MAE550 STR : Vibration Control of Slewing Structures

Mid-Term Exam, Spring 04

by

Student : Kok-Lam Lai

Person Number : 2838-4658

Email: klai2@eng.buffalo.edu

Instructor : Prof. Tarunraj Singh

State University of New York at Buffalo

Buffalo, New York 14260

March 29, 2004

Contents

1 st Problem - Time Delay Filter

1.1 Part 1 - Pre-filter

The original linear system with nominally $\tau_{nom} = 0.5$ and $K_{nom} = 2.0$

$$\dot{y} + \frac{1}{\tau} y = Ku \tag{1.1}$$

or in transfer function form

$$\frac{Y}{U} = \frac{K}{s + \frac{1}{\tau}} \tag{1.2}$$

which clearly with zero $s = -1/\tau$. The form of time-delay pre-filter is given by [1]

$$A_0 + e^{sT} = 0 \tag{1.3}$$

Figure 1.1 shows the single time-delay pre-filter controlled of our system. Since our original system is a first order system, we may choose whatever T we would like. Let's choose $T = 1$, now A_0 is simply

$$\begin{aligned} A_0 &= -e^{T/\tau} \\ &= -e^2 \end{aligned} \tag{1.4}$$

Applying the final value theorem [2] with step input

$$\lim_{s \to 0} s \frac{1}{s} \frac{(A_0 + e^{-sT})K}{s + \frac{1}{\tau}} = (A_0 + 1)K\tau \tag{1.5}$$

thus we need to normalize the time-delay controller with $(A_0 + 1)K\tau$ to ensure the final

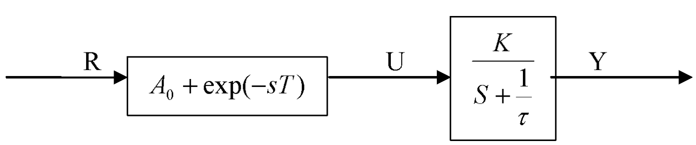


Figure 1.1: Single time-delay controlled

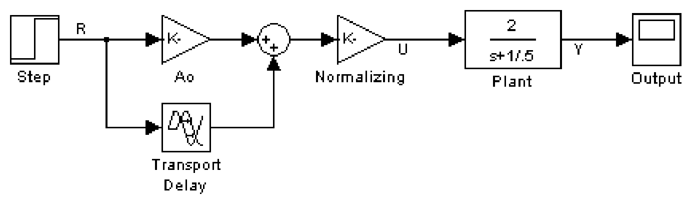


Figure 1.2: Simulink Implementation of the Pre-filter Controller

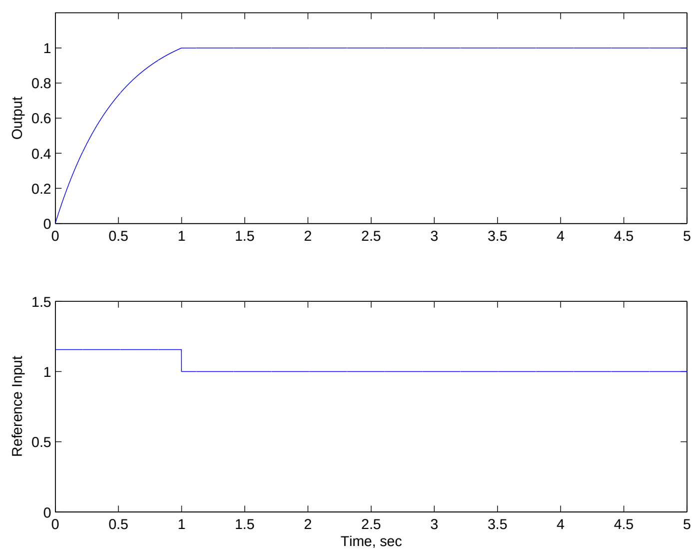


Figure 1.3: Part 1- Time-Delay Filter Input and Output

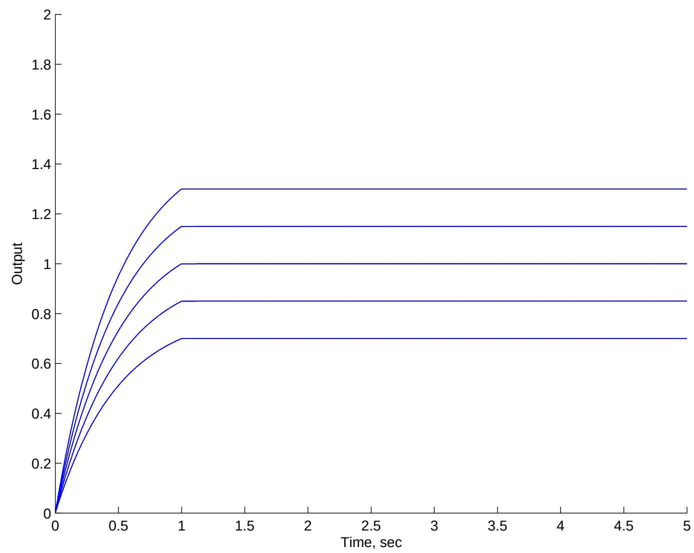


Figure 1.4: Part 2 - Output plot for τ at nominal value, K varying between $\pm 30\%$

output would be one. So the final form of the time-delay controller transfer function is

$$\frac{A_0 + e^{-sT}}{(A_0 + 1)K\tau} \tag{1.6}$$

with $A_0 = -e^2$, $T = 1$, $K = 2.0$ and $\tau = 0.5$. Figure 1.2 shows the Simulink implementation of this controller. However, we choose to reprogram it in Matlab for easier evaluation of Parts 2 and 3 in the next sections. Figure 1.3 shows the reference input and output of the system.

1.2 Part 2 - Sensitivity

In this section we investigate the output under varying K and τ as shown in Figs. 1.4 and 1.5 respectively. The residual is chosen to be Residual $= (y_f - 1)^2$ where y_f is the final value of the output. The output plot when both K and τ varying together is shown in Fig. 1.6. The varyings are between $\pm 30\%$ of their respective nominal values. Figure **??**p1d-3d-both-change) is an interesting residual plot of both τ and K varying between $\pm 30\%$ off their respective nominal values.

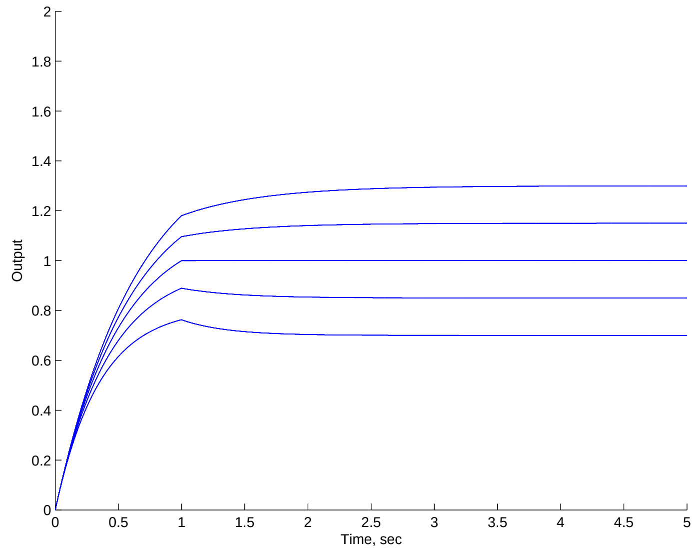


Figure 1.5: Part 2 - Output plot for K at nominal value, τ varying between $\pm 30\%$

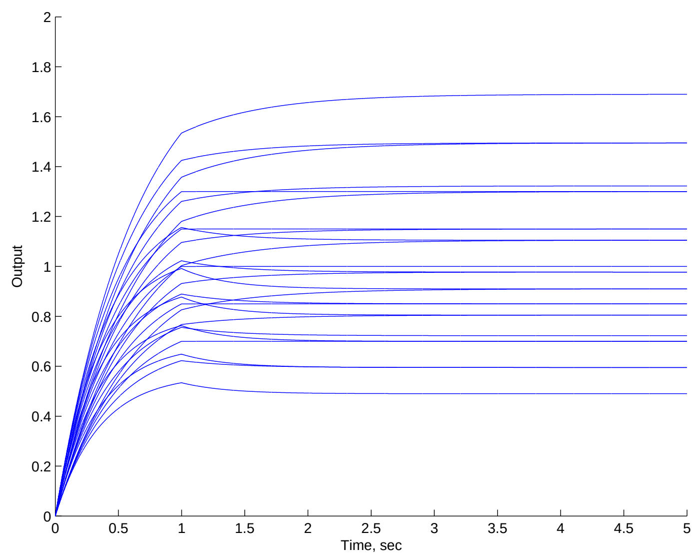
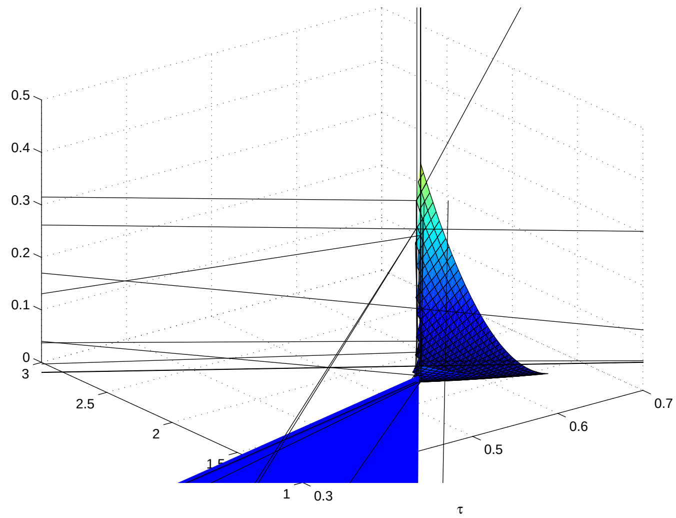


Figure 1.6: Part 2 - Output plot for both τ and K varying between $\pm 30\%$



0.5
0.4
0.3
0.2
0.1
0
3
2.5
2
1.5
1
0.3
0.5
0.6
0.7
τ

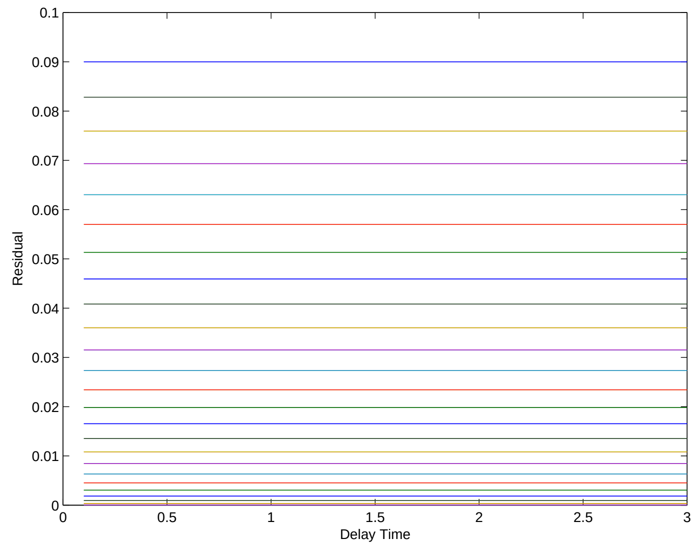


Figure 1.9: Part 3 - Residual plot of delay-time varying

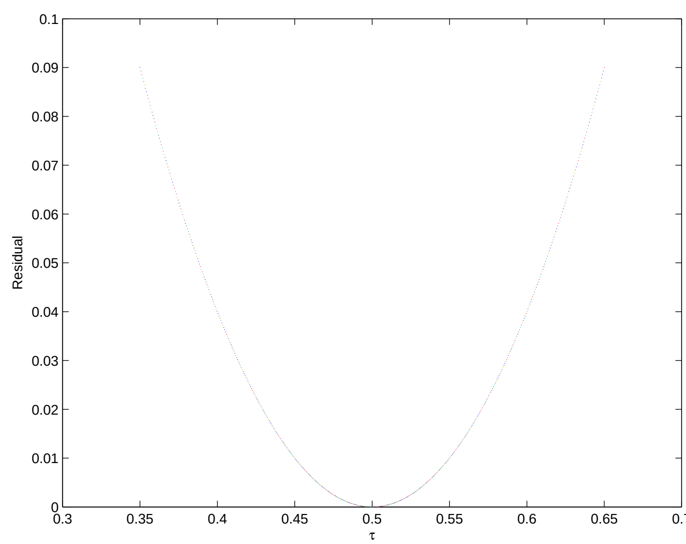


Figure 1.10: Part 3 - Residual plot of τ varying

1.3 Part 3 - Effect to Uncertainty in delay-time and τ

In this section we examine the effect of varying time and τ on residual. Figure 1.8 shows the residual for different delay-time and τ. Figures 1.9 and 1.10 show clearer the relationship between varying each of these components, delay-time and τ, on residual. As we can see from Fig. 1.9, delay-time has no effect on the residual. However, varying τ varies the residual as in Fig. 1.10 as we have shown in Part 2 too. This does not come into surprise however, since our system is a first-order system and we have the freedom to choose our delay-time.

2 $^{\text{nd}}$ Problem - LQR, Root Locus

2.1 Part 1 - Velocity Hold

The system, medium-size helicopter (OH-6A), near hover are modelled by

$$\begin{Bmatrix} \dot{u} \\ \dot{q} \\ \dot{\theta} \end{Bmatrix} = \begin{bmatrix} -0.026 & 0.013 & -0.322 \\ 1.26 & -1.765 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} u \\ q \\ \theta \end{Bmatrix} + \begin{bmatrix} 0.086 \\ -7.41 \\ 0 \end{bmatrix} \delta_{cy} \tag{2.1}$$

where u = forward velocity (ft/sec), q = pitch rate (crad/sec), θ = pitch angle (crad/sec), δ_{cy} = longitudinal cyclic pitch (deci-inches).

The velocity-hold performance index being considered is

$$J = \int_0^\infty (Qu^2 + \delta_{cy}^2) dt \tag{2.2}$$

Which is a classic LQR problem. The root locus plot of closed-loop poles vs. Q is shown in Fig. 2.1

2.2 Part 2 - Position Hold

We augment the original system to include the position, x, into the equation of motion

$$\begin{Bmatrix} \dot{x} \\ \dot{u} \\ \dot{q} \\ \dot{\theta} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.026 & 0.013 & -0.322 \\ 0 & 1.26 & -1.765 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x \\ u \\ q \\ \theta \end{Bmatrix} + \begin{bmatrix} 0 \\ 0.086 \\ -7.41 \\ 0 \end{bmatrix} \delta_{cy} \tag{2.3}$$

The position-hold performance index being considered is

$$J = \int_0^\infty (Qx^2 + \delta_{cy}^2) dt \tag{2.4}$$

Which is also a classic LQR problem. The root locus plot of closed-loop poles vs. Q is shown in Fig. 2.2

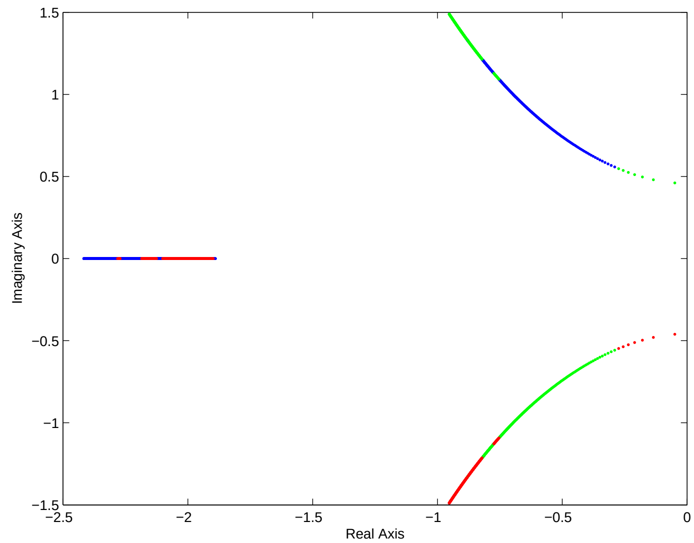


Figure 2.1: Part 1 - Root Locus for $Q = 1$ to $Q = 10$

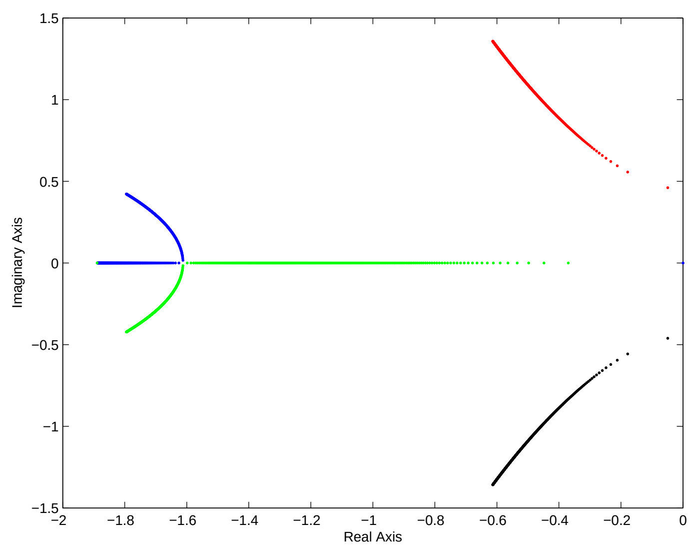


Figure 2.2: Part 2 - Root Locus for $Q = 1$ to $Q = 10$

3 rd Problem - Two-Mass Spring System, Minimum Power

This problem is based on two-mass spring system shown in Fig. 3.1 with nominal parameters as

$$\begin{Bmatrix} m_1^{nom} \\ m_2^{nom} \\ k^{nom} \\ c^{nom} \end{Bmatrix} = \begin{Bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 0 \end{Bmatrix} \tag{3.1}$$

Thus the equation of motion of the two-mass spring system is

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{M} \begin{Bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{Bmatrix} + \underbrace{\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}}_{K} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \underbrace{\begin{Bmatrix} 1 \\ 0 \end{Bmatrix}}_{P} u \tag{3.2}$$

with initial and final conditions

$$\begin{Bmatrix} y_1 \\ y_2 \\ \dot{y}_1 \\ \dot{y}_2 \end{Bmatrix}(0) = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = y_0 \qquad \text{and} \qquad \begin{Bmatrix} y_1 \\ y_2 \\ \dot{y}_1 \\ \dot{y}_2 \end{Bmatrix}(\pi) = \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{Bmatrix} = y_{tf} \tag{3.3}$$

Using similarity transformation

$$\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \underbrace{\begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}}_{V} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} \tag{3.4}$$

where V is the eigenvectors of $M^{-1}K$. Then eqn. 3.2 the equation of motion becomes

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = - \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} u \tag{3.5}$$

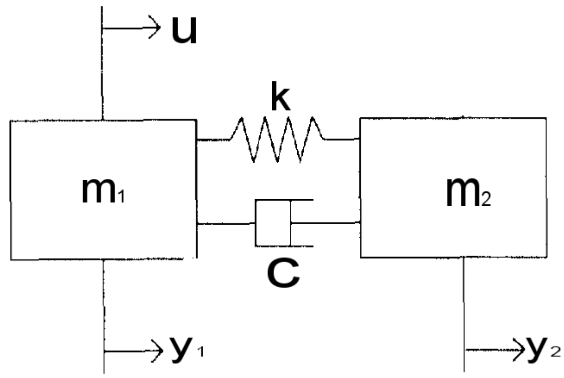


Figure 3.1: Two-mass spring system

which in state-space form is

$$\begin{Bmatrix} \dot{x}_1 \\ \ddot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_2 \end{Bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -2 & 0 \end{bmatrix}}_{\mathbb{A}} \begin{Bmatrix} x_1 \\ \dot{x}_1 \\ x_2 \\ \dot{x}_2 \end{Bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix}}_{\mathbb{B}} u \tag{3.6}$$

Now the boundary condition of eqn. 3.3 becomes

$$\begin{Bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} (0) = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = x_0 \qquad \text{and} \qquad \begin{Bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} (\pi) = \begin{Bmatrix} -1.4142 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = x_{tf} \tag{3.7}$$

The performance index being consider is

$$J = \frac{1}{2} \int_0^{\pi} \dot{u}^2 dt \tag{3.8}$$

Equation 3.6 could be written for boundary form as (also from eqn. (1.289)[3])

$$\exp(-\mathbb{A} t_f) x_{tf} - x_0 = \int_0^{t_f} \exp(-\mathbb{A} t) \mathbb{B} u(t) dt \tag{3.9}$$

by incorporating eqn. 3.9 into eqn. 3.8 we obtain the augmented cost function

$$\begin{aligned} J_a &= \frac{1}{2}\int_0^{\pi} \dot{u}^2 dt + \boldsymbol{\lambda}^T \left(\exp(-\mathbb{A}t_f) y_{tf} - y_0 - \int_0^{t_f} \exp(-\mathbb{A}t)\mathbb{B}u(t)dt \right) \\ &= \int_0^{\pi} \left\{ \frac{1}{2}\dot{u}^2 - \boldsymbol{\lambda}^T \left[\exp(-\mathbb{A}t)\mathbb{B}u(t) \right] \right\} dt + \boldsymbol{\lambda}^T \left(\exp(-\mathbb{A}t_f) y_{tf} - y_0 \right) \end{aligned} \tag{3.10}$$

Notice that the terms outside the integration are constants and do not contribute to the variation. We now employ calculus of variation and perform first-order variation on eqn. 3.10

$$\delta J_a = \int_0^{\pi} \left\{ \dot{u}\delta\dot{u} - \boldsymbol{\lambda}^T \exp(-\mathbb{A}t)\mathbb{B}\delta u \right\} dt \tag{3.11}$$

Now we perform integration-by-part on the term $\int_0^{\pi} \dot{u}\delta\dot{u}\, dt$ with

$$\begin{aligned} \int \mu d\nu &= \mu\nu - \int \nu d\mu \\ \mu = \dot{u} & \qquad d\nu = \delta\dot{u}dt \\ d\mu = \ddot{u}dt & \qquad \nu = \delta u \end{aligned} \tag{3.12}$$

thus it becomes

$$\int_0^{\pi} \dot{u}\delta\dot{u}\, dt = \left[\dot{u}\delta u\right]_0^{\pi} - \int_0^{\pi} \ddot{u}\delta u\, dt \tag{3.13}$$

Thus eqn. 3.11 now becomes [4] [5]

$$\delta J_a = \int_0^{\pi} \left\{ -\frac{d^2u}{dt^2} - \boldsymbol{\lambda}^T \exp(-\mathbb{A}t)\mathbb{B} \right\} \delta u dt + \left[\frac{du}{dt}\delta u \right]_0^{\pi} \tag{3.14}$$

In order for J_a to be an extremal, J_a must be equal to zero and thus the terms inside $\{\cdot\}$ must be equal to zero

$$\frac{d^2u}{dt^2} = -\boldsymbol{\lambda}^T \exp(-\mathbb{A}t)\mathbb{B} \tag{3.15}$$

The $\exp(-\mathbb{A}t)$ evaluated from Matlab is given to be

$$\exp(-\mathbb{A}t) = \begin{bmatrix} 1 & -t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\sqrt{2}t) & -\frac{\sqrt{2}}{2}\sin(\sqrt{2}t) \\ 0 & 0 & \sqrt{2}\sin(\sqrt{2}t) & \cos(\sqrt{2}t) \end{bmatrix} \tag{3.16}$$

Thus eqn. 3.15 becomes

$$\begin{aligned} \ddot{u}(t) = \frac{d^2u}{dt^2} &= -\boldsymbol{\lambda}^T \begin{bmatrix} \frac{\sqrt{2}}{2}t \\ -\frac{\sqrt{2}}{2} \\ \frac{1}{2}\sin(\sqrt{2}t) \\ -\frac{\sqrt{2}}{2}\cos(\sqrt{2}t) \end{bmatrix} \\ &= -\frac{\sqrt{2}}{2}t\lambda_1 + \frac{\sqrt{2}}{2}\lambda_2 - \frac{1}{2}\sin(\sqrt{2}t)\lambda_3 + \frac{\sqrt{2}}{2}\cos(\sqrt{2}t)\lambda_4 \end{aligned} \tag{3.17}$$

integrate this equation w.r.t. to t once

$$\dot{u}(t) = -\frac{\sqrt{2}}{4}t^2\lambda_1 + \frac{\sqrt{2}}{2}t\lambda_2 + \frac{1}{2\sqrt{2}}\cos(\sqrt{2}t)\lambda_3 + \frac{1}{2}\sin(\sqrt{2}t)\lambda_4 + \lambda_5 \tag{3.18}$$

which λ_5 is the integration constant. And integrate this once more to get

$$u(t) = -\frac{\sqrt{2}}{12}t^3\lambda_1 + \frac{\sqrt{2}}{4}t^2\lambda_2 + \frac{1}{4}\sin(\sqrt{2}t)\lambda_3 - \frac{1}{2\sqrt{2}}\cos(\sqrt{2}t)\lambda_4 + t\lambda_5 + \lambda_6 \tag{3.19}$$

again, λ_6 is the integration constant. Note that now our $\boldsymbol{\lambda}$ vector contains 6 terms, $\lambda_1, ..., \lambda_6$.

It is required that the initial and final time $\dot{u}$ to be equal to zero in order to minimize the jerk. Therefore we have the boundary conditions

$$\dot{u}(0) = \frac{\sqrt{2}}{4}\lambda_3 + \lambda_5 \tag{3.20a}$$

$$\dot{u}(\pi) = -\frac{\sqrt{2}}{4}\pi^2\lambda_1 + \frac{\sqrt{2}}{2}\pi\lambda_2 + \frac{\sqrt{2}}{4}\cos(\sqrt{2}\pi)\lambda_3 + \frac{1}{2}\sin(\sqrt{2}\pi)\lambda_4 + \lambda_5 \tag{3.20b}$$

From eqn. 3.9

$$\begin{bmatrix} \exp(-\mathbb{A}\pi)x_{tf} \\ 0 \\ 0 \end{bmatrix} = S\boldsymbol{\lambda} \tag{3.21}$$

with

$$S = \text{Jacobian} \begin{bmatrix} \int_0^\pi \exp(-\mathbb{A}t)\mathbb{B}u(t)dt \\ \dot{u}(0) \\ \dot{u}(\pi) \end{bmatrix}_{w.r.t.\boldsymbol{\lambda}} \tag{3.22}$$

so $\boldsymbol{\lambda}$ is simply obtain from

$$\begin{aligned} \boldsymbol{\lambda} &= S^{-1} \begin{bmatrix} \exp(-\mathbb{A}\pi)x_{tf} \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -148.5931 \\ -233.4096 \\ -147.9705 \\ 137.4511 \\ 52.3155 \\ 53.5675 \end{bmatrix} \end{aligned} \tag{3.23}$$

and the control input and its rate becomes

$$\begin{aligned} u(t) = 17.5119t^3 - 82.5227t^2 - 36.9926\sin(\sqrt{2}t) - 48.5963\cos(\sqrt{2}t) \\ +52.3155t + 53.5675 \end{aligned} \tag{3.24a}$$

$$\begin{aligned} \dot{u}(t) = 52.5356t^2 - 165.0455t - 52.3155\cos(\sqrt{2}t) \\ +68.7255\sin(\sqrt{2}t)\lambda_4 + 52.3155 \end{aligned} \tag{3.24b}$$

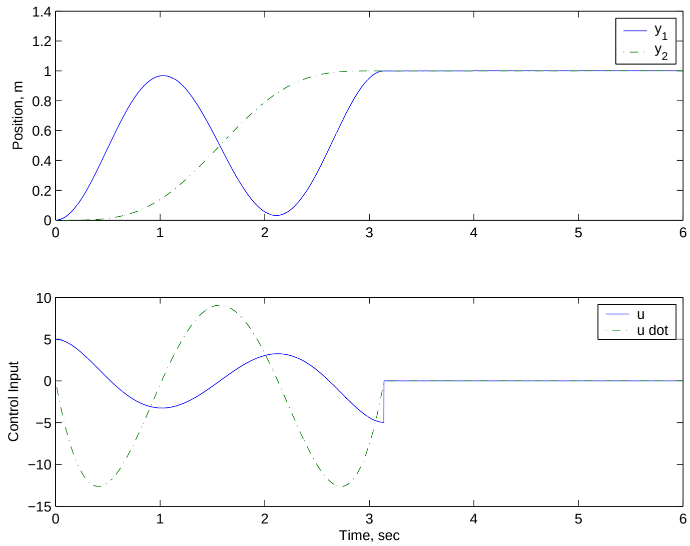


Figure 3.2: Output and Control Input

A Appendix - Matlab Code

Problem 1 - Part 1 - p1a.m

```
clear all; close all;

tau_nom=0.5; K_nom=2; m=5;
tau_vec=linspace(tau_nom-0.3*tau_nom,tau_nom+0.3*tau_nom,m);
K_vec=linspace(K_nom-0.3*K_nom,K_nom+0.3*K_nom,m);

A=[-1/tau_nom]; B=[K_nom]; C=[1]; D=[0];
sys=ss(A,B,C,D);                  % nominal
figure(10)
step(sys)

T      = 1;                       % Half-Time Periods, selected
A0     =-exp(T/tau_nom);          % coefficients for non-Robust
                                  % time-delay filter
A0n   = A0/((A0+1)*K_nom*tau_nom);    % normalizing
A1n   = 1/((A0+1)*K_nom*tau_nom);

t   = 0:0.001:10;
U1 = ones(size(t));         % without time-delay filter
U2 = zeros(size(t));        % with time-delay filter
U3 = U2;                    % with robust time-delay filter

indt       = find(t<T);     % Generating the input for
                            % non-Robust time-delay filter
U2(indt) = (1)*(1)*A0n;
indt       = find(t>=T);
U2(indt) = A0n+A1n;         % equal to one

sys=ss(A,B,C,D);
[Y1,T1,X1] = lsim(sys,U2,t);    % Nominal
figure(1)
subplot(211)
```

```
plot(T1,Y1)
axis([0 5 0 2])
a=axis;
ylabel('Output')
subplot(212)
plot(T1,U2)
axis([0 5 0 1.5])
b=axis;
ylabel('Reference Input')
xlabel('Time, sec')

for i=1:m          % for tau_vec
    for j=1:m      % for K_vec
        A=[-1/tau_vec(i)];
        B=[K_vec(j)];
        C=[1]; D=[0];

        sys_temp=ss(A,B,C,D);
        [Y2,T2,X2] = lsim(sys_temp,U2,t); % Non-Robust

        figure(5)
        hold on
        plot(T1,Y2)
        axis(a)
        ylabel('Output')
        hold off
        xlabel('Time, sec')
        pause(1)
    end
end
```

Problem 1 - Part 2 - p1b.m

```
clear all; close all;

tau_nom=0.5; K_nom=2; m=50;
tau_vec=linspace(tau_nom-0.3*tau_nom,tau_nom+0.3*tau_nom,m);
K_vec=linspace(K_nom-0.3*K_nom,K_nom+0.3*K_nom,m);

A=[-1/tau_nom]; B=[K_nom]; C=[1]; D=[0];
sys=ss(A,B,C,D);                 % nominal

```

```
T       = 1;                  % Half-Time Periods, selected
A0     =-exp(T/tau_nom);      % coefficients for non-Robust
                              % time-delay filter
A0n    = A0/((A0+1)*K_nom*tau_nom);      % normalizing
A1n    = 1/((A0+1)*K_nom*tau_nom);

t    = 0:0.001:10;
U1 = ones(size(t));       % without time-delay filter
U2 = zeros(size(t));      % with time-delay filter
U3 = U2;                  % with robust time-delay filter

indt       = find(t<T);    % Generating the input for
                           % non-Robust time-delay filter
U2(indt) = (1)*(1)*A0n;
indt       = find(t>=T);
U2(indt) = A0n+A1n;       % equal to one

% Part 2
tau_changes=zeros(m,2);        % tau varying
for  i=1:m
     A=[-1/tau_vec(i)];
     B=[K_nom];
     C=[1]; D=[0];
     sys_temp=ss(A,B,C,D);
     [Y2,T2,X2] = lsim(sys_temp,U2,t);     % Non-Robust
     res = (Y2(end,1)-1)^2;
     tau_changes(i,:)= [tau_vec(i) res];
end
figure(2)
plot(tau_changes(:,1),tau_changes(:,2))
ylabel('Residual')
xlabel('\tau')

K_changes=zeros(m,2);          % K varying
for  i=1:m
     A=[-1/tau_nom];
     B=[K_vec(i)];
     C=[1]; D=[0];
     sys_temp=ss(A,B,C,D);
     [Y2,T2,X2] = lsim(sys_temp,U2,t);     % Non-Robust
     res = (Y2(end,1)-1)^2;
```

```
    K_changes(i,:)= [K_vec(i) res];
end
figure(3)
plot(K_changes(:,1),K_changes(:,2))
ylabel('Residual')
xlabel('K')

tau_K_changes=zeros(m,m,3);          % K varying
wait = waitbar(0,'Please wait...');
for i=1:m         % for tau
    for j=1:m   % for K
        A=[-1/tau_vec(i)];
        B=[K_vec(j)];
        C=[1]; D=[0];
        sys_temp=ss(A,B,C,D);
        [Y2,T2,X2] = lsim(sys_temp,U2,t);    % Non-Robust
        res = (Y2(end,1)-1)^2;
        tau_K_changes(i,j,:)=[tau_vec(i) K_vec(j) res];
        waitbar(((i-1)*j+j)/m^2,wait)
    end
end
close(wait)
figure(4)
surf(tau_K_changes(:,:,1),tau_K_changes(:,:,2),...
    tau_K_changes(:,:,3))
xlabel('\tau')
ylabel('K')
zlabel('Residual')
```

Problem 1 - Part 3 - p1c.m

```
clear all; close all;

tau_nom=0.5; K_nom=2; m=50;
tau_vec=linspace(tau_nom-0.3*tau_nom,tau_nom+0.3*tau_nom,m);
K_vec=linspace(K_nom-0.3*K_nom,K_nom+0.3*K_nom,m);
t_vec=linspace(0.1,3,m);

A=[-1/tau_nom]; B=[K_nom]; C=[1]; D=[0];
sys=ss(A,B,C,D);      % nominal

% Part 3
```

```
t_tau_changes=zeros(m,m,4);      % tau varying
wait = waitbar(0,'Please wait...');
for i=1:m         % for t_vec
    for j=1:m    % for tau_vec
        T     = t_vec(i);                % Half-Time Periods
        A0    = -exp(T/tau_vec(j));  % coefficients for
                                         % non-Robust
                                         % time-delay filter
        A0n   = A0/((A0+1)*K_nom*tau_nom);    % normalizing
        A1n   = 1/((A0+1)*K_nom*tau_nom);

        t    = 0:0.001:10;
        U1 = ones(size(t));       % without time-delay filter
        U2 = zeros(size(t));      % with time-delay filter
        U3 = U2;

        indt       = find(t<T);    % Generating the input for
                                   % non-Robust time-delay
                                   % filter
        U2(indt) = (1)*(1)*A0n;
        indt       = find(t>=T);
        U2(indt) = A0n+A1n;        % equal to one

        A=[-1/tau_vec(j)];
        B=[K_nom];
        C=[1]; D=[0];
        sys_temp=ss(A,B,C,D);
        [Y2,T2,X2] = lsim(sys_temp,U2,t);    % Non-Robust
        res = (Y2(end,1)-1)^2;
        t_tau_changes(i,j,:)= [t_vec(i) tau_vec(j) res A0n];
        waitbar(((i-1)*j+j)/m^2,wait)

        %figure(10)
        %plot(T2,Y2)
        %pause(.1)

    end
end
close(wait)
figure(1)
surf(t_tau_changes(:,:,1),t_tau_changes(:,:,2),...
```

```
    t_tau_changes(:,:,3))
xlabel('Delay Time')
ylabel('\tau')
zlabel('Residual')

figure(2)
plot(t_tau_changes(:,:,1),t_tau_changes(:,:,3))
xlabel('Delay Time')
ylabel('Residual')

figure(3)
plot(t_tau_changes(:,:,2),t_tau_changes(:,:,3))
xlabel('\tau')
ylabel('Residual')

figure(4)
plot(t_tau_changes(:,:,2),t_tau_changes(:,:,4))
xlabel('time')
ylabel('A0n')
```

Problem 2 - Part 1 - p2a.m

```
clear all; close all;

A = [ -0.026    0.013   -0.322
       1.26    -1.765    0
       0        1        0     ];
B = [ 0.086 -7.41 0 ]';
C = [1 0 0 ];
D = [0];

damp(eig(A))
R=1;
Q=zeros(3);
q_vec=0:.01:10;
m=length(q_vec);
locus_mat=zeros(m,7);

for i=1:m
    Q(1,1)=q_vec(i);
    [K,S,E]=lqr(A,B,Q,R);
    [Wn,Z,P]=damp(eig(A-B*K));
```

```
    locus_mat(i,:)=[q_vec(i) real(P(1,1)) imag(P(1,1)) ...
      real(P(2,1)) imag(P(2,1)) real(P(3,1)) imag(P(3,1))];
end

figure(1)
plot(locus_mat(:,2),locus_mat(:,3),'b.', locus_mat(:,4), ...
    locus_mat(:,5),'g.', locus_mat(:,6),locus_mat(:,7),'r.')
ylabel('Imaginary Axis')
xlabel('Real Axis')
a=axis;

figure(2)
subplot(311)
plot(locus_mat(:,2),locus_mat(:,3),'b.')
axis(a)
subplot(312)
plot(locus_mat(:,4),locus_mat(:,5),'g.')
axis(a)
subplot(313)
plot(locus_mat(:,6),locus_mat(:,7),'r.')
axis(a)
```

Problem 2 - Part 2 - p2b.m

```
clear all; close all;

A = [    0       1       0       0
         0    -0.026   0.013  -0.322
         0     1.26   -1.765    0
         0       0       1       0    ];
B = [ 0 0.086 -7.41 0 ]';
C = [1 0 0 0];
D = [0];

damp(eig(A))
R=1;
Q=zeros(4);
q_vec=0:.01:10;
m=length(q_vec);
locus_mat=zeros(m,9);

for i=1:m
```

```
    Q(1,1)=q_vec(i);
    [K,S,E]=lqr(A,B,Q,R);
    [Wn,Z,P]=damp(eig(A-B*K));
    locus_mat(i,:)=[q_vec(i) real(P(1,1)) imag(P(1,1)) ...
            real(P(2,1)) imag(P(2,1)) real(P(3,1)) ...
            imag(P(3,1)) real(P(4,1)) imag(P(4,1))];
end

figure(1)
plot(locus_mat(:,2),locus_mat(:,3),'b.', ...
    locus_mat(:,4),locus_mat(:,5),'g.', ...
    locus_mat(:,6),locus_mat(:,7),'r.', ...
    locus_mat(:,8),locus_mat(:,9),'k.')
ylabel('Imaginary Axis')
xlabel('Real Axis')
a=axis;

figure(2)
subplot(221)
plot(locus_mat(:,2),locus_mat(:,3),'b.')
axis(a)
subplot(222)
plot(locus_mat(:,4),locus_mat(:,5),'g.')
axis(a)
subplot(223)
plot(locus_mat(:,6),locus_mat(:,7),'r.')
axis(a)
subplot(224)
plot(locus_mat(:,8),locus_mat(:,9),'k.')
axis(a)
```

Problem 3 - p3a.m

```
clear all; close all;

m1=1.0; m2=1.0; k=1.0; c=0;

M = [m1 0;0 m1];
K = [k -k;-k k];

Ao = [zeros(2) eye(2); -inv(M)*K zeros(2)]; % original system
Bo = [0;0;inv(M)*[1;0]];
```

```
Co = [1 0 0 0 ; 0 1 0 0];
Do = [0;0];
y0 = [0 0 0 0]';
yf = [1 1 0 0]';

[V,D]=eig(inv(M)*K)

A   = [   0   1   0   0        % after similarity transform
          0   0   0   0
          0   0   0   1
          0   0  -2   0  ];
B   = [0 -1/sqrt(2) 0 -1/sqrt(2)]';
x0  = [0 0 0 0]';
xf  = [-1.4142 0 0 0]';

syms t lam1 lam2 lam3 lam4 lam5 lam6

e_at    = expm(-A*t);
lam     = [lam1;lam2;lam3;lam4;lam5;lam6];
u       = [-sqrt(2)/12*t^3 sqrt(2)/4*t^2 1/4*sin(sqrt(2)*t) ...
            -1/(2*sqrt(2))*cos(sqrt(2)*t) t 1]*lam;
u_d     = diff(u,t);
aaa=subs(u_d,t,0)
bbb=subs(u_d,t,pi)
aa=[int(e_at*B*u,t,0,pi);subs(u_d,t,0);subs(u_d,t,pi)];
lambda = inv(jacobian(aa,lam))*[subs(e_at,t,pi)*xf;0;0];
lambda = eval(lambda)
lam1=lambda(1,1); lam2=lambda(2,1); lam3=lambda(3,1);
lam4=lambda(4,1); lam5=lambda(5,1); lam6=lambda(6,1);

t=[0:0.001:6];
m=length(t);
U=zeros(size(t));    % Control Input
U_d=zeros(size(t)); % Control Input rate

u1=-sqrt(2)/12*lam1
u2=sqrt(2)/4*lam2
u3=1/4*lam3
u4=-1/(2*sqrt(2))*lam4
u5=lam5
u6=lam6
```

```
ud1=-1/4*2^(1/2)*lam1
ud2=1/2*2^(1/2)*lam2
ud3=1/4*2^(1/2)*lam3
ud4=1/2*lam4
ud5=lam5

% generating control input
for i=1:m
    U(i)    = [ -sqrt(2)/12*t(i)^3 ...
                sqrt(2)/4*t(i)^2 ...
                1/4*sin(sqrt(2)*t(i)) ...
                -1/(2*sqrt(2))*cos(sqrt(2)*t(i)) ...
                t(i) ...
                1                              ]*lambda;
    U_d(i) = -1/4*2^(1/2)*t(i)^2*lam1 + 1/2*2^(1/2)*t(i)*lam2 ...
             +1/4*2^(1/2)*cos(2^(1/2)*t(i))*lam3 ...
             +1/2*sin(2^(1/2)*t(i))*lam4 + lam5;
    if t(i)>pi
        U(i)    = 0;
        U_d(i) = 0;
    end
end

sys=ss(Ao,Bo,Co,Do);
[Y,T,X]=lsim(sys,U,t,y0);

figure(1)
subplot(211)
plot(T,Y(:,1),'-', T,Y(:,2),'-.')
ylabel('Position, m')
legend('y_1','y_2')
subplot(212)
plot(T,U,'-', T,U_d,'-.')
xlabel('Time, sec')
ylabel('Control Input')
legend('u','u_dot')
```

Bibliography

[1] Singh, T. and Vadali, S. R., "Robust Time-Delay Control of Multimode Systems," *International Journal of Control*, Vol. 62, No. 6, 1995, pp. 1319–1339.

[2] Kuo, B. C. and Golnaraghi, F., *Automatic Control System*, Wiley Text Books, 2002.

[3] Singh, T., *Vibration Control of Slewing Structures*, Class Notes, to be published.

[4] Kirk, D. R., *Optimal Control Theory, An Introduction*, Prentice-Hall, Englewood Cliffs, NJ, 1970.

[5] Hindle, T. A. and Singh, T., "Desensitized Minimum Power/Jerk Control Profiles for Rest-to-Rest Maneuvers," *American Control Conference*, Chicago, Il., June 28-30 2000, pp. 3064–3068.