```
% Tim Coon 12 January 2015
% Qualifying Exam Question #1
% Covering Material from MECH 719, Vibration Damping and Control
% Advisor: Dr. Richard Cobb
clear; close all; clc;
scrsz = get(0,'ScreenSize');
scrwidth = scrsz(3);
scrheight = scrsz(4);

%% load the experimental data
load('MECH719_QualDATA.mat');

% split the data into multiple samples. there are 10000 data points
sample_interval = 2^11;         % use a power of two so FFT doesn't pad with↙
zeros
num_samples = floor(length(data)/sample_interval);
for sample = 1:num_samples
    Sstart = sample_interval*(sample-1)+1;
    Send = sample_interval*sample;
    F(:,sample) = data(Sstart:Send,1);
    X(:,:,sample) = data(Sstart:Send,2:3);
    p1(:,sample) = cumtrapz(X(:,1,sample));
end

% test parameters
Fs = 5;             % (1/s) sampling frequency
dt = 1/Fs;          % (s) sample time
N = length(F);      % (-) number of samples
T = N*dt;           % (s) time duration of sample
t = (0:N-1)*dt;     % (s) time vector
% frequency vector
fnyq = 1/(2*dt);                % (Hz) nyquist frequency
df = 1/T;                       % (Hz) frequency interval
f = 0:df:fnyq/2;

%% process the Frequency Response Function and Coherence Function
% calculate the auto-PSD of the input
for sample = 1:num_samples
    % calculate the auto-PSD of the input
    Sai(:,sample) = FFT_PSD(F(:,sample),F(:,sample),T);
end
Sff = mean(Sai,2);

for out = 1:2
    for sample = 1:num_samples
        x = X(:,out,sample);
        % calculate the auto-PSD of the output
        Sao(:,sample) = FFT_PSD(x,x,T);
```

```
        % calculate the cross-PSD in/out
        Scr(:,sample) = FFT_PSD(F(:,sample),x,T);
    end
    % find average PSDs for accuracy
    Sxx(:,out) = mean(Sao,2);
    % calculate the cross-PSDs of in/outs
    Sfx(:,out) = mean(Scr,2);
    % calculate the FRF data
    H(:,out) = Sfx(:,out)./Sff;
    % calculate the Coherence
    C(:,out) = abs(Sfx(:,out)).^2./(Sff.*Sxx(:,out));
end
Phase = rad2deg(angle(H));
H_mag = 20*log10(abs(H));       % dB

%% Plot Simulated and measured FRFs
% system parameter values
m = 1;      % (kg)
c = 0.6;      % (N-s/m)
k = 10;      % (N/m)

% first-order state-space
A = [   0   1   0    0;
     -3*k/m 0  k/m   0;
        0   0   0    1;
      k/2*m 0 -k/m -c/2*m];
B = [0 1/m 0 0]';
C1 = eye(4);
D = zeros(4,1);

% calculate transfer functions
s = tf('s');
TF1 = C1*inv(s*eye(4)-A)*B;
F_p1 = TF1(1);      F_p2 = TF1(3);      % position
F_v1 = TF1(2);      F_v2 = TF1(4);      % velocity
F_a1 = TF1(2)*s;    F_a2 = TF1(4)*s;    % acceleration
w = f*(2*pi);                           % (rad/sec) frequency vector

% simulations
[magp(:,1),phasep(:,1)] = bode(F_p1,w);
[magv(:,1),phasev(:,1)] = bode(F_v1,w);
[maga(:,1),phasea(:,1)] = bode(F_a1,w);
[magp(:,2),phasep(:,2)] = bode(F_p2,w);
[magv(:,2),phasev(:,2)] = bode(F_v2,w);
[maga(:,2),phasea(:,2)] = bode(F_a2,w);
magp = 20*log10(magp); magv = 20*log10(magv); maga = 20*log10(maga); % (dB)

% overlay data Bode plots and calculated bode plots
```

```
L = min([length(f),length(H_mag)])-50;
start = 100;
pos = [0 0 scrwidth/2 scrheight; scrwidth/2 0 scrwidth/2 scrheight];
for fig = 1:2
    figure('Position',pos(fig,:))
    suptitle(titles1(fig))
    subplot(311)
    plot(f(start:L),H_mag(start:L,fig))
    hold on
    plot(f(start:L),magp(start:L,fig))
    plot(f(start:L),magv(start:L,fig),'k--')
    plot(f(start:L),maga(start:L,fig),'g-.')
    ylabel('Magnitude (dB)')
    hold off
    legend('test','pos','vel','acc')
    % phase plots
    subplot(312)
    plot(f(start:L),Phase(start:L,fig))
    hold on
    plot(f(start:L),phasep(start:L,fig))
    plot(f(start:L),phasev(start:L,fig),'k--')
    plot(f(start:L),phasea(start:L,fig),'g-.')
    ylabel('Phase (deg)')
    hold off
    legend('test','pos','vel','acc')
    % coherence plots
    subplot(313)
    plot(f(start:L),C(start:L,fig))
    xlabel('Frequency (Hz)'); ylabel('Coherence');
end

%% Compare measured mode shapes (from FRFs) to theoretical mode shapes
% to find the theoretical mode shapes, use the second-order system
M = [m 0; 0 2*m];
C = [0 0; 0 c];
K = [3*k -k; -k 2*k];
fm = [1; 0];
[V,D] = eig(K,M);
V1_ind = find(max(V(:,1)));
V2_ind = find(max(V(:,2)));
eVector1_calc = V(:,1)/V(V1_ind,1)
eVector2_calc = V(:,2)/V(V2_ind,2)


%% Output data for Ezera
FreqV = f(start:L);
frf = H(start:L,:);
save('MECH791_Qual_EZERA_DATA.mat','FreqV','frf')
```

```
%% Plot mode shapes from measured data

H = H(start:L,:);
f = f(start:L);

Hp1 = H(:,1)./(1i*2*pi*f');     % integrate data in frequency domain
Hp1_mag = abs(Hp1);
Hp2_mag = abs(H(:,2));

figure()
plot(f,Hp1_mag,f,Hp2_mag)
title('FRFs')
legend('Mass #1 Pos','Mass #2 Pos');
ylabel('Absolute Magnitude (dB)'); xlabel('Frequency (Hz)');

% measured data from plots
w_r1 = 0.437;      % (Hz)
Hmag_p1r1 = 0.1708;    % (-) NOT dB
Hmag_p2r1 = 0.3286;
w_r2 = 0.9033;
Hmag_p1r2 = 5.818;
Hmag_p2r2 = 1.283;

% % extract values automatically (not so great)
% r1_ind = find(abs(f-0.4370) < df/2);
% r2_ind = find(abs(f-0.9033) < df/2);
% Hmag_p1r1 = Hp1_mag(r1_ind);
% Hmag_p2r1 = Hp2_mag(r1_ind);
% Hmag_p1r2 = Hp1_mag(r2_ind);
% Hmag_p2r2 = Hp2_mag(r2_ind);

% to determine the sign, use cosd(phaseangle)
% use phase angles from analytical FRF plots
S_p1r1 = sign(cosd(-55.61));
S_p2r1 = sign(cosd(-74.12));
S_p1r2 = sign(cosd(-82.1));
S_p2r2 = sign(cosd(-257.7));

Vm =[S_p1r1*Hmag_p1r1 S_p1r2*Hmag_p1r2;
     S_p2r1*Hmag_p2r1 S_p2r2*Hmag_p2r2];

Vm1_ind = find(max(Vm(:,1)));
Vm2_ind = find(max(Vm(:,2)));
eVector1_meas = Vm(:,1)/Vm(Vm1_ind,1)
eVector2_meas = Vm(:,2)/Vm(Vm2_ind,2)
```