta,G);
    olp_ic = sel(olp,1,1);
    olp_g = frsp(olp_ic,omega);
    figure(1)
    vplot('bode',olp_g,'c-')
    subplot(2,1,1)
    hold on
    subplot(2,1,2)
    hold on
end
subplot(2,1,1)
olp_ic = sel(G,3,3);
olp_g = frsp(olp_ic,omega);
vplot('bode',olp_g,'r--')
subplot(2,1,1)
title('BODE PLOTS OF PERTURBED PLANTS')
hold off
subplot(2,1,2)
hold off

%%  Design Requirements of Closed-loop System
% Robust performance:
% Define the chosen weighting functions
nuWp = [1 1.8 10];
dnWp = [1 8 0.01];
gainWp = 0.95;
Wp = nd2sys(nuWp,dnWp,gainWp);
nuWu = 1;
dnWu = 1;
gainWu = 50^(-3);
Wu = nd2sys(nuWu,dnWu,gainWu);
% Calculate the inverse weighting function
omega = logspace(-4,4,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);
figure
vplot('liv,lm',Wpi_g)
title('Inverse of Performance Weighting Function')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

%% System Interconnections
systemnames = ' G Wp Wu ';
```

```
inputvar = '[ pert{2}; dist; control ]';
outputvar = '[ G(1:2); Wp; -Wu; -G(3)-dist ]';
input_to_G = '[ pert; control ]';
input_to_Wp = '[ G(3)+dist ]';
input_to_Wu = '[ control ]';
sysoutname = 'sys_ic';
cleanupsysic = 'no';
% create the structure of open-loop systems
sysic

% To analyse the open-loop system, the following commands can be
    used.
minfo(sys_ic)
spoles(sys_ic)
spoles(Wp)

% The model of the open-loop system with uncertainties is set
systemnames = ' G ';
inputvar = '[ pert{2}; ref; dist; control ]';
outputvar = '[ G(1:2); G(3)+dist; ref - G(3) - dist ]';
input_to_G = '[ pert; control ]';
sysoutname = 'sim_ic';
cleanupsysic = 'yes';
sysic

%% 8.5 Suboptimal  H   Controller Design
% number of measurements
nmeas = 1;
% number of controls
ncon = 1;
% lower bound of bisection
gmin = 1;
% upper bound of bisection
gmax = 10;
% absolute tolerance for the bisection method
tol = 0.001;
% open-loop interconnection is saved in the variable "hin_ic"
hin_ic = sel(sys_ic,3:5,3:4);
% K_hin: controller (matrix of type SYSTEM)
% clp: closed-loop system (matrix of type SYSTEM)
[K_hin,clp] = hinfsyn(hin_ic,nmeas,ncon,gmin,gmax,tol);

%% Analysis of Closed-loop System with Khin
% Singular values of the closed-loop system with Khin
minfo(K_hin)
spoles(K_hin)
omega = logspace(-2,6,100);
clp_g = frsp(clp,omega);
vplot('liv,lm',vsvd(clp_g))
title('Singular Value Plot of clp')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Sensitivity function with Khin plot
K = K_hin;
clp = starp(sim_ic,K);
```

```
% inverse performance weighting function
omega = logspace(-4,2,100);
Wp_g = frsp(Wp,omega);
Wpi_g = minv(Wp_g);

% sensitivity function
sen_loop = sel(clp,3,4);
sen_g = frsp(sen_loop,omega);
vplot('liv,lm',Wpi_g,'m--',vnorm(sen_g),'y-')
title('CLOSED-LOOP SENSITIVITY FUNCTION')
xlabel('Frequency (rad/sec)')
ylabel('Magnitude')

% Robust stability analysis of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1;-1 1];
rob_stab = sel(clp_g,[1:3],[1:3]);
pdim = ynum(rob_stab);
fixl = [eye(pdim); 0.1*eye(pdim)]; % 1% Complex
fixr = fixl';
blkrs = [blkrsR; abs(blkrsR)];
clp_mix = mmult(fixl,rob_stab,fixr);
[rbnds,rowd,sens,rowp,rowg] = mu(clp_mix,blkrs);
disp(' ')
disp(['mu-robust stability: ' ...
num2str(pkvnorm(sel(rbnds,1,1)))])
disp(' ')
vplot('liv,lm',sel(rbnds,1,1),'y--',sel(rbnds,1,2),'m-', ...
vnorm(rob_stab),'c-.')
title('ROBUST STABILITY')
xlabel('Frequency (rad/s)')
ylabel('mu')

% Nominal and robust performance of Khin
clp_ic = starp(sys_ic,K);
omega = logspace(-1,2,100);
clp_g = frsp(clp_ic,omega);
blkrsR = [-1 1;-1 1];

% Nominal performance
nom_perf = sel(clp_g,3,3);

% Robust performance
rob_perf = clp_g;
blkrp = [blkrsR;[1 2]];
bndsrp = mu(rob_perf,blkrp);
vplot('liv,lm',vnorm(nom_perf),'y-',sel(bndsrp,1,1),'m--',...
sel(bndsrp,1,2),'c--')
tmp1 = 'NOMINAL PERFORMANCE (solid) and';
tmp2 = ' ROBUST PERFORMANCE (dashed)';
title([tmp1 tmp2])
xlabel('Frequency (rad/s)')
disp(' ')
```

```
disp(['mu-robust performance: ' ...
num2str(pkvnorm(sel(bndsrp,1,1)))])
disp(' ')

% Transient response to reference/disturbance input
clp = starp(sim_ic,K_hin);
timedata = [0 20 40];
stepdata = [1 0 1];
dist = 0;
ref = step_tr(timedata,stepdata,0.1,60);
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',ref,'r--')
title('CLOSED-LOOP TRANSIENT RESPONSE')
xlabel('Time (secs)')
ylabel('y (m)')

% Response to the disturbance
timedata = [0 20 40];
stepdata = [1 0 1];
dist = step_tr(timedata,stepdata,0.1,60);
ref = 0;
u = abv(0,0,ref,dist);
y = trsp(clp,u,60,0.1);
figure
vplot(sel(y,3,1),'y-',dist,'r--')
title('TRANSIENT RESPONSE TO THE DISTURBANCE')
xlabel('Time (secs)')
ylabel('y (m)')

%% Servo Problem measurement speed
clp = starp(sim_ic,K_hin);
w = [0.1, 1];
dist = 0;
figure;
for i = 1:length(w)
    freq = w(i);
    ref = sin_tr (freq ,1 ,0.1 ,100) ;
    u = abv(0,0 ,ref , dist );
    y = trsp(clp ,u ,100 ,0.1) ;
    subplot (2,1,i);
    vplot (sel(y,3,1),'b',ref ,'r')
    xlabel ('time (sec)');
    ylabel ('xDot');
    legend ('Response','Ref ')
    title (sprintf('Servo response for %.2f rad /s sinusoidal
   reference',w(i)));
end

%% Regulator Problem
clp = starp(sim_ic ,K_hin);
w = [0.1, 1];
ref = 0;
figure ;
for i = 1: length(w)
```

```
    freq = w(i);
    dist = sin_tr(freq ,1 ,0.1 ,100) ;
    u = abv (0,0 ,ref , dist );
    y = trsp (clp ,u ,100 ,0.1) ;

    % Sinusoidal disturbance for regulator system
    subplot (2 ,1 ,1);
    vplot (dist);
    hold on;
    xlabel ('time (sec)');
    ylabel ('disturbance');
    title ('Sinusoidal disturbance for regulator system');

    % Response for regulator system
    subplot (2 ,1 ,2)
    vplot (sel(y ,3 ,1))
    hold on;
    legendStr (i) = sprintf (" %.2f rad /s", w(i));
    xlabel ('time (sec)');
    ylabel ('xDot');
    title ('Response for regulator system');
end
subplot (2 ,1 ,1)
legend (legendStr);
subplot (2 ,1 ,2)
legend (legendStr);
```