WRIGHT STATE UNIVERSITY
DEPARTMENT OF MECHANICAL AND MATERIALS ENGINEERING

ME7690 - Vibration Testing and Machine Health Monitoring

System Identification using Frequency Domain Modal Analysis

Koorosh Gobal

April 29, 2014

PROBLEM DESCRIPTION

Experimental modal analysis has become a commonly-used technique for studying the dynamical behavior of mechanical and civil structures, such as cars, aircrafts, bridges and industrial machinery. During a modal test, both the applied forces and vibration responses of the structure are measured when excited in one or more locations. Based on this data, a modal model of the structure, that essentially contains the same information as the original vibration data, is derived by means of system identification. The identification techniques are categorized based on number of degrees of freedom of the system and number of inputs/outputs.

METHODOLOGY

SINGLE DEGREE OF FREEDOM CURVE FITTING WITH RESIDUAL

"Single Degree of Freedom Curve Fitting with Residual" is one of the methods that is used when several natural frequencies occur in the FRF data. Each of these frequencies contribute over the entire frequency range. Therefore, when fitting a curve to a single peak, the effect of other natural frequencies should be accounted in the procedure. Therefore, the residuals are introduced to compensate the effect of other natural frequencies when fitting only one peak. This is shown in Equation (1). It should be noted that R_I and R_K represent residual contributions from the lower and high frequency modes and does not have any physical meaning.

$$H(j\omega) = R_I + \frac{1}{-\omega^2 m + j\omega c + k} + R_K \tag{1}$$

With some manipulation, above can be written as a ration of two polynomials as

$$H(j\omega) = \frac{\alpha_4 (j\omega)^4 + \alpha_3 (j\omega)^3 + \alpha_2 (j\omega)^2 + \alpha_1 (j\omega) + \alpha_0}{\beta_4 (j\omega)^4 + \beta_3 (j\omega)^3 + \beta_2 (j\omega)^2} \tag{2}$$

This can be written in for all the frequency range around one of the peaks in the matrix form as

$$\begin{bmatrix} (j\omega_1)^2 H(j\omega_1) & (j\omega_1)^3 H(j\omega_1) & -(j\omega_1)^0 & \dots & -(j\omega_1)^4 \\ (j\omega_2)^2 H(j\omega_2) & (j\omega_2)^3 H(j\omega_2) & -(j\omega_2)^0 & \dots & -(j\omega_2)^4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (j\omega_{n_s})^2 H(j\omega_{n_s}) & (j\omega_{n_s})^3 H(j\omega_{n_s}) & -(j\omega_{n_s})^0 & \dots & -(j\omega_{n_s})^4 \end{bmatrix} \begin{bmatrix} \beta_2 \\ \beta_1 \\ \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} j\omega_1)^4 H(j\omega_1) \\ j\omega_2)^4 H(j\omega_2) \\ \vdots \\ j\omega_{n_s})^4 H(j\omega_{n_s}) \end{bmatrix} \tag{3}$$

After solving Equation (3), the two nonzero roots of the polynomial $\lambda^4 + \beta_3\lambda^3 + \beta_2\lambda^2 + 0 + 0 = 0$ yield the poles of the system. The natural frequency and damping coefficient can be calculated using the following equations.

$$\omega_n = ||\lambda|| \tag{4a}$$

$$\zeta = -\frac{\text{real}(\lambda)}{||\lambda||} \tag{4b}$$

Furthermore, this can be followed by a subsequent curve fit to obtain the residue as follows.

$$\begin{bmatrix} H(j\omega_1) \\ H(j\omega_2) \\ \vdots \\ H(j\omega_{n_s}) \end{bmatrix} = \begin{bmatrix} -\frac{1}{\omega_1^2} & 1 & \frac{1}{j\omega_1 - \lambda_1} & \frac{1}{j\omega_1 - \bar{\lambda}_1} \\ -\frac{1}{\omega_2^2} & 1 & \frac{1}{j\omega_2 - \lambda_2} & \frac{1}{j\omega_2 - \bar{\lambda}_2} \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{1}{\omega_{ns}^2} & 1 & \frac{1}{j\omega_{ns} - \lambda_{ns}} & \frac{1}{j\omega_{ns} - \bar{\lambda}_{ns}} \end{bmatrix} \begin{bmatrix} \frac{1}{m_r} \\ \frac{1}{k_r} \\ b \\ \bar{b} \end{bmatrix} \tag{5}$$

The value for the residues can be used to qualify the fitted curve.

MDOF Curve Fitting - Single Input Collocated with Sensor

It is possible to curve fit a large number of FRFs simultaneously to approximate different properties of the system. The "MDOF Curve Fitting" is used when the system is defined to have a single input at a location where the is also a sensor. This method will work provided that the number of modes appearing in the FRFs is equal to the number of sensors.

For the range of n_s frequencies in the domain, the second order equation of motion can be written in the matrix form as

$$\begin{bmatrix} -\omega_1^2 \mathbf{H}(j\omega_1)^T & j\omega_1 \mathbf{H}(j\omega_1)^T & \mathbf{H}(j\omega_1)^T \\ -\omega_2^2 \mathbf{H}(j\omega_2)^T & j\omega_2 \mathbf{H}(j\omega_2)^T & \mathbf{H}(j\omega_2)^T \\ \vdots & \vdots & \vdots \\ -\omega_{n_s}^2 \mathbf{H}(j\omega_{n_s})^T & j\omega_{n_s} \mathbf{H}(j\omega_{n_s})^T & \mathbf{H}(j\omega_{n_s})^T \end{bmatrix} \begin{bmatrix} \mathbf{M} \\ \mathbf{C} \\ \mathbf{K} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{B}}^T \\ \tilde{\mathbf{B}}^T \\ \vdots \\ \tilde{\mathbf{B}}^T \end{bmatrix} \tag{6}$$

it should be noted that the $\mathbf{H}(j\omega)$ is the full FRF matrix as shown in the following equation. Where "o", represents the inputs and "p" represents the output.

$$\mathbf{H}(\omega) = \begin{bmatrix} H_{1,1}(\omega) & H_{1,2}(\omega) & \cdots & H_{1,o}(\omega) \\ H_{2,1}(\omega) & H_{2,2}(\omega) & & \vdots \\ \vdots & & \ddots & \\ H_{p,1}(\omega) & \cdots & & H_{p,o}(\omega) \end{bmatrix} \tag{7}$$

Experimental Setup

In this lab, we derive the system properties of the cruise missile wing that is available in the vibrations lab. Three accelerometers are placed on the test article (wing) and the model is excited using the hammer. The excitation locations are collocated with sensors so we can use MDOF method afterwards. The test setup is shown in Figure 1 where accelerometers are shown using white arrow. In this lab, we have used "Single Degree of Freedom Curve Fitting with Residual" and "MDOF Curve fitting" to identify the system properties.

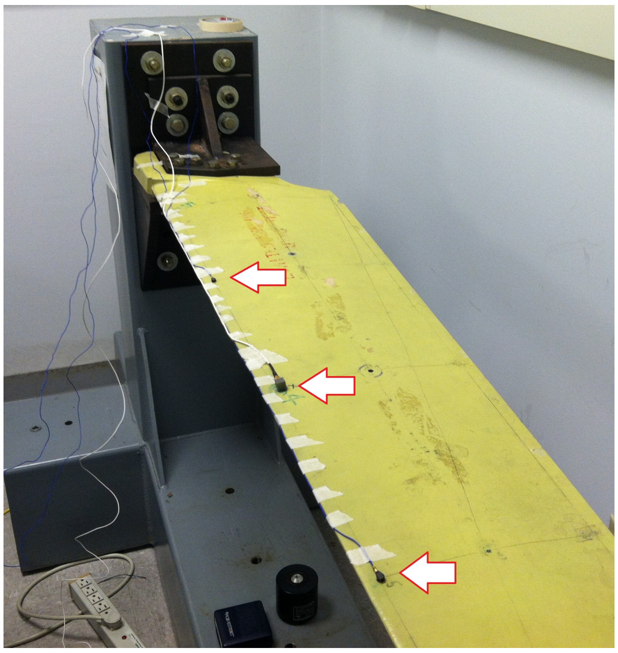


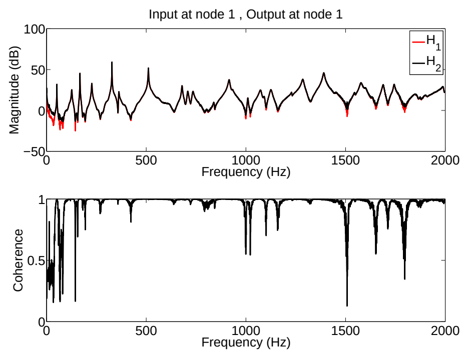
Figure 1: The experimental setup.

Results and Discussion

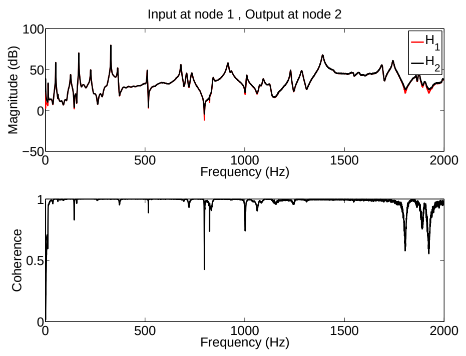
Data is collected by exciting the test article at the collocated locations with the accelerometers. For each excitation, 10 sets of realizations were gathered for post processing. This raw time domain data is then transferred to the frequency domain using Matlab's `fft` function. Based on the frequency domain data, the auto and spectral density functions are calculated for each realization. The auto and spectral density functions are averaged in the frequency domain and used to calculate two different frequency response functions as described in Equation (8). The coherence is also calculate to quantify the quality of the FRF estimates. As shown in Figures 5, 6 and 7, the coherence is around one in most of the frequency range. This represents a good quality FRF.

$$H_1(j\omega) = \frac{G_{fx}(j\omega)}{G_{ff}(j\omega)} \tag{8a}$$

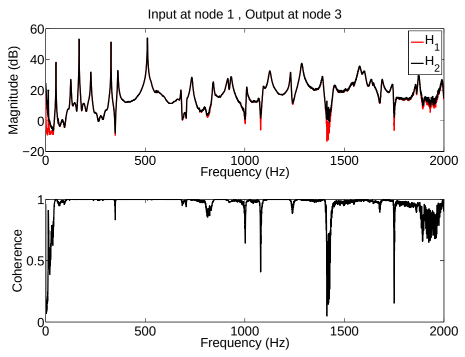
$$H_2(j\omega) = \frac{G_{xx}(j\omega)}{G_{xf}(j\omega)} \tag{8b}$$



(a)

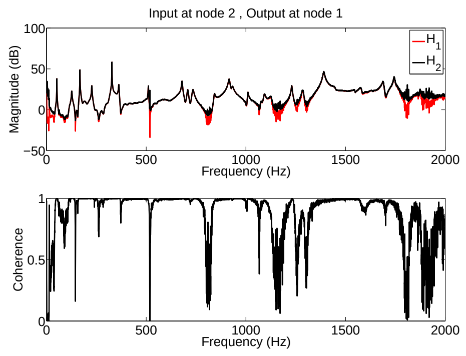


(b)

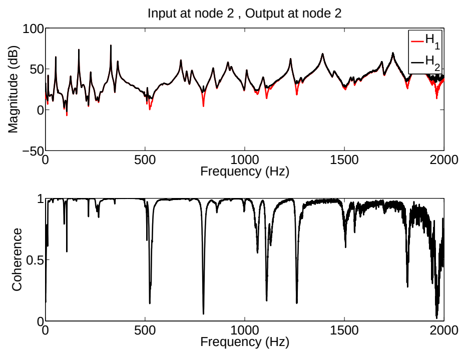


(c)

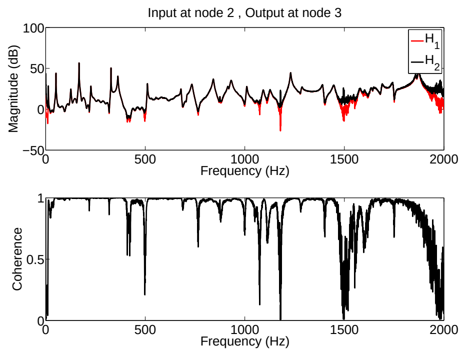
Figure 2: Frequency response function for input at node 1.



(a)

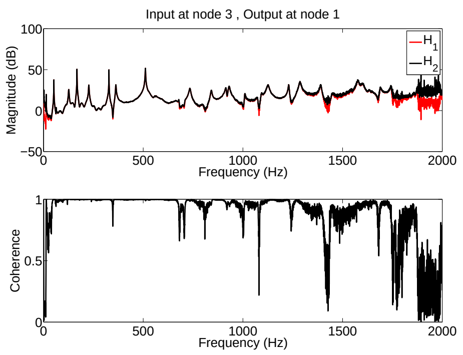


(b)

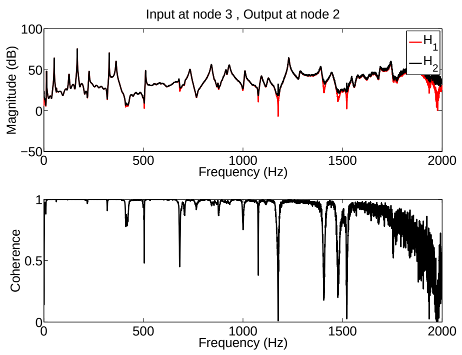


(c)

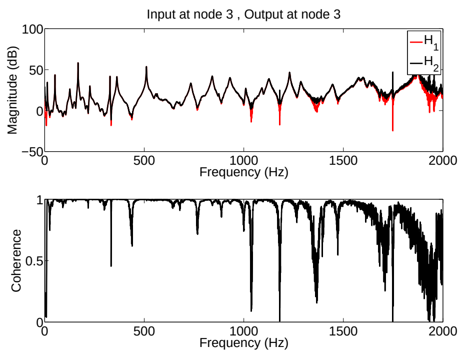
Figure 3: Frequency response function for input at node 2.



(a)



(b)



(c)

Figure 4: Frequency response function for input at node 3.

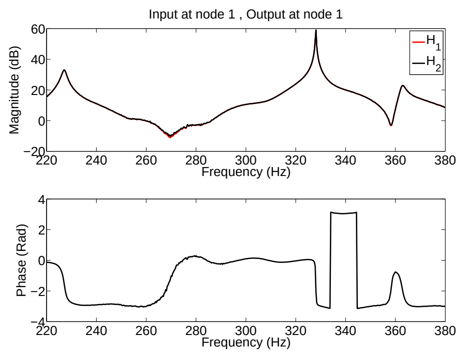
For the MDOF identification technique, we chose the frequency range of 220 Hz to 380 Hz. In this range, the coherence is near one for all the FRFs and also we can see three modes in the FRF data. This is required for the MDOF identification technique since we have three FRFs. The magnitude and phase of the FRFs for the specified frequency range is shown in Figure **??**, **??** and **??**. To use Equation (6), the negative frequency range is also added to the frequency data. The FRF value in the negative frequency range is the complex-conjugate of the the FRF data. The system properties are calculated as follows.

$$M = \begin{bmatrix} 0.0064506 & 0.014113 & 0.02188 \\ 0.00043015 & 0.00084479 & 0.0013725 \\ 0.005462 & 0.013586 & 0.01835 \end{bmatrix}$$

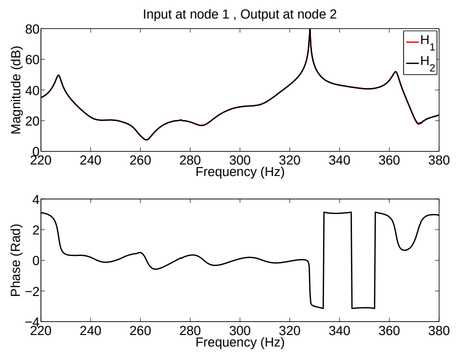
$$C = \begin{bmatrix} 0.0026488 & 0.01265 & 0.0079285 \\ 0.032937 & 0.15733 & 0.098673 \\ 0.00079274 & 0.0037837 & 0.0023673 \end{bmatrix}$$

$$K = \begin{bmatrix} 1.49E-09 & 1.41E-08 & 2.20E-08 \\ 1.94E-08 & 1.14E-07 & 1.13E-07 \\ 2.00E-09 & 1.49E-08 & 2.03E-08 \end{bmatrix}$$

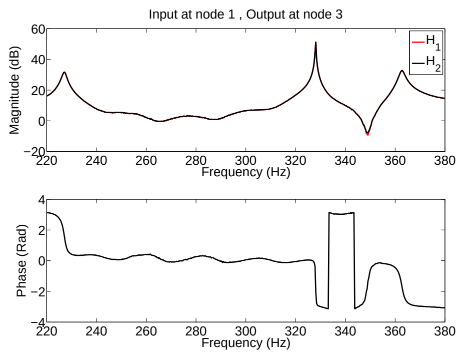
The matrices should have larger values. The low values of system properties are most probably due to having more than three modes in the sampled domain.



(a)

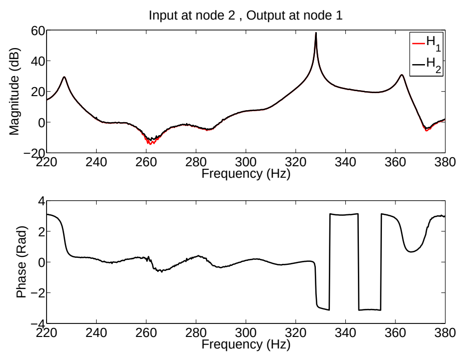


(b)

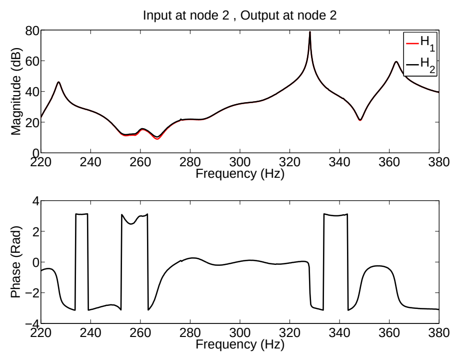


(c)

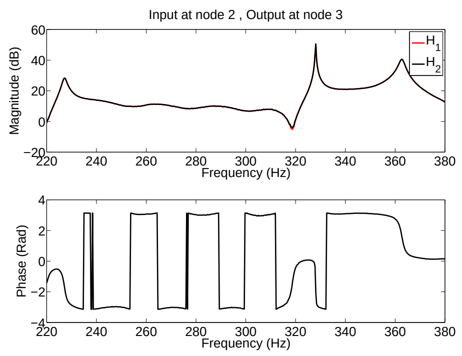
Figure 5: Frequency response function for input at node 1.



(a)

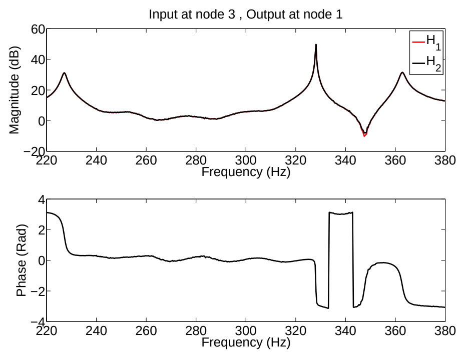


(b)

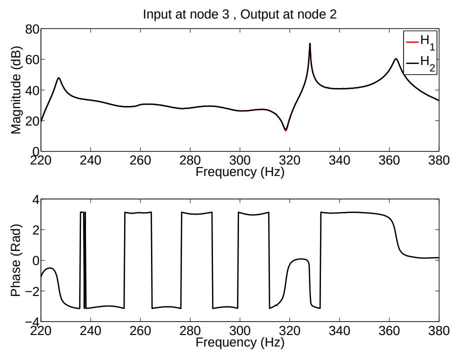


(c)

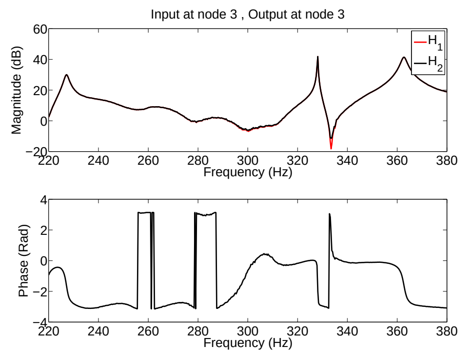
Figure 6: Frequency response function for input at node 2.



(a)



(b)



(c)

Figure 7: Frequency response function for input at node 3.

We have also used “Single Degree of Freedom Curve Fitting with Residual” to fit a curve to one of the peaks to calculate the natural frequency and the damping coefficient related to that peak. We chose H_{31} as the FRF with the frequency range of 485 Hz to 540 Hz. This frequency isolates a peak as shown in Figure 8.

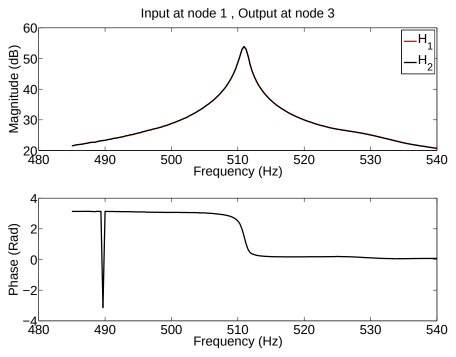


Figure 8: H_{31} for SDOF curve fitting with residual method.

In this method we have also added the negative frequency range to the frequency domain data. The FRF value for the negative frequency range is equal to the complex conjugate value of the FRF in the positive range. Using the experimental data and Equation (3) the natural frequency and damping coefficient is calculate as 510.81 Hz and 0.00126. The natural frequency agrees with what can be seen from the plot. The residue matrix is also shown in the following equation.

Large values of this matrix indicate that the curve fit is not accurate.

$$\begin{bmatrix} \frac{1}{m_r} \\ \frac{1}{k_r} \\ b \\ \bar{b} \end{bmatrix} = \begin{bmatrix} 0.21711 - 171.86i \\ 0.85579 + 8.1014e - 05i \\ -15.976 + 303.17i \\ -15.992 + 303.17i \end{bmatrix} \tag{9}$$

APPENDIX

EXTRACT DATA FROM BOBCAT OUTPUT

```
for inputLocation = 1:3
    Gfx_n1 = 0;
    Gxf_n1 = 0;
    Gff_n1 = 0;
    Gxx_n1 = 0;

    Gfx_n2 = 0;
    Gxf_n2 = 0;
    Gff_n2 = 0;
    Gxx_n2 = 0;

    Gfx_n3 = 0;
    Gxf_n3 = 0;
    Gff_n3 = 0;
    Gxx_n3 = 0;
    fileName = 'C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Data\';
    fileName = [fileName num2str(inputLocation)];
    for ii=1:10
        readName = [fileName '\' num2str(ii) '.mat'];
        load(readName)
        dt = Time_domain(2) - Time_domain(1);
        T = Time_domain(end);
        fs = 1 / dt;
        fNyq = 0.25 * fs;
        df = 1 / T;

        omega = Freq_domain;
        F = fft(Time_chan_1) * dt; F = F(1:length(omega));
        X1 = fft(Time_chan_2) * dt; X1 = X1(1:length(omega));
        X2 = fft(Time_chan_3) * dt; X2 = X2(1:length(omega));
        X3 = fft(Time_chan_4) * dt; X3 = X3(1:length(omega));

        % 'n' represents the input location. For 'far' it is (3)
        Gfx_n1 = Gfx_n1 + conj(F) .* X1 / T;
        Gxf_n1 = Gxf_n1 + conj(X1) .* F / T;
        Gff_n1 = Gff_n1 + real(conj(F) .* F) / T;
        Gxx_n1 = Gxx_n1 + real(conj(X1) .* X1) / T;

        Gfx_n2 = Gfx_n2 + conj(F) .* X2 / T;
        Gxf_n2 = Gxf_n2 + conj(X2) .* F / T;
        Gff_n2 = Gff_n2 + real(conj(F) .* F) / T;
        Gxx_n2 = Gxx_n2 + real(conj(X2) .* X2) / T;

        Gfx_n3 = Gfx_n3 + conj(F) .* X3 / T;
        Gxf_n3 = Gxf_n3 + conj(X3) .* F / T;
```

```
          Gff_n3 = Gff_n3 + real(conj(F) .* F) / T;
          Gxx_n3 = Gxx_n3 + real(conj(X3) .* X3) / T;
      end
      Gfx_n1 = Gfx_n1 / 10;
      Gxf_n1 = Gxf_n1 / 10;
      Gff_n1 = Gff_n1 / 10;
      Gxx_n1 = Gxx_n1 / 10;

      Gfx_n2 = Gfx_n2 / 10;
      Gxf_n2 = Gxf_n2 / 10;
      Gff_n2 = Gff_n2 / 10;
      Gxx_n2 = Gxx_n2 / 10;

      Gfx_n3 = Gfx_n3 / 10;
      Gxf_n3 = Gxf_n3 / 10;
      Gff_n3 = Gff_n3 / 10;
      Gxx_n3 = Gxx_n3 / 10;

      H1_n1 = Gfx_n1 ./ Gff_n1;
      H2_n1 = Gxx_n1 ./ Gxf_n1;
      gamma_n1 = real(H1_n1 ./ H2_n1);

      H1_n2 = Gfx_n2 ./ Gff_n2;
      H2_n2 = Gxx_n2 ./ Gxf_n2;
      gamma_n2 = real(H1_n2 ./ H2_n2);

      H1_n3 = Gfx_n3 ./ Gff_n3;
      H2_n3 = Gxx_n3 ./ Gxf_n3;
      gamma_n3 = real(H1_n3 ./ H2_n3);

      nameH1_n1 = genvarname(['H1_1' num2str(inputLocation)]);
      assignin('base',nameH1_n1,H1_n1);
      namegamma_n1 = genvarname(['gamma_1' num2str(inputLocation)]);
      assignin('base',namegamma_n1,gamma_n1);
      nameH1_n2 = genvarname(['H1_2' num2str(inputLocation)]);
      assignin('base',nameH1_n2,H1_n2);
      namegamma_n2 = genvarname(['gamma_2' num2str(inputLocation)]);
      assignin('base',namegamma_n2,gamma_n2);
      nameH1_n3 = genvarname(['H1_3' num2str(inputLocation)]);
      assignin('base',nameH1_n3,H1_n3);
      namegamma_n3 = genvarname(['gamma_3' num2str(inputLocation)]);
      assignin('base',namegamma_n3,gamma_n3);

      clearvars -except H1_12 gamma_12 H1_22 gamma_22 H1_32 gamma_32 ...
                         H1_11 gamma_11 H1_21 gamma_21 H1_31 gamma_31 ...
                         H1_13 gamma_13 H1_23 gamma_23 H1_33 gamma_33 ...
                         inputLocation fontSize lineWidth omega
  end
  % close all;
  H(1,1,:) = H1_11;
```

```
H(1,2,:) = H1_12;
H(1,3,:) = H1_13;
H(2,1,:) = H1_21;
H(2,2,:) = H1_22;
H(2,3,:) = H1_23;
H(3,1,:) = H1_31;
H(3,2,:) = H1_32;
H(3,3,:) = H1_33;

save 'H1_11.mat' H1_11
save 'H1_12.mat' H1_12
save 'H1_13.mat' H1_13
save 'H1_21.mat' H1_21
save 'H1_22.mat' H1_22
save 'H1_23.mat' H1_23
save 'H1_31.mat' H1_31
save 'H1_32.mat' H1_32
save 'H1_33.mat' H1_33
save 'H.mat' H
save 'omega.mat' omega
```

MDOF Curve Fitting

```
clc;
format short g;
close all;
fclose all;
clear all;
% ------------------------------------------------------------------------ %
lineWidth = 3;
fontSize = 20;
% ------------------------------------------------------------------------ %
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H1_13.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H1_21.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H1_22.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H1_23.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H1_31.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H1_32.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H1_33.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\omega.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
    \Lab4\Matlab\H data\H.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_11.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_12.mat')

Hbig = [];
BtildeBig = [];
rangeMin = 220;
rangeMax = 380;
[r cMin] = min(abs(omega - rangeMin));
[r cMax] = min(abs(omega - rangeMax));

omega = omega * 2 * pi;
omegaR = omega(cMin:cMax);
omegaR = [fliplr(-omegaR')';omegaR];
HH = H(:,:,cMin:cMax);
H_ = zeros(3,3,length(omegaR));
for ii=1:length(omegaR)
    if omegaR(ii) < 0
        H_(:,:,ii) = conj(HH(:,:,length(HH) - ii + 1));
    else
        H_(:,:,ii) = HH(:,:,ii - length(HH));
    end
end

for ii=1:length(omegaR)
        Hbig = [Hbig;...
        -omegaR(ii)^2*H_(:,:,ii).' ...
        i*omegaR(ii)*H_(:,:,ii).' ...
        H_(:,:,ii).'];
    BtildeBig = [BtildeBig;eye(3,3) * -omegaR(ii)^2];
end

MCK = pinv(Hbig) * BtildeBig;

M = abs(MCK(1:3,1:3));
C = abs(MCK(4:6,1:3));
K = abs(MCK(7:9,1:3));
```

Single Degree of Freedom Curve Fitting with Residual

```
clc;
clear all;
format short g;
close all;
% ----------------------------------------------------------------------- %
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_13.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_21.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_22.mat')
```

```
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_23.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_31.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_32.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_33.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\omega.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_11.mat')
load('C:\Users\ecsroot\Desktop\Koorosh\Courses\ME7690_Vibration_testing
        \Lab4\Matlab\H data\H1_12.mat')

rangeMin = 485;
rangeMax = 540;
[r cMin] = min(abs(omega - rangeMin));
[r cMax] = min(abs(omega - rangeMax));
omegaR = omega(cMin:cMax);
omegaR = [fliplr(-omegaR')';omegaR];
H_31R = H1_31(cMin:cMax);
H_31R = [fliplr(conj(H_31R'))';H_31R];

P = [(i * omegaR).^2 .* H_31R, ...
    (i * omegaR).^3 .* H_31R, ...
    -(i * omegaR).^0, ...
    -(i * omegaR).^1, ...
    -(i * omegaR).^2, ...
    -(i * omegaR).^3, ...
    -(i * omegaR).^4];
BETA_ALPHA = pinv(P) * (-(i * omegaR).^4 .* H_31R);
BETA = fliplr([0 0 (BETA_ALPHA(1:2,1)).' 1]);
LAMBDA = roots(BETA);

OMEGA_n = abs(LAMBDA)
ZETA = - real(LAMBDA) ./ abs(LAMBDA)

pinv([-1./omegaR, ...
    ones(length(omegaR),1), ...
    1 ./ (i * omegaR - LAMBDA(3)), ...
    1 ./ (i * omegaR - LAMBDA(4))]) * H_31R;
```