```
%% Preliminary Commands
clc;
close all;
clear all;

currentFolder = strcat(pwd,'\Plots');

set(0,'defaulttextinterpreter','latex')
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

%% Known Quantities
m_cart = 0.3759; % kg
m_disk = 0.1396; % kg
m_tot = m_cart + m_disk; % kg

n1 = 200;
n2 = 2000;

%% Experimantal Data Acquisition
OneDOF1 = importdata('Laboratory_Data\1dof_1.txt');
OneDOF2 = importdata('Laboratory_Data\1dof_2.txt');
OneDOF3 = importdata('Laboratory_Data\1dof_3.txt');
OneDOF4 = importdata('Laboratory_Data\1dof_4.txt');
OneDOF5 = importdata('Laboratory_Data\1dof_5.txt');

% Time is the same for all tests
time1 = OneDOF1(:,1);
% Prepare a matrix to store acceleration
data_size1 = size(OneDOF1,1);
acc_mat = zeros(data_size1,5);
for i=1:data_size1
    acc_mat(i,1) = OneDOF1(i,3);
    acc_mat(i,2) = OneDOF2(i,3);
    acc_mat(i,3) = OneDOF3(i,3);
    acc_mat(i,4) = OneDOF4(i,3);
    acc_mat(i,5) = OneDOF5(i,3);
end

n_test = 5;
% Plot the raw data
figure('Name','Raw Data')
for i=1:n_test
    plot(time1,acc_mat(:,i),'k');
    title(['Test ',num2str(i),' Cart Acceleration']);
    xlabel('time [s]');
    ylabel('Acceleration [$m/s^2$]');
    pbaspect([3 1 1])

    figure_name = strcat('\',num2str(i),'.Raw_Test_',num2str(i),'.png');
    exportgraphics(gcf,strcat(currentFolder,figure_name))
end

%% Logarithmic Decrement Technique
delta = zeros(n_test,1);
```

```
xi = zeros(n_test,1);
T = zeros(n_test,1);
omega_N = zeros(n_test,1);
K = zeros(n_test,1);
C = zeros(n_test,1);
PeakProminence = 0.4;

for i=1:n_test
    figure('Name','Smooth')
    plot(time1,acc_mat(:,i),'c');
    acc_len = length(acc_mat(:,i));
    hold on

    % Smoothed Signal
    span1 = n1/acc_len;
    acc_smooth1 = smooth(acc_mat(:,i),span1,'lowess');
    plot(time1,acc_smooth1,'k');

    % Oversmoothed Signal
    span2 = n2/acc_len;
    acc_smooth2 = smooth(acc_smooth1,span2,'lowess');
    plot(time1,acc_smooth2,'r');
    hold on;

    % Find the peak values and locations on the oversmoothed signal:
    [p_val,p_loc] = findpeaks(acc_smooth2,'MinPeakProminence',↙
PeakProminence,'MinPeakHeight',0.5);

    % Plot the peaks
    peak_time = time1(p_loc);
    scatter(peak_time,acc_smooth2(p_loc),'r','*');
    scatter(peak_time,acc_smooth1(p_loc),'k','*');
    title(['Peaks Test ', num2str(i)]);
    xlim([1 5]);
    xlabel('Time [s]');
    ylabel('Acceleration [$m/s^2$]');
    legend('Experimental','Smoothed','Oversmoothed');
    hold off

    figure_name = strcat('\',num2str(i+5),'.Peaks_Test_',num2str(i),'.png');
    exportgraphics(gcf,strcat(currentFolder,figure_name))

    % Compute the Damping Ratio through the Lograithmic Decrement method
    % delta = ln(x[1]/x[n+1])/n
    % xi = delta/(sqrt(4*(pi^2)+delta^2)

    % Neglect the first peak
    n = 2;
    % Avoid noise in adjacent peak by calculating delta over m cycles
    m = 6;

    % Logarithmic Decrement delta
    delta(i) = log(acc_smooth1(p_loc(n))/acc_smooth1(p_loc(n+1+m)))/m;
    % Damping Ratio xi
    xi(i) = delta(i)/(sqrt(4*pi^2 + (delta(i))^2));
```

```
    % Pseudo-Period T[n+1] - T[n]
    Tn = time1(p_loc(n));
    Tnp1 = time1(p_loc(n+m));
    T(i) = (Tnp1 - Tn)/m;
end

%% Mean Value and Standard Deviation
Mean_xi = mean(xi);
Sigma_xi = std(xi);

Mean_T = mean(T);
Sigma_T = std(T);

% Uncertainty on Damping Ratio is 1 order of magnitude smaller and should
%    be considered, but can be neglected since it does not have a significant
%    impact
% Uncertainty on Period is very small and can be neglected

%% Natural Frequency, Stiffness, Damping Coefficient
for i=1:n_test
    % Natural Frequency
    omega_N(i) = (2*pi)/(T(i)*sqrt(1 - (xi(i))^2));

    % Stiffness
    K(i) = omega_N(i)^2*m_tot;

    % Damping Coefficient
    C(i) = 2*xi(i)*m_tot*omega_N(i);
end

% Mean Value and Standard Deviation
Mean_omega_N = mean(omega_N);
Sigma_omega_N = std(omega_N);

Mean_K = mean(K);
Sigma_K = std(K);

Mean_C = mean(C);
Sigma_C = std(C);

%% 2 DoF System
% Known Quantities
m_cart = 0.3759; % kg
m_disk = 0.1396; % kg
m_beam = 4.7764; % kg
m_shaker = 0.2000; % kg
l_rod = 0.1; % m
w_rod = 0.025; % m
t_rod = 0.0015; % m
C_rod = 0.01;
l_beam = 605; % mm
w_beam = 30; % mm
E = 210; % GPa
rho = 7850; % kg/m^3
```

```
fs = 6400; % Hz

% Choose a linear model for the beams and define the equivalent system
m_rod = rho*l_rod*w_rod*t_rod; % kg
I = (t_rod^3)*w_rod/12; % kg*m^2
K_rod_eq = 2*12*(E*10^9)*I/(l_rod^3);
C_rod_eq = 2*C_rod*sqrt(K_rod_eq*m_beam);

m1 = m_beam;
c1 = C_rod_eq;
k1 = K_rod_eq;

m2 = m_tot;
c2 = Mean_C;
c2_UP = c2 + 3*Sigma_C;
c2_DOWN = c2 - 3*Sigma_C;
C2 = [c2_DOWN,c2,c2_UP];
k2 = Mean_K;
k2_UP = k2 + 3*Sigma_K;
k2_DOWN = k2 - 3*Sigma_K;
K2 = [k2_DOWN,k2,k2_UP];

%% Compute the Analytical Transfer Functions between Cart/Beam Acceleration and Force
% The TF for the acceleration is G(s)*s^2

% Cart Acceleration VS Force
G_Cart_Numerator = zeros(3,5);
G_Cart_Denominator = zeros(3,5);

for i=1:3
    tmp_Num = [0,C2(i),K2(i),0,0];
    tmp_Den = [m1*m2, ...
               m1*C2(i) + m2*(c1 + C2(i)), ...
               m1*K2(i) + c1*C2(i) + m2*(k1 + K2(i)), ...
               K2(i)*c1 + k1*C2(i), ...
               k1*K2(i)];

    G_Cart_Numerator(i,:) = tmp_Num;
    G_Cart_Denominator(i,:) = tmp_Den;
end

G_Cart_DOWN = tf(G_Cart_Numerator(1,:),G_Cart_Denominator(1,:));
G_Cart = tf(G_Cart_Numerator(2,:),G_Cart_Denominator(2,:));
G_Cart_UP = tf(G_Cart_Numerator(3,:),G_Cart_Denominator(3,:));

figure('Name','Analytical Cart Acceleration Transfer Function')
k = bodeplot(G_Cart,'k');
setoptions↙
(k,'FreqUnits','Hz','FreqScale','log','PhaseVisible','off','MagUnits','abs');
hold on
grid on
bodeplot(G_Cart_UP,'r--');
bodeplot(G_Cart_DOWN,'c--');
title('Analytical Cart Acceleration Transfer Function');
xlabel('Frequency ');
```

```
ylabel('Magnitude ');
xlim([5 30]);
hold off
legend('TF','TF + Error','TF - Error');

exportgraphics(gcf,strcat(currentFolder,'\11.Cart_Analytical_TF.png'))

% Beam Acceleration VS Force
G_Beam_Numerator = zeros(3,5);
G_Beam_Denominator = zeros(3,5);

for i=1:3
    tmp_Num = [m2,C2(i),K2(i),0,0];
    tmp_Den = [m1*m2, ...
               m1*C2(i) + m2*(c1 + C2(i)), ...
               m1*K2(i) + c1*C2(i) + m2*(k1 + K2(i)), ...
               K2(i)*c1 + k1*C2(i), ...
               k1*K2(i)];

    G_Beam_Numerator(i,:) = tmp_Num;
    G_Beam_Denominator(i,:) = tmp_Den;
end

G_Beam_DOWN = tf(G_Beam_Numerator(1,:),G_Beam_Denominator(1,:));
G_Beam = tf(G_Beam_Numerator(2,:),G_Beam_Denominator(2,:));
G_Beam_UP = tf(G_Beam_Numerator(3,:),G_Beam_Denominator(3,:));

figure('Name','Analytical Beam Acceleration Transfer Function')
k = bodeplot(G_Beam,'k');
setoptions↙
(k,'FreqUnits','Hz','FreqScale','log','PhaseVisible','off','MagUnits','abs');
hold on
grid on
bodeplot(G_Beam_UP,'r--');
bodeplot(G_Beam_DOWN,'c--');
title('Analytical Beam Acceleration Transfer Function');
xlabel('Frequency ');
ylabel('Magnitude ');
xlim([5 30]);
hold off
legend('TF','TF + Error','TF - Error');

exportgraphics(gcf,strcat(currentFolder,'\12.Beam_Analytical_TF.png'))

%% Experimental Data Acquisition and TF Estimation
TwoDOF1 = importdata('Laboratory_Data\2dof_1.txt');
TwoDOF2 = importdata('Laboratory_Data\2dof_2.txt');
TwoDOF3 = importdata('Laboratory_Data\2dof_3.txt');
TwoDOF4 = importdata('Laboratory_Data\2dof_4.txt');
TwoDOF5 = importdata('Laboratory_Data\2dof_5.txt');

% Time is the same for all tests
time2 = TwoDOF1(:,1);
% Prepare a matrix to store acceleration
data_size2 = size(TwoDOF1,1);
```

```
force_mat = zeros(data_size2,5);
acc_cart_mat = zeros(data_size2,5);
acc_beam_mat = zeros(data_size2,5);
for i=1:data_size2
    force_mat(i,1) = TwoDOF1(i,2);
    force_mat(i,2) = TwoDOF2(i,2);
    force_mat(i,3) = TwoDOF3(i,2);
    force_mat(i,4) = TwoDOF4(i,2);
    force_mat(i,5) = TwoDOF5(i,2);

    acc_cart_mat(i,1) = TwoDOF1(i,3);
    acc_cart_mat(i,2) = TwoDOF2(i,3);
    acc_cart_mat(i,3) = TwoDOF3(i,3);
    acc_cart_mat(i,4) = TwoDOF4(i,3);
    acc_cart_mat(i,5) = TwoDOF5(i,3);

    acc_beam_mat(i,1) = TwoDOF1(i,4);
    acc_beam_mat(i,2) = TwoDOF2(i,4);
    acc_beam_mat(i,3) = TwoDOF3(i,4);
    acc_beam_mat(i,4) = TwoDOF4(i,4);
    acc_beam_mat(i,5) = TwoDOF5(i,4);
end

% Prepare matrices to store TFs data
TF_dim = 32769;
TF_Cart_Mag = zeros(TF_dim,n_test);
TF_Cart = zeros(TF_dim,n_test);
TF_Beam_Mag = zeros(TF_dim,n_test);
TF_Beam = zeros(TF_dim,n_test);

for i=1:n_test
    [tmp_TF_Cart,Freq_Cart] = tfestimate(force_mat(:,i),acc_cart_mat(:,i),[],[],[],↙
fs);
    TF_Cart(:,i) = tmp_TF_Cart;
    TF_Cart_Mag(:,i) = abs(tmp_TF_Cart);

    [tmp_TF_Beam,Freq_Beam] = tfestimate(force_mat(:,i),acc_beam_mat(:,i),[],[],[],↙
fs);
    TF_Beam(:,i) = tmp_TF_Beam;
    TF_Beam_Mag(:,i) = abs(tmp_TF_Beam);

    % Plot the raw data
    figure('Name','Experimental Cart/Beam Acceleration Transfer Functions')
    nexttile
    plot(Freq_Cart,TF_Cart_Mag(:,i),'k');
    grid on
    title(strcat('Experimental Cart TF - Test', num2str(i)));
    xlabel('Frequency ');
    ylabel('Magnitude ');
    xlim([5 30]);
    xscale log;

    nexttile
    plot(Freq_Beam,TF_Beam_Mag(:,i),'k');
    grid on
```

```
    title(strcat('Experimental Beam TF - Test', num2str(i)));
    xlabel('Frequency ');
    ylabel('Magnitude ');
    xlim([5 30]);
    xscale log;

    figure_name = strcat('\',num2str(i+12),'.Cart-Beam_Experimental_TF_Test_',num2str↙
(i),'.png');
    exportgraphics(gcf,strcat(currentFolder,figure_name))
end

%% Compute the Mean Experimental Transfer Function and Its Standard Deviation
TF_Cart_Mag_Transposed = TF_Cart_Mag';
Mean_TF_Cart = mean(TF_Cart_Mag_Transposed)';
Sigma_TF_Cart = std(TF_Cart_Mag_Transposed)';

TF_Beam_Mag_Transposed = TF_Beam_Mag';
Mean_TF_Beam = mean(TF_Beam_Mag_Transposed)';
Sigma_TF_Beam = std(TF_Beam_Mag_Transposed)';

figure('Name','Mean Cart/Beam Acceleration Transfer Function')
nexttile
plot(Freq_Cart,Mean_TF_Cart,'k');
title('Mean Experimenta Cart TF');
xlabel('Frequency ');
ylabel('Magnitude ');
xlim([5 30]);
xscale log;
%yscale log;
hold on
grid on
plot(Freq_Cart,Mean_TF_Cart+3*Sigma_TF_Cart,'r--');
plot(Freq_Cart,Mean_TF_Cart-3*Sigma_TF_Cart,'c--');
hold off
legend('Mean TF','Mean TF + Error','Mean TF - Error');

nexttile
plot(Freq_Beam,Mean_TF_Beam,'k');
title('Mean Experimenta Beam TF');
xlabel('Frequency ');
ylabel('Magnitude ');
xlim([5 30]);
xscale log;
%yscale log;
hold on
grid on
plot(Freq_Beam,Mean_TF_Beam+3*Sigma_TF_Beam,'r--');
plot(Freq_Beam,Mean_TF_Beam-3*Sigma_TF_Beam,'c--');
hold off
legend('Mean TF','Mean TF + Error','Mean TF - Error');

exportgraphics(gcf,strcat(currentFolder,'\18.Mean_Cart-Beam_Experimental_TF.png'))

%% Compare Analytical and Experimental TFs
figure('Name','Analytical VS Experimental Cart Acceleration Transfer Function')
```

```
p1 = plot(Freq_Cart,Mean_TF_Cart,'k');
xscale log;
hold on
p2 = bodeplot(G_Cart,'b');
setoptions(p2,'FreqUnits','Hz','PhaseVisible','off', 'MagUnits',↙
'abs','FreqScale','log');
title('Analytical VS Experimental Cart TF');
xlim([5 30]);
h = [p1;findobj(gcf,'type','line')];
legend(h,'Experimental TF','Analytical TF');
grid on
hold off

exportgraphics(gcf,strcat(currentFolder,'\19.Analytical_VS_Experimental_Cart_TF.png'))

figure('Name','Analytical VS Experimental Beam Acceleration Transfer Function')
p1 = plot(Freq_Beam,Mean_TF_Beam,'k');
xscale log;
hold on
p2 = bodeplot(G_Beam,'b');
setoptions(p2,'FreqUnits','Hz','PhaseVisible','off', 'MagUnits',↙
'abs','FreqScale','log');
title('Analytical VS Experimental Beam TF');
xlim([5 30]);
h = [p1;findobj(gcf,'type','line')];
legend(h,'Experimental TF','Analytical TF');
grid on
hold off

exportgraphics(gcf,strcat(currentFolder,'\20.Analytical_VS_Experimental_Beam_TF.png'))

%% Fitting
% Assume the Spring Stiffness, the Rod Stiffness, the Damping Factor of
% the linear guide and the Damping Factor of the rods are unknowns.

% Select frequencies from 5 to 30 Hz
range = find(Freq_Beam>5 & Freq_Beam<30);
Freq_Range = Freq_Beam(range);

TF_Beam_Range = Mean_TF_Beam(range);
TF_Cart_Range = Mean_TF_Cart(range);

s = sqrt(-1)*Freq_Range*2*pi;

% Redefine the TFs with the new unknowns
G_Cart_Fit = @(k_beam,c_beam,k_cart,c_cart) ...
             (c_cart.*s.^3 + k_cart.*s.^2)./ ...
             ((m1*m2).*s.^4 + ...
              (m1*c_cart + m2*(c_beam + c_cart)).*s.^3 + ...
              (m1*k_cart + m2*(k_beam + k_cart) + c_beam*c_cart).*s.^2 + ...
              (k_cart*c_beam + k_beam*c_cart).*s + ...
              k_beam*k_cart);
G_Beam_Fit = @(k_beam,c_beam,k_cart,c_cart) ...
             (m2.*s.^4 + c_cart.*s.^3 + k_cart.*s.^2)./ ...
             ((m1*m2).*s.^4 + ...
```

```
                (m1*c_cart + m2*(c_beam + c_cart)).*s.^3 + ...
                (m1*k_cart + m2*(k_beam + k_cart) + c_beam*c_cart).*s.^2 + ...
                (k_cart*c_beam + k_beam*c_cart).*s + ...
                k_beam*k_cart);

% Define the Error
err_Cart = @(x) rms(TF_Cart_Range - abs(G_Cart_Fit(x(1),x(2),x(3),x(4))));
err_Beam = @(x) rms(TF_Beam_Range - abs(G_Beam_Fit(x(1),x(2),x(3),x(4))));

% Define the Initial Guess
x0 = [k1,c1,k2,c2];

% From the initial guess minimize the error
options = optimset('MaxFunEvals', 10000);
x_opt_Cart = fminsearch(err_Cart,x0,options);
x_opt_Beam = fminsearch(err_Beam,x0,options);

figure('Name','Fitted Transfer Function')
nexttile
plot(Freq_Range,abs(G_Cart_Fit(x_opt_Cart(1),x_opt_Cart(2),x_opt_Cart(3),x_opt_Cart ↙
(4))),'k');
hold on
grid on
plot(Freq_Range,abs(G_Cart_Fit(x0(1),x0(2),x0(3),x0(4))),'b:');
plot(Freq_Range,TF_Cart_Range,'r--');
title('Cart TF Fitting')
xlabel('Frequency ');
ylabel('Magnitude ');
xlim([5 30]);
xscale log;
legend('Fitted TF','Initial Guess TF','Experimental TF')
hold off

nexttile
plot(Freq_Range,abs(G_Beam_Fit(x_opt_Beam(1),x_opt_Beam(2),x_opt_Beam(3),x_opt_Beam ↙
(4))),'k');
hold on
grid on
plot(Freq_Range,abs(G_Beam_Fit(x0(1),x0(2),x0(3),x0(4))),'b:');
plot(Freq_Range,TF_Beam_Range,'r--');
title('Beam TF Fitting')
xlabel('Frequency ');
ylabel('Magnitude ');
xlim([5 30]);
xscale log;
legend('Fitted TF','Initial Guess TF','Experimental TF')
hold off

exportgraphics(gcf,strcat(currentFolder,'\21.Fitting.png'))
```