

CHAPTER 10

MODAL ANALYSIS: STATE SPACE FORM

10.1 Introduction

In Chapters 5, 6 and 7 we developed the state space (first order differential equation) form of the equations of motion and used them to solve for the eigenvalues and eigenvectors (with real **or** complex modes) and frequency and transient responses. The state space methodology presented so far was independent of the amount of damping in the system, hence the possibility of complex modes.

In Chapters 8 and 9 we developed the modal analysis method using the second order differential equation form. If the amount of damping in the system is low, we can make the approximation that normal modes exist and solve for the undamped (real) modes of the system. Proportional damping can then be added to the equations of motion in principal coordinates while keeping the equations uncoupled.

In the next three chapters we will combine the state space techniques in Chapters 5, 6 and 7 with the modal analysis techniques in Chapters 8 and 9. In real world situations, finite element models are used to describe dynamic systems. The finite element program is used to solve for eigenvalues and eigenvectors, which are then used to create a state space model in MATLAB. However, one may have the need to solve for eigenvalues and eigenvectors in state space form for a model that is not created using finite elements. For this reason, the chapter will start out with a closed form solution to the tdof eigenvalue problem in state space form. The eigenvalues and eigenvectors which result from the state space eigenvalue problem will contain the same information as in the second order eigenvalue problem, but will be in a different form. The differences will be highlighted and discussed.

We then will use the eigenvalues to form the uncoupled homogeneous equations of motion in the state space principal coordinate system by inspection. Forcing function and initial conditions will then be converted to principal coordinates using the normalized modal matrix, creating the final state equations of motion in the principal coordinate system. As in the second order form, proportional damping can be added to the modal formulation and the solution in principal coordinates back-transformed to physical coordinates for the final result. We will use a method of formulating the input matrix **B** such that the transformation of forces to principal coordinates and conversion to state space form can happen in one step instead of two. A similar

formulation will be developed for the output matrix $\mathbf{C}$, where we will define the output vector and convert back to physical coordinates in one step. The method described here can be used for both transient and frequency response solutions.

One might ask why we are going to all the trouble of doing a state space version of modal analysis. Chapter 5 showed that given the state space equations of motion of a system, we can use MATLAB to solve for both frequency and time domain responses without knowing anything about eigenvalues and eigenvectors. The reason we are going to this trouble is that most mechanical simulations are performed using finite element techniques, where the equations of motion are too numerous to be able to be used directly in MATLAB or in a servo system simulation. Since modal analysis results, the eigenvalues and eigenvectors, are available from an ANSYS eigenvalue solution, it would be nice if we knew how to use these results by developing them into a MATLAB state space model. We could then use the power of MATLAB to perform any further analysis.

The techniques described above can be further extended by taking the results set from a large finite element problem and defining a small state space model that accurately describes the pertinent dynamics of the system (Chapters 15 to 19). The small MATLAB state space model can then be used in lieu of the frequency and transient analysis capabilities in the finite element program. The MATLAB state space model can also be combined with a servo system model, allowing complete servo-mechanical system simulations.

10.2 Eigenvalue Problem

We start with the undamped homogeneous equations of motion in state space form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} \quad (10.1)$$

In Chapter 7 we defined a normal mode as:

$$x_i = x_{mi} \sin(\omega_i t + \phi_i) = x_{mi} \operatorname{Im}(e^{j\omega_i t + \phi_i}) \quad (10.2)$$

For our three degree of freedom (z_1 to z_3), six state (x_1 to x_6) system, for the i^{th} eigenvalue and eigenvector, the equation would appear as:

$$\begin{bmatrix} z_{1i} \\ \dot{z}_{1i} \\ z_{2i} \\ \dot{z}_{2i} \\ z_{3i} \\ \dot{z}_{3i} \end{bmatrix} = \begin{bmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \\ x_{4i} \\ x_{5i} \\ x_{6i} \end{bmatrix} = \mathbf{x}_{mi} \sin(\omega_i t + \phi_i) = \begin{bmatrix} x_{m1i} \\ x_{m2i} \\ x_{m3i} \\ x_{m4i} \\ x_{m5i} \\ x_{m6i} \end{bmatrix} \sin(\omega_i t + \phi_i) \quad (10.3)$$

Differentiating the modal displacement equation above to get the modal velocity equation:

$$\begin{aligned} \frac{d}{dt} [\mathbf{x}_{mi} \sin(\omega_i t + \phi_i)] &= \mathbf{x}_{mi} \frac{d}{dt} \text{Im} [e^{j(\omega_i t + \phi_i)}] \\ &= \mathbf{x}_{mi} \text{Im} [j\omega_i e^{j(\omega_i t + \phi_i)}] \\ &= \mathbf{x}_{mi} \text{Im} [j\omega_i (\cos(\omega_i t + \phi_i) + j\sin(\omega_i t + \phi_i))] \\ &= \mathbf{x}_{mi} \text{Im} [j\omega_i \cos(\omega_i t + \phi_i) - \omega_i \sin(\omega_i t + \phi_i)] \\ &= \mathbf{x}_{mi} \omega_i \cos(\omega_i t + \phi_i) \end{aligned} \quad (10.4)$$

Substituting the derivatives into the state equation we arrive at the eigenvalue problem:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} \\ j\omega_i \mathbf{x}_{mi} \sin(\omega_i t + \phi_i) &= \mathbf{A}\mathbf{x}_{mi} \sin(\omega_i t + \phi_i) \\ j\omega_i \mathbf{x}_{mi} &= \mathbf{A}\mathbf{x}_{mi} \\ (j\omega_i \mathbf{I} - \mathbf{A})\mathbf{x}_{mi} &= 0 \end{aligned} \quad (10.5)$$

10.3 Eigenvalue Problem – Laplace Transform

We can also use Laplace transforms to define the eigenvalue problem. Taking the matrix Laplace transform of the homogeneous state equation and solving for $\mathbf{x}(s)$:

$$\begin{aligned} s\mathbf{I}\mathbf{x}(s) &= \mathbf{A}\mathbf{x}(s) \\ (s\mathbf{I} - \mathbf{A})\mathbf{x}(s) &= 0 \end{aligned} \quad (10.6a,b)$$

This is another form of the eigenvalue problem, again where the determinant of the term $(s\mathbf{I} - \mathbf{A})$ has to equal zero to have anything other than a trivial solution.

$$|(s\mathbf{I} - \mathbf{A})| = 0 \quad (10.7)$$

Letting $m_1 = m_2 = m_3 = m$, $k_1 = k_2 = k$, $c_1 = c_2 = 0$ and rewriting the matrix equations of motion to match the original undamped problem used in (6.8).

$$(s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s & -1 & 0 & 0 & 0 & 0 \\ \frac{k}{m} & s & \frac{-k}{m} & 0 & 0 & 0 \\ 0 & 0 & s & -1 & 0 & 0 \\ \frac{-k}{m} & 0 & \frac{2k}{m} & s & \frac{-k}{m} & 0 \\ 0 & 0 & 0 & 0 & s & -1 \\ 0 & 0 & \frac{-k}{m} & 0 & \frac{k}{m} & s \end{bmatrix} \quad (10.8)$$

In Section 6.3 we used this form of the equation to find the state space transfer function matrix, where we needed the inverse of $(s\mathbf{I} - \mathbf{A})$. Here we need the determinant of $(s\mathbf{I} - \mathbf{A})$. Using a symbolic algebra program results in the following characteristic equation:

$$\begin{aligned} s^2(m^3s^4 + 4m^2ks^2 + 3mk^2) &= 0 \\ m^3s^6 + 4m^2ks^4 + 3mk^2s^2 &= 0 \end{aligned} \quad (10.9a,b)$$

This is the same equation we found in (2.58) for the characteristic equation, whose roots were found to be the poles (eigenvalues). Repeating from Chapter 2, (2.67), (2.70) and (2.71):

$$s_{1,2} = 0$$

$$s_{3,4} = \pm j\sqrt{\frac{k}{m}} = \pm j1$$

$$s_{5,6} = \pm j\sqrt{\frac{3k}{m}} = \pm j1.732 \quad (10.10a,b,c)$$

In Chapter 5, the state space chapter, we showed that for arbitrary damping the eigenvalues would be complex numbers with both real and imaginary components, where the real part was indicative of there being damping in the system as the poles were offset to the left of the imaginary axis (Figure 5.3).

We defined the damped eigenvalues as $(\lambda_{n1,n2} = \sigma_{n1} \pm j\omega_{n1})$ (5.48). Note for the undamped eigenvalues above, the σ values are zero, with all poles lying on the imaginary axis.

10.4 Eigenvalue Problem – Eigenvectors

Let us now solve for the eigenvectors in state space form, going back to the original equations of motion for the i^{th} mode, similar to (10.5):

$$\begin{bmatrix} j\omega_i & -1 & 0 & 0 & 0 & 0 \\ \frac{k}{m} & j\omega_i & \frac{-k}{m} & 0 & 0 & 0 \\ 0 & 0 & j\omega_i & -1 & 0 & 0 \\ \frac{-k}{m} & 0 & \frac{2k}{m} & j\omega_i & \frac{-k}{m} & 0 \\ 0 & 0 & 0 & 0 & j\omega_i & -1 \\ 0 & 0 & \frac{-k}{m} & 0 & \frac{k}{m} & j\omega_i \end{bmatrix} \begin{bmatrix} x_{m1i} \\ x_{m2i} \\ x_{m3i} \\ x_{m4i} \\ x_{m5i} \\ x_{m6i} \end{bmatrix} = 0 \quad (10.11)$$

Expanding the equations:

$$\begin{aligned} j\omega_i x_{m1i} - x_{m2i} &= 0 \\ \frac{k}{m} x_{m1i} + j\omega_i x_{m2i} - \frac{k}{m} x_{m3i} &= 0 \\ j\omega_i x_{m3i} - x_{m4i} &= 0 \\ -\frac{k}{m} x_{m1i} + \frac{2k}{m} x_{m3i} + j\omega_i x_{m4i} - \frac{k}{m} x_{m5i} &= 0 \\ j\omega_i x_{m5i} - x_{m6i} &= 0 \\ -\frac{k}{m} x_{m3i} + \frac{k}{m} x_{m5i} + j\omega_i x_{m6i} &= 0 \end{aligned} \quad (10.12a-f)$$

Dropping the “m” and “i” terms from the eigenvectors:

$$\begin{aligned}
j\omega_1 x_1 - x_2 &= 0 \\
\frac{k}{m} x_1 + j\omega_1 x_2 - \frac{k}{m} x_3 &= 0 \\
j\omega_1 x_3 - x_4 &= 0 \\
-\frac{k}{m} x_1 + \frac{2k}{m} x_3 + j\omega_1 x_4 - \frac{k}{m} x_5 &= 0 \\
j\omega_1 x_5 - x_6 &= 0 \\
-\frac{k}{m} x_3 + \frac{k}{m} x_5 + j\omega_1 x_6 &= 0
\end{aligned} \tag{10.13a-f}$$

Selecting the first state, x_1 , as a reference and solving for x_2 through x_6 in terms of x_1 .

Solving for x_2 from (10.13a):

$$\begin{aligned}
j\omega_1 x_1 - x_2 &= 0 \\
x_2 &= j\omega_1 x_1 \\
\frac{x_2}{x_1} &= j\omega_1
\end{aligned} \tag{10.14}$$

Solving for x_3 from (10.13b):

$$\begin{aligned}
\frac{k}{m} x_1 + j\omega_1 x_2 - \frac{k}{m} x_3 &= 0 \\
j\omega_1 (j\omega_1 x_1) - \frac{k}{m} x_3 &= -\frac{k}{m} x_1 \\
-\frac{k}{m} x_3 &= -\frac{k}{m} x_1 + \omega_1^2 x_1 \\
x_3 &= \left(\frac{k - \omega_1^2 m}{k} \right) x_1 \\
\frac{x_3}{x_1} &= \frac{k - \omega_1^2 m}{k}
\end{aligned} \tag{10.15}$$

Solving for x_4 from (10.13c):

$$\begin{aligned}
j\omega_1 x_3 - x_4 &= 0 \\
j\omega_1 \left(\frac{k - \omega_1^2 m}{k} \right) x_1 - x_4 &= 0 \\
x_4 &= j\omega_1 \left(\frac{k - \omega_1^2 m}{k} \right) x_1 = j\omega_1 x_3 \\
\frac{x_4}{x_1} &= j\omega_1 \left(\frac{k - \omega_1^2 m}{k} \right) = j\omega_1 \frac{x_3}{x_1}
\end{aligned} \tag{10.16}$$

Solving for x_5 from (10.13d):

$$\begin{aligned}
-\frac{k}{m} x_1 + \frac{2k}{m} x_3 + j\omega_1 x_4 - \frac{k}{m} x_5 &= 0 \\
-\frac{k}{m} x_1 + \frac{2k}{m} \left(\frac{k - \omega_1^2 m}{k} \right) x_1 \\
+ j\omega_1 \left[j\omega_1 \left(\frac{k - \omega_1^2 m}{k} \right) \right] x_1 - \frac{k}{m} x_5 &= 0 \\
x_5 &= \left(\frac{m^2 \omega_1^4 - 3mk\omega_1^2 + k^2}{k^2} \right) x_1 \\
\frac{x_5}{x_1} &= \frac{m^2 \omega_1^4 - 3mk\omega_1^2 + k^2}{k^2}
\end{aligned} \tag{10.17}$$

Solving for x_6 from (10.13e):

$$\begin{aligned}
j\omega_1 x_5 - x_6 &= 0 \\
x_6 &= j\omega_1 \left(\frac{m^2 \omega_1^4 - 3mk\omega_1^2 + k^2}{k^2} \right) x_1 = j\omega_1 x_5 \\
\frac{x_6}{x_1} &= j\omega_1 \left(\frac{m^2 \omega_1^4 - 3mk\omega_1^2 + k^2}{k^2} \right) = j\omega_1 \frac{x_5}{x_1}
\end{aligned} \tag{10.18}$$

Note that the results for the displacement eigenvector components in (10.15) and (10.17) match the two displacement eigenvectors calculated in (7.24) and (7.29), respectively. Also note that all three velocity eigenvector components are equal to $j\omega_1$ times their respective displacement eigenvector components.

Unlike the complex eigenvectors found in Chapter 5 for the damped model, these undamped eigenvector displacement states are all real; they have no complex terms.

10.5 Modal Matrix

We will see that when we transform to principal coordinates, create the state equations in principal coordinates and back transform results to physical coordinates we only require a 3x3 displacement modal matrix. This is because we can transform positions and velocities separately. The modal matrix (7.46) and normalized modal matrix (7.77) are repeated below, again for $m = k = 1$:

$$\mathbf{z}_m = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & -2 \\ 1 & -1 & 1 \end{bmatrix} \quad (10.19)$$

$$\mathbf{z}_n = \frac{1}{\sqrt{m}} \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & \frac{-2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{bmatrix} = \begin{bmatrix} 0.5774 & -0.707 & 0.4082 \\ 0.5774 & 0 & -0.8165 \\ 0.5774 & 0.707 & 0.4082 \end{bmatrix} \quad (10.20)$$

10.6 MATLAB Code `tdofss_eig.m`: Solving for Eigenvalues and Eigenvectors

10.6.1 Code Description

The MATLAB code `tdofss_eig.m` solves for the eigenvalues and eigenvectors in the state space form of the system. The code will be listed in sections with commented results and explanations following each section.

10.6.2 Eigenvalue Calculation

```
% tdofss_eig.m eigenvalue problem solution for tdof undamped model

clear all;

% define the values of masses, springs, dampers and forces

m1 = 1;
m2 = 1;
```

```

m3 = 1;

c1 = 0;
c2 = 0;

k1 = 1;
k2 = 1;

% define the system matrix, a

a = [ 0      1      0      0      0      0
      -k1/m1  -c1/m1  k1/m1  c1/m1  0      0
        0      0      0      1      0      0
      k1/m2   c1/m2 -(k1+k2)/m2 -(c1+c2)/m2 k2/m2  c2/m2
        0      0      0      0      0      1
        0      0      k2/m3  c2/m3  -k2/m3  -c2/m3];

% solve for the eigenvalues of the system matrix

[xm,omega] = eig(a)

```

The resulting eigenvalues, in units of rad/sec, are below. Note that MATLAB uses “i” for imaginary numbers instead of “j” which is used in the text.

```

omega =
Columns 1 through 4
    0 + 1.7321i    0    0    0
    0    0 - 1.7321i    0    0
    0    0    0    0
    0    0    0    0 + 1.0000i
    0    0    0    0
    0    0    0    0
Columns 5 through 6
    0    0
    0    0
    0    0
    0    0
    0 - 1.0000i    0
    0    0

```

The eigenvalues, what MATLAB calls “generalized eigenvalues,” are the diagonal elements of the omega matrix. The six values that MATLAB calculates are: $1.7321i$, $-1.7321i$, 0 , $1.0000i$, $-1.0000i$, 0 , in that order. These are the same values we found using our closed form calculations. Also, the values are all imaginary, as we would expect with a system with no damping and as we found above from our $|(s\mathbf{I} - \mathbf{A})|=0$ derivation.

10.6.3 Eigenvector Calculation

The resulting eigenvectors, directly from MATLAB output are:

xm =			
Columns 1 through 4			
0.2041	0.2041	0.5774	0 + 0.5000i
0 + 0.3536i	0 - 0.3536i	0	-0.5000
-0.4082	-0.4082	0.5774	0 + 0.0000i
0 - 0.7071i	0 + 0.7071i	0	0.0000
0.2041	0.2041	0.5774	0 - 0.5000i
0 + 0.3536i	0 - 0.3536i	0	0.5000
Columns 5 through 6			
0 - 0.5000i	-0.5774		
-0.5000	0.0000		
0 - 0.0000i	-0.5774		
0.0000	0.0000		
0 + 0.5000i	-0.5774		
0.5000	0.0000		

Note that unlike the eigenvectors calculated in the Modal Analysis section, which had three rows, these eigenvectors each have six rows, each row corresponding to its respective state. Repeating the state definitions from (5.4) to (5.9):

$$\begin{aligned}
 x_1 &= z_1 \text{ Position of Mass 1} \\
 x_2 &= \dot{z}_1 \text{ Velocity of Mass 1} \\
 x_3 &= z_2 \text{ Position of Mass 2} \\
 x_4 &= \dot{z}_2 \text{ Velocity of Mass 2} \\
 x_5 &= z_3 \text{ Position of Mass 3} \\
 x_6 &= \dot{z}_3 \text{ Velocity of Mass 3}
 \end{aligned}$$

Thus, the first, third and fifth rows represent the positions of the three masses for each mode, and the second, fourth and sixth rows represent the velocities of the three masses for each mode. Separating into position and velocity components:

xm(position) =					
0.2041	0.2041	0.5774	0 + 0.5000i	0 - 0.5000i	-0.5774
-0.4082	-0.4082	0.5774	0 + 0.0000i	0 - 0.0000i	-0.5774
0.2041	0.2041	0.5774	0 - 0.5000i	0 + 0.5000i	-0.5774
xm(velocity) =					
0 + 0.3536i	0 - 0.3536i	0	-0.5000	-0.5000	0.0000
0 - 0.7071i	0 + 0.7071i	0	0.0000	0.0000	0.0000
0 + 0.3536i	0 - 0.3536i	0	0.5000	0.5000	0.0000

What is the relationship between the position and velocity terms in each of the eigenvectors? Once again, knowing that at each undamped frequency a normal mode exists and that the position and velocity can be defined as:

$$\mathbf{z}_i = \mathbf{z}_{ni} e^{j\omega_i t} \quad (10.21a,b)$$

$$\dot{\mathbf{z}}_i = j\omega_i \mathbf{z}_{ni} e^{j\omega_i t}$$

Taking the amplitudes of the position and velocity:

$$|\dot{\mathbf{z}}_n| = \omega |\mathbf{z}_n| \quad (10.22)$$

The amplitude of the velocity eigenvector terms should be equal to the eigenvalue times its respective position eigenvector term. The fact that the velocity entries are complex numbers by virtue of multiplying the “real” position eigenvector entries by the eigenvalue does not make the eigenvectors “complex,” but refers to the fact that in the undamped case velocity is 90° out of phase with position.

Checking the first eigenvector by multiplying the position term (state 1) by the eigenvalue to get the velocity term (state 2): (highlighted in bold type above)

$$0.2041 * 1.7321j = .3535j \quad (10.23)$$

Note that for the third and sixth eigenvectors, which have zero eigenvalues, the velocity entries are zero because the position entry is multiplied by zero.

10.6.4 MATLAB Eigenvectors – Real and Imaginary Values

It is interesting to see how MATLAB handles real and imaginary values in its eigenvectors.

xm =					
0.2041	0.2041	0.5774	0 + 0.5000i	0 - 0.5000i	-0.5774
0 + 0.3536i	0 - 0.3536i	0	-0.5000	-0.5000	0.0000
-0.4082	-0.4082	0.5774	0 + 0.0000i	0 - 0.0000i	-0.5774
0 - 0.7071i	0 + 0.7071i	0	0.0000	0.0000	0.0000
0.2041	0.2041	0.5774	0 - 0.5000i	0 + 0.5000i	-0.5774
0 + 0.3536i	0 - 0.3536i	0	0.5000	0.5000	0.0000

We know that the position and velocity entries are related by “j” times the eigenvalue, but why are some position eigenvector entries real and some imaginary? For example, the position eigenvector entries for all except the

mode at 1 rad/sec (the fourth and fifth columns), are real, while the fourth and fifth column position entries are imaginary. From the original normal modes analysis, we know that only the **ratios** of eigenvector entries are important, and that the eigenvectors can be normalized in several fashions. Therefore, each eigenvector can be multiplied by an arbitrary constant. The fourth and fifth eigenvectors can be multiplied by “i” to make their position entries real for consistency with the hand-calculated results.

10.6.5 Sorting Eigenvalues / Eigenvectors

Typically some housekeeping is done on the eigenvalues and eigenvectors before continuing, sorting the eigenvalues from small to large (done by default in ANSYS), rearranging the eigenvectors accordingly and checking for eigenvectors with imaginary position entries and converting them to real by multiplying by “i.” Also, the signs of the real portion of state 1 are set positive to ensure that sets of eigenvectors are complex conjugates of each other for consistency.

Continuing the listing of **tdofss_eig.m**, showing the sorting code:

```
% take the diagonal elements of the generalized eigenvalue matrix omega
omegad = diag(omega);

% in real problems, we would next convert to hz from radians/sec
omegahz = omegad/(2*pi);

% now reorder the eigenvalues and eigenvectors from low to high frequency,
% keeping track of how the eigenvalues are ordered to reorder the
% eigenvectors to match, using indexhz

[omegaorder,indexhz] = sort(abs(imag(omegad)))

for cnt = 1:length(omegad)

    omegao(cnt,1) = omegad(indexhz(cnt)); % reorder eigenvalues

    xmo(:,cnt) = xm(:,indexhz(cnt)); % reorder eigenvector columns

end

omegao

xmo

% check for any eigenvectors with imaginary position elements by checking
% the first three position entries for each eigenvector (first, third and
% and fifth rows) and convert to real
```

```

for cnt = 1:length(omegad)
    if (real(xmo(1,cnt)) & real(xmo(3,cnt)) & real(xmo(5,cnt))) == 0
        xmo(:,cnt) = i*(xmo(:,cnt)); % convert whole column if imaginary
    else
    end
end

xmo

% check for any eigenvectors with negative position elements for the first
% displacement, if so change to positive to that eigenvectors for the same mode
% are complex conjugates

for cnt = 1:length(omegad)
    if real(xmo(1,cnt)) < 0
        xmo(:,cnt) = -1*(xmo(:,cnt)); % convert whole column if negative
    else
    end
end

xmo

```

Printing the results of the MATLAB reordering:

```

omegaorder =
    0
    0
    1.0000      These are the re-ordered eigenvalues, from low to high.
    1.0000
    1.7321
    1.7321
indexhz =
    3
    6
    4      This is the ordering of the original eigenvalues.
    5
    1
    2
omegao =
    0
    0
    0 + 1.0000i
    0 - 1.0000i
    0 + 1.7321i
    0 - 1.7321i

```

Here are the reordered eigenvectors.

```

xmo =
Columns 1 through 4
    0.5774      -0.5774      0 + 0.5000i      0 - 0.5000i
    0           0.0000      -0.5000      -0.5000
    0.5774      -0.5774      0 + 0.0000i      0 - 0.0000i
    0           0.0000      0.0000      0.0000
    0.5774      -0.5774      0 - 0.5000i      0 + 0.5000i
    0           0.0000      0.5000      0.5000
Columns 5 through 6
    0.2041      0.2041
    0 + 0.3536i  0 - 0.3536i
    -0.4082     -0.4082
    0 - 0.7071i  0 + 0.7071i
    0.2041      0.2041
    0 + 0.3536i  0 - 0.3536i

```

Here the converting of imaginary position values to real is performed, note that the third and fourth eigenvectors are converted.

```

xmo =
Columns 1 through 4
0.5774      -0.5774      -0.5000      0.5000
0           0.0000      0 - 0.5000i    0 - 0.5000i
0.5774      -0.5774      0.0000      0.0000
0           0.0000      0 - 0.0000i    0 - 0.0000i
0.5774      -0.5774      0.5000      -0.5000
0           0.0000      0 + 0.5000i    0 + 0.5000i
Columns 5 through 6
0.2041      0.2041
0 + 0.3536i  0 - 0.3536i
-0.4082     -0.4082
0 - 0.7071i  0 + 0.7071i
0.2041      0.2041
0 + 0.3536i  0 - 0.3536i

```

In this step the first row elements are checked to see that they are positive; if not, the column is multiplied by -1 .

```

xmo =
Columns 1 through 4
0.5774      0.5774      0.5000      0.5000
0          -0.0000      0 + 0.5000i    0 - 0.5000i
0.5774      0.5774      0.0000      0.0000
0          -0.0000      0 + 0.0000i    0 - 0.0000i
0.5774      0.5774     -0.5000     -0.5000
0          -0.0000      0 - 0.5000i    0 + 0.5000i
Columns 5 through 6
0.2041      0.2041
0 + 0.3536i  0 - 0.3536i
-0.4082     -0.4082
0 - 0.7071i  0 + 0.7071i
0.2041      0.2041
0 + 0.3536i  0 - 0.3536i

```

10.6.6 Normalizing Eigenvectors

Now that the eigenvalues and eigenvectors are available, we can normalize the eigenvectors with respect to mass. Then we will check the resulting diagonalization by multiplying the original mass and stiffness matrices by the normalized eigenvectors to see if the mass matrix becomes the identity matrix and the stiffness matrix becomes a diagonal matrix with squares of the eigenvalues on the diagonal (spectral matrix).

Since we need to deal only with the displacement entries of the 6x6 modal matrix in order to transform the 3x3 mass and stiffness matrices, the $\mathbf{x}_m$ matrix below is a 3x3 matrix with only displacement entries.

Reviewing, the mass matrix is diagonalized by pre- and postmultiplying by the normalized eigenvector matrix:

$$\mathbf{x}_n^T \mathbf{m} \mathbf{x}_n = \mathbf{I} , \quad (10.24)$$

yielding the identity matrix. The stiffness matrix is also diagonalized by pre- and postmultiplying by the normalized eigenvector matrix:

$$\mathbf{x}_n^T \mathbf{k} \mathbf{x}_n = \mathbf{k}_p , \quad (10.25)$$

yielding the stiffness matrix in principal coordinates, the spectral matrix, a diagonal matrix with squares of the eigenvalues on the diagonal.

Repeating from Section 7.4.2, the normalized modal matrix $\mathbf{x}_n$ is made up of eigenvectors as defined below:

$$\mathbf{x}_{ni} = \frac{\mathbf{x}_{mi}}{\left[\mathbf{x}_{mi}^T \mathbf{m} \mathbf{x}_{mi} \right]^{\frac{1}{2}}} = \frac{\mathbf{x}_{mi}}{q_i} \quad (10.26)$$

Where q_i is defined as:

$$q_i = \left[\sum_{j=1}^n x_{mji} \left(\sum_{k=1}^n m_{jk} x_{mki} \right) \right]^{\frac{1}{2}} \quad (10.27)$$

For a diagonal mass matrix, simplifying q because all the m_{jk} terms are zero:

$$q_i = \left[\sum_{k=1}^n m_k x_{mki}^2 \right]^{\frac{1}{2}} \quad (10.28)$$

Continuing with code from `tdofss_eig.m`:

```
% define the mass and stiffness matrices for normalization of eigenvectors
% and for checking values in principal coordinates

m = [m1    0    0
```

```

        0   m2   0
        0   0  m3];

k = [ k1   -k1   0
      -k1  k1+k2 -k2
        0   -k2  k2];

% define the position eigenvectors by taking the first, third and fifth
% rows of the original six rows in xmo

xmop1 = [xmo(1,:); xmo(3,:); xmo(5,:)]

% define the three eigenvectors for the three degrees of freedom by taking
% the second, fourth and sixth columns

xmop = [xmop1(:,2) xmop1(:,4) xmop1(:,6)]

% normalize with respect to mass

for mode = 1:3

    xn(:,mode) = xmop(:,mode)/sqrt(xmop(:,mode)'*m*xmop(:,mode));

end

xn

% calculate the normalized mass and stiffness matrices for checking

mm = xn'*m*xn

km = xn'*k*xn

% check that the sqrt of diagonal elements of km are eigenvalues

p = (diag(km)).^0.5;

[p abs(imag(omegao(1:2:5;)))]

% rename the three eigenvalues for convenience in later calculations

w1 = abs(imag(omegao(1)));

w2 = abs(imag(omegao(3)));

w3 = abs(imag(omegao(5)));

```

Back to MATLAB output, with comments added in bold type:

```

Repeating xmo, the full, rearranged eigenvector matrix:

xmo =

Columns 1 through 4

```

0.5774	0.5774	0.5000	0.5000
0	-0.0000	0 + 0.5000i	0 - 0.5000i
0.5774	0.5774	0.0000	0.0000
0	-0.0000	0 + 0.0000i	0 - 0.0000i
0.5774	0.5774	-0.5000	-0.5000
0	-0.0000	0 - 0.5000i	0 + 0.5000i

Columns 5 through 6

0.2041	0.2041
0 + 0.3536i	0 - 0.3536i
-0.4082	-0.4082
0 - 0.7071i	0 + 0.7071i
0.2041	0.2041
0 + 0.3536i	0 - 0.3536i

Taking only the position rows:

xmop1 =

0.5774	0.5774	0.5000	0.5000	0.2041	0.2041
0.5774	0.5774	0.0000	0.0000	-0.4082	-0.4082
0.5774	0.5774	-0.5000	-0.5000	0.2041	0.2041

Taking every other column to form the 3x3 position eigenvector matrix:

xmop =

0.5774	0.5000	0.2041
0.5774	0.0000	-0.4082
0.5774	-0.5000	0.2041

Normalizing with respect to mass:

xn =

0.5774	0.7071	0.4082
0.5774	0.0000	-0.8165
0.5774	-0.7071	0.4082

Checking the mass matrix in principal coordinates, should be the identity matrix:

mm =

1.0000	-0.0000	0.0000
-0.0000	1.0000	-0.0000
0.0000	-0.0000	1.0000

Checking the stiffness matrix in principal coordinates, should be squares of eigenvalues:

km =

0.0000	-0.0000	0.0000
-0.0000	1.0000	-0.0000
0.0000	-0.0000	3.0000

Comparing the square root of the diagonal elements of the stiffness matrix in principal coordinates with the eigenvalues:

ans =

0.0000	0
1.0000	1.0000
1.7321	1.7321

10.6.7 Writing Homogeneous Equations of Motion

Now that we know the eigenvalues, we can write the homogeneous equations of motion in the principal coordinate system by inspection. We can also use the normalized eigenvectors to transform the forcing function and initial conditions to principal coordinates, yielding the complete solution for either transient or frequency domain problems in principal coordinates. We can then back-transform to the physical coordinate system to get the desired results in physical coordinates. Through the modal formulation we can define the contributions of various modes to the total response.

For a problem of this size, there is no need to use the modal formulation. When solving real problems, however, whether they be large MATLAB based problems or ANSYS finite element models, using the modal formulation has advantages. As mentioned earlier, ANSYS gives the eigenvalues and eigenvectors normalized with respect to mass as normal output of an eigenvalue run. Therefore, all one has to do to solve in MATLAB is to take that ANSYS output information and build the equations of motion in state space form and solve, taking advantage of the flexibility, plotting capability and speed of MATLAB to perform other studies. The modal approach is what gives us the capability to create complete state space models of the system mechanical dynamics in a form that can be used by the servo engineers in their state space servo/mechanical models.

10.6.7.1 Equations of Motion – Physical Coordinates

We will start with the equations of motion in physical coordinates with forces as shown in (10.29) and assume zero initial conditions. The reason we are assuming zero initial conditions is that converting initial conditions requires the inverse of the complete modal matrix, which is not convenient when using ANSYS modal results to build a reduced (smaller size) model. Fortunately, a large majority of real life problems can be solved with zero initial conditions.

$$\begin{aligned}
 m\ddot{z}_1 + kz_1 - kz_2 &= F_1 \\
 m\ddot{z}_2 - kz_1 + 2kz_2 - kz_3 &= F_2 \\
 m\ddot{z}_3 - kz_2 + kz_3 &= F_3 \\
 \text{IC's: } z_1, z_2, z_3, \dot{z}_1, \dot{z}_2, \dot{z}_3 &= 0
 \end{aligned}
 \tag{10.29}$$

Knowing the eigenvalues and eigenvectors normalized with respect to mass, we can write the damped homogeneous equations of motion in principal coordinates by inspection. The forces in principal coordinates, F_{p1} , F_{p2} and F_{p3} are obtained by premultiplying the force vector in physical coordinates by the transpose of the normalized eigenvector:

$$\mathbf{F}_p = \mathbf{x}_n^T \mathbf{F} \quad (10.30)$$

$\mathbf{x}_n$ was defined in (10.20) as a 3x3 matrix of normalized displacement eigenvectors. The multiplication then results in a 3x1 vector of forces in principal coordinates. The resulting elements are entered in the appropriate positions in the equations in principal coordinates below.

10.6.7.2 Equations of Motion – Principal Coordinates

The three equations of motion in principal coordinates become:

$$\begin{aligned} \ddot{x}_{p1} &= F_{p1} \\ \ddot{x}_{p2} + 2\zeta_2\omega_2\dot{x}_{p2} + \omega_2^2 x_{p2} &= F_{p2} \\ \ddot{x}_{p3} + 2\zeta_3\omega_3\dot{x}_{p3} + \omega_3^2 x_{p3} &= F_{p3} \end{aligned} \quad (10.31a,b,c)$$

where ω_1 , ω_2 , and ω_3 are the three eigenvalues, with units of radians/sec. The “zeta” terms, ζ_1 , ζ_2 and ζ_3 , represent the percentages of critical damping for each of the three modes, all of which can be different and are typically obtained from experimental results. For example, 2% of critical damping would give a ζ value of 0.02.

Now we can convert the second order differential equations above to state space form by solving for the highest derivative:

$$\begin{aligned} \ddot{x}_{p1} &= F_{p1} \\ \ddot{x}_{p2} &= F_{p2} - \omega_2^2 x_{p2} - 2\zeta_2\omega_2\dot{x}_{p2} \\ \ddot{x}_{p3} &= F_{p3} - \omega_3^2 x_{p3} - 2\zeta_3\omega_3\dot{x}_{p3} \end{aligned} \quad (10.32a,b,c)$$

Defining states:

$$\begin{aligned}
 x_1 &= x_{p1} && \text{displacement of mode 1 (not of mass 1)} \\
 x_2 &= \dot{x}_{p1} && \text{derivative of displacement of mode 1} \\
 x_3 &= x_{p2} && \text{displacement of mode 2} \\
 x_4 &= \dot{x}_{p2} && \text{derivative of displacement of mode 2} \\
 x_5 &= x_{p3} && \text{displacement of mode 3} \\
 x_6 &= \dot{x}_{p3} && \text{derivative of displacement of mode 3}
 \end{aligned}$$

Rewriting the equations of motion using the states:

$$\begin{aligned}
 \dot{x}_1 &= x_2 \\
 \dot{x}_2 &= F_{p1} \\
 \dot{x}_3 &= x_4 \\
 \dot{x}_4 &= F_{p2} - \omega_2^2 x_3 - 2\zeta_2 \omega_2 x_4 \\
 \dot{x}_5 &= x_6 \\
 \dot{x}_6 &= F_{p3} - \omega_3^2 x_5 - 2\zeta_3 \omega_3 x_6
 \end{aligned} \tag{10.33a-f}$$

Rewriting in matrix form:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \tag{10.34}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\omega_2^2 & -2\zeta_2 \omega_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\omega_3^2 & -2\zeta_3 \omega_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ F_{p1} \\ 0 \\ F_{p2} \\ 0 \\ F_{p3} \end{bmatrix} u \tag{10.35}$$

Now that the complete state space equations of motion are known, the six states in principal coordinates can be solved for their frequency and/or time domain responses.

Let us assume that we are interested in the three displacements and the three velocities. The output matrix equation then becomes, where $\mathbf{y}_p$ is the displacements in principal coordinates:

$$\mathbf{y}_p = \mathbf{C}\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \quad (10.36)$$

With the six desired outputs in principal coordinates, we can back-transform them into physical coordinates by the following transform:

$$\begin{bmatrix} z_1 \\ \dot{z}_1 \\ z_2 \\ \dot{z}_2 \\ z_3 \\ \dot{z}_3 \end{bmatrix} = \mathbf{x}_n \mathbf{y}_p = \begin{bmatrix} x_{n11} & 0 & x_{n12} & 0 & x_{n13} & 0 \\ 0 & x_{n11} & 0 & x_{n12} & 0 & x_{n13} \\ x_{n21} & 0 & x_{n22} & 0 & x_{n23} & 0 \\ 0 & x_{n21} & 0 & x_{n22} & 0 & x_{n23} \\ x_{n31} & 0 & x_{n32} & 0 & x_{n33} & 0 \\ 0 & x_{n31} & 0 & x_{n32} & 0 & x_{n33} \end{bmatrix} \begin{bmatrix} y_{p1} \\ y_{p2} \\ y_{p3} \\ y_{p4} \\ y_{p5} \\ y_{p6} \end{bmatrix}$$

$$= \begin{bmatrix} x_{n11}y_{p1} + x_{n12}y_{p3} + x_{n13}y_{p5} \\ x_{n11}y_{p2} + x_{n12}y_{p4} + x_{n13}y_{p6} \\ x_{n21}y_{p1} + x_{n22}y_{p3} + x_{n23}y_{p5} \\ x_{n21}y_{p2} + x_{n22}y_{p4} + x_{n23}y_{p6} \\ x_{n31}y_{p1} + x_{n32}y_{p3} + x_{n33}y_{p5} \\ x_{n31}y_{p2} + x_{n32}y_{p4} + x_{n33}y_{p6} \end{bmatrix} \quad (10.37)$$

Instead of doing the two multiplications shown in (10.36) and (10.37), $\mathbf{C}$ times $\mathbf{x}$ to get $\mathbf{y}_p$ and then premultiplying $\mathbf{y}_p$ by $\mathbf{x}_n$ to get the displacements and velocities in physical coordinates, we could have done a single multiplication if $\mathbf{C}$ were defined as shown in (10.38), using eigenvector entries directly in the definition:

$$\mathbf{C} = \begin{bmatrix} x_{n11} & 0 & x_{n12} & 0 & x_{n13} & 0 \\ 0 & x_{n11} & 0 & x_{n12} & 0 & x_{n13} \\ x_{n21} & 0 & x_{n22} & 0 & x_{n23} & 0 \\ 0 & x_{n21} & 0 & x_{n22} & 0 & x_{n23} \\ x_{n31} & 0 & x_{n32} & 0 & x_{n33} & 0 \\ 0 & x_{n31} & 0 & x_{n32} & 0 & x_{n33} \end{bmatrix} \quad (10.38)$$

Rewriting the output equation using $\mathbf{C}$ defined in (10.38) and expanding to see individual terms:

$$\begin{bmatrix} z_1 \\ \dot{z}_1 \\ z_2 \\ \dot{z}_2 \\ z_3 \\ \dot{z}_3 \end{bmatrix} = \mathbf{C}\mathbf{x} = \begin{bmatrix} x_{n11} & 0 & x_{n12} & 0 & x_{n13} & 0 \\ 0 & x_{n11} & 0 & x_{n12} & 0 & x_{n13} \\ x_{n21} & 0 & x_{n22} & 0 & x_{n23} & 0 \\ 0 & x_{n21} & 0 & x_{n22} & 0 & x_{n23} \\ x_{n31} & 0 & x_{n32} & 0 & x_{n33} & 0 \\ 0 & x_{n31} & 0 & x_{n32} & 0 & x_{n33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \quad (10.39)$$

$$= \begin{bmatrix} x_{n11}x_1 + x_{n12}x_3 + x_{n13}x_5 \\ x_{n11}x_2 + x_{n12}x_4 + x_{n13}x_6 \\ x_{n21}x_1 + x_{n22}x_3 + x_{n23}x_5 \\ x_{n21}x_2 + x_{n22}x_4 + x_{n23}x_6 \\ x_{n31}x_1 + x_{n32}x_3 + x_{n33}x_5 \\ x_{n31}x_2 + x_{n32}x_4 + x_{n33}x_6 \end{bmatrix} = \begin{bmatrix} x_{n11}y_{p1} + x_{n12}y_{p3} + x_{n13}y_{p5} \\ x_{n11}y_{p2} + x_{n12}y_{p4} + x_{n13}y_{p6} \\ x_{n21}y_{p1} + x_{n22}y_{p3} + x_{n23}y_{p5} \\ x_{n21}y_{p2} + x_{n22}y_{p4} + x_{n23}y_{p6} \\ x_{n31}y_{p1} + x_{n32}y_{p3} + x_{n33}y_{p5} \\ x_{n31}y_{p2} + x_{n32}y_{p4} + x_{n33}y_{p6} \end{bmatrix}$$

10.6.8 Individual Mode Contributions, Modal State Space Form

In Section 8.7 we discussed in detail how individual modes contribute to the overall frequency response. Here we will show how to calculate individual modal contributions in modal state space form.

We start with repeating (10.35), the modal state space equations of motion.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\omega_2^2 & -2\zeta_2\omega_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\omega_3^2 & -2\zeta_3\omega_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ F_{p1} \\ 0 \\ F_{p2} \\ 0 \\ F_{p3} \end{bmatrix} \mathbf{u} \quad (10.40)$$

Notice how the three sets of uncoupled first order equations in (10.40) appear as blocks of 2x2 coefficients along the diagonal. Note also that if the eigenvalues, ω_i , and damping ratios, ζ_i , are known, the entire system matrix **A** can be filled out by inspection, as we will do in future chapters where ANSYS results are used to automatically build a model.

The first 2x2 block along the diagonal

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad (10.41)$$

represents the response of the first mode, the second 2x2 block

$$\begin{bmatrix} 0 & 1 \\ -\omega_2^2 & -2\zeta_2\omega_2 \end{bmatrix} \quad (10.42)$$

represents the response of the second mode and the third 2x2 block

$$\begin{bmatrix} 0 & 1 \\ -\omega_3^2 & -2\zeta_3\omega_3 \end{bmatrix} \quad (10.43)$$

represents the response of the third mode.

Note that the three modes are not coupled and the equations of motion in state space modal form may be rewritten separately as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ F_{p1} \end{bmatrix} u \quad \text{mode 1} \quad (10.44)$$

$$\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_2^2 & -2\zeta_2\omega_2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ F_{p2} \end{bmatrix} u \quad \text{mode 2} \quad (10.45)$$

$$\begin{bmatrix} \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_3^2 & -2\zeta_3\omega_3 \end{bmatrix} \begin{bmatrix} x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ F_{p3} \end{bmatrix} u \quad \text{mode 3} \quad (10.46)$$

For the output equation, defining a version of (10.42) which will output only displacements, not velocities:

$$\mathbf{z} = \mathbf{y} = \mathbf{C}\mathbf{x} \quad (10.47)$$

Expanding:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_{n11} & 0 & x_{n12} & 0 & x_{n13} & 0 \\ x_{n21} & 0 & x_{n22} & 0 & x_{n23} & 0 \\ x_{n31} & 0 & x_{n32} & 0 & x_{n33} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \quad (10.48)$$

Similarly, the output equations can be written separately as (10.49) to (10.51), where the $z_{31,m3}$ subscript notation stands for the displacement of mass 3 due to force at mass 1 contributed by mode 3. Here we are dealing with only the z_{11} transfer function. The modal contributions to any of the four unique transfer functions can be solved in a similar fashion.

$$\begin{bmatrix} z_{11,m1} \\ z_{21,m1} \\ z_{31,m1} \end{bmatrix} = \begin{bmatrix} x_{n11} & 0 \\ x_{n21} & 0 \\ x_{n31} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{mode 1} \quad (10.49)$$

$$\begin{bmatrix} z_{11,m2} \\ z_{21,m2} \\ z_{31,m2} \end{bmatrix} = \begin{bmatrix} x_{n12} & 0 \\ x_{n22} & 0 \\ x_{n32} & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \quad \text{mode 2} \quad (10.50)$$

$$\begin{bmatrix} z_{11,m3} \\ z_{21,m3} \\ z_{31,m3} \end{bmatrix} = \begin{bmatrix} x_{n13} & 0 \\ x_{n23} & 0 \\ x_{n33} & 0 \end{bmatrix} \begin{bmatrix} x_5 \\ x_6 \end{bmatrix} \quad \text{mode 3} \quad (10.51)$$

We are familiar with using (10.35) and (10.39) to solve for frequency responses for systems. With the use of (10.44) to (10.51) we can plot and see how each individual mode contributes to the overall response. We will examine this further in the code seen in the next chapter.

10.7 Real Modes – Argand Diagrams, Initial Condition Responses of Individual Modes

In Chapter 5, we introduced the concept of using Argand diagrams to visualize complex modes and to show how the complex eigenvector components combine to create “real” displacements and velocities. We will use the

MATLAB code **tdof_prop_damped.m** to define the eigenvectors for Argand plotting and solve for the transient responses.

The methodology followed is:

- 1) **Solve** the original undamped system equation for eigenvalues and eigenvectors.
- 2) **Plot** the eigenvectors normalized to unity using a deformed mode shape plot.
- 3) **Normalize** the displacement eigenvector entries with respect to mass to convert to principal coordinates for the proportionally damped case.
- 4) **Form** the system matrix in principal coordinates using proportional damping.
- 5) **Solve** for the eigenvalues and eigenvectors of the system matrix in principal coordinates.
- 6) **Plot** the real and imaginary displacements of each of the normal modes separately, since the three modes are uncoupled with proportional damping.
- 7) **Back transform** to physical coordinates using the normalized displacement eigenvectors.
- 8) **Plot** the real and imaginary displacements of each of the degrees of freedom separately.

For the undamped case we will use $c_1 = c_2 = 0$ and the result will be “normal” modes with “real” eigenvectors.

For proportional damping, we will start with the undamped eigenvectors and add a percentage of critical damping to each mode. This will result in “real” eigenvectors since proportional damping satisfies the Rayleigh damping criterion $\mathbf{c} = \mathbf{a}\mathbf{m} + \mathbf{b}\mathbf{k}$ as discussed in Chapter 7.

10.7.1 Undamped Model, Eigenvectors, Real Modes

The code starts with executing **tdofss_eig.m**, which calculates the eigenvalues and eigenvectors for the undamped problem, $c_1 = c_2 = 0$. The eigenvectors are then normalized with respect to unity for plotting in Argand form.

```
% tdof_prop_damped.m      proportionally damped tdof model
% solve for the eigenvalues of the undamped system model
```

```

    tdofss_eig;

    subplot(1,1,1);

% now normalize the undamped eigenvectors with respect to the position of
% mass 1, which will be set to 1.0 - for plotting of undamped Argand diagram

    for cnt = 1:length(omegad)

        xmon1(:,cnt) = xmo(:,cnt)/xmo(1,cnt);

    end

    xmon1

```

The eigenvalues and eigenvectors are:

```

omegaro =
    0          (Note the two poles
    0          at the origin)
    0 + 1.0000i
    0 - 1.0000i
    0 + 1.7321i
    0 - 1.7321i

```

```

xmron1 =
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    0         0         0 + 1.0000i  0 - 1.0000i  0 + 1.7321i  0 - 1.7321i
    1.0000    1.0000    0.0000    0.0000   -2.0000   -2.0000
    0         0         0 + 0.0000i  0 - 0.0000i  0 - 3.4641i  0 + 3.4641i
    1.0000    1.0000   -1.0000   -1.0000    1.0000    1.0000
    0         0         0 - 1.0000i  0 + 1.0000i  0 + 1.7321i  0 - 1.7321i

```

Note that the pairs of eigenvalues for each mode are complex conjugates of each other and that the pairs of eigenvectors for each mode are also complex conjugates of each other.

Once again, some eigenvector elements have complex parts. **Why do we call them “real” when they contain imaginary parts?**

“Real” eigenvectors refers to the fact that all of the **position** entries in the eigenvector are not complex numbers [i.e., not of the form $(a+jb)$], but are real

numbers. The fact that the velocity entries are complex numbers by virtue of multiplying the “real” position eigenvector entries by the eigenvalue does not make the eigenvectors “complex” but refers to the fact that in the undamped case velocity is 90° out of phase with position.

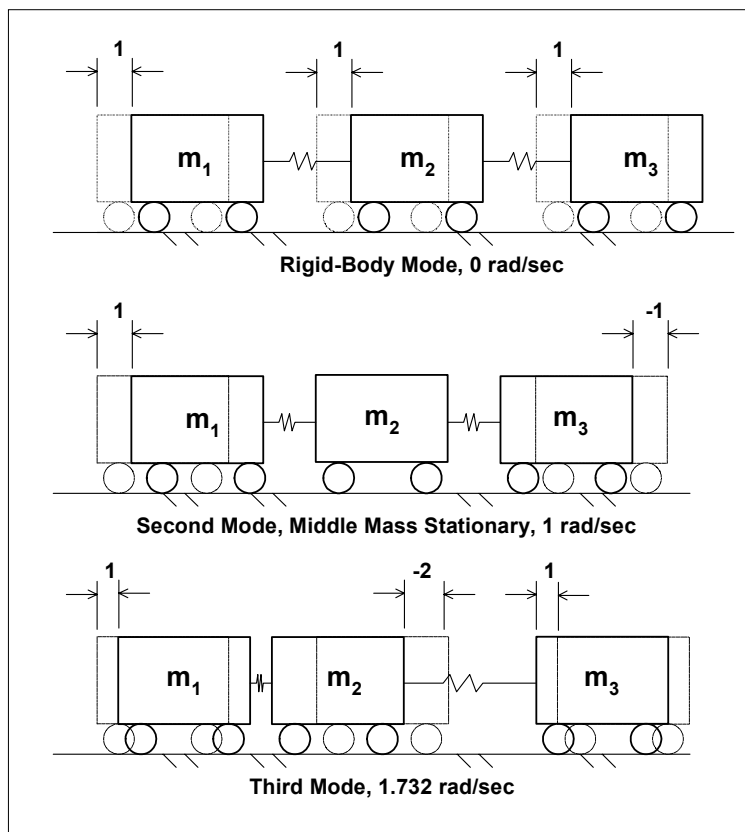


Figure 10.1: Mode shape plots, “real” modes.

For “real” eigenvectors, there are two ways of visualizing the mode shapes and resulting motions. One method we have used several times before, the mode shape plot, shows the deformed shapes of the system for each eigenvector.

Since for real eigenvectors all the degrees of freedom reach their maxima and minima at the same times, any snapshot in time will show the relative displacements, which is why we can plot a deformed mode plot as shown in [Figure 10.1](#).

10.7.2 Principal Coordinate Eigenvalue Problem

The section of code below prompts for the amount of proportional damping, zeta, and then sets up the equations of motion in principal coordinates. After solving the eigenvalue problem, the eigenvalues and eigenvectors are sorted and the magnitude and phase angle of the each eigenvector is defined.

```
% input proportional damping for equations in principal coordinate system

zeta = input('input value for zeta, default = 0.02, 2% of critical ... ');

if (isempty(zeta))
    zeta = 0.02;
else
    end

% setup proportionally damped state-space system matrix in principal coordinates

a_ss = [ 0      1      0      0      0      0
          0      0      0      0      0      0
          0      0      0      1      0      0
          0      0     -w2^2   -2*zeta*w2  0      0
          0      0      0      0      0      1
          0      0      0      0     -w3^2   -2*zeta*w3];

% solve for the eigenvalues of the system matrix with proportional damping

[xmp,omegap] = eig(a_ss);

% take the diagonal elements of the generalized eigenvalue matrix omegap

omegapd = diag(omegap);

% now reorder the eigenvalues and eigenvectors from low to high frequency,
% keeping track of how the eigenvalues are ordered in reorder the
% eigenvectors to match, using indexhz

[omegaporder,indexhz] = sort(abs(imag(omegapd)));

for cnt = 1:length(omegapd)

    omegapo(cnt,1) = omegapd(indexhz(cnt)); % reorder eigenvalues

    xmpo(:,cnt) = xmp(:,indexhz(cnt)); % reorder eigenvector columns

end

% now calculate the magnitude and phase angle of each of the eigenvector
% entries

for row = 1:length(omegapd)
```

```

        for col = 1:length(omegapd)

            xmpomag(row,col) = abs(xmpo(row,col));

            xmpoang(row,col) = (180/pi)*angle(xmpo(row,col));

        end

    end

    omegapo

    xmpo

    xmpomag

    xmpoang

```

10.7.3 Damping Calculation, Eigenvalue Complex Plane Plot

The section below calculates the percentage of critical damping due to the defined amount of input damping, zeta. For example, if 2% of critical damping is defined as input, then we should see that the eigenvalues of the equations of motion in principal coordinates plot as shown in [Figure 5.2](#).

```

% calculate the percentage of critical damping for each mode

zeta1 = 0

theta2 = atan(real(omegapo(3))/imag(omegapo(3)));
zeta2 = abs(sin(theta2))

theta3 = atan(real(omegapo(5))/imag(omegapo(5)));
zeta3 = abs(sin(theta3))

plot(omegap,'kx')
grid on
axis([-3 1 -2 2])
axis('square')
title('Proportionally Damped Eigenvalues')
xlabel('real')
ylabel('imaginary')
text(real(omegapo(3))-1,imag(omegapo(3))+0.1,['zeta = ',num2str(zeta2)])
text(real(omegapo(5))-1,imag(omegapo(5))+0.1,['zeta = ',num2str(zeta3)])

disp('execution paused to display figure, "enter" to continue'); pause

```

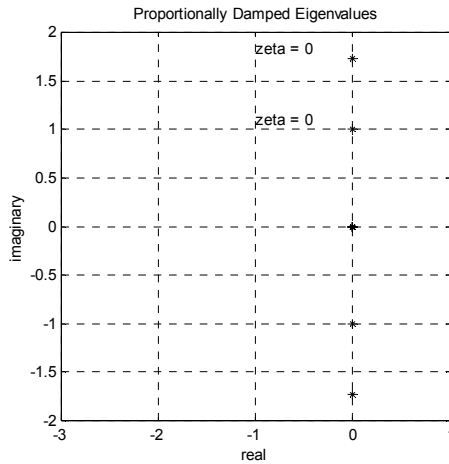


Figure 10.2: Undamped model eigenvalue plot in complex plane.

For the undamped model, we should see that the eigenvalues, poles, should lie on the imaginary axis – and they do.

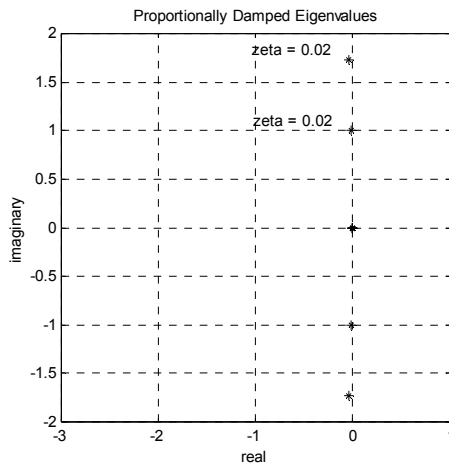


Figure 10.3: Proportionally damped eigenvalue plot, zeta = 2% was input.

The eigenvalues for zeta = 0.02 plot slightly to the left of the imaginary axis.

10.7.4 Principal Displacement Calculations

We showed in (5.54), repeated below, how to calculate the displacements when the system is started with a set of initial conditions which match the eigenvector:

$$\begin{aligned}\mathbf{x}(t) &= e^{\sigma_{n1}t} (e^{j\omega_{n1}t} \mathbf{x}_{n1} + e^{-j\omega_{n2}t} \mathbf{x}_{n2}) \\ &= e^{\sigma_{n1}t} (e^{j\omega_{n1}t} \mathbf{x}_{n1}) + e^{\sigma_{n1}t} (e^{-j\omega_{n2}t} \mathbf{x}_{n2})\end{aligned}\quad (10.52)$$

Since our eigenvalues lie along the imaginary axis, their σ values are zero and $e^{0t} = 1$, the equations can be simplified to:

$$\mathbf{x}(t) = e^{j\omega_{n1}t} \mathbf{x}_{n1} + e^{-j\omega_{n2}t} \mathbf{x}_{n2} \quad (10.53)$$

A time vector from 0 to 15 seconds is defined, and real and imaginary parts are picked from the eigenvalues. Equation (10.52) is used to calculate the motions.

```
% calculate the motions of the three masses for all three modes - damped case

t = 0:12:15;

sigma11 = real(omegapo(1)); % sigma for first eigenvalue for mode 1
omegap11 = imag(omegapo(1)); % omegap for first eigenvalue for mode 1

sigma12 = real(omegapo(2)); % sigma for second eigenvalue for mode 1
omegap12 = imag(omegapo(2)); % omegap for second eigenvalue for mode 1

sigma21 = real(omegapo(3)); % sigma for first eigenvalue for mode 2
omegap21 = imag(omegapo(3)); % omegap for first eigenvalue for mode 2

sigma22 = real(omegapo(4)); % sigma for second eigenvalue for mode 2
omegap22 = imag(omegapo(4)); % omegap for second eigenvalue for mode 2
sigma31 = real(omegapo(5)); % sigma for first eigenvalue for mode 3
omegap31 = imag(omegapo(5)); % omegap for first eigenvalue for mode 3

sigma32 = real(omegapo(6)); % sigma for second eigenvalue for mode 3
omegap32 = imag(omegapo(6)); % omegap for second eigenvalue for mode 3
% displacements of mode 1 in principal coordinates

zp111 = exp(sigma11*t).*(exp(i*omegap11*t)*xmpo(1,1)); % mass 1
zp112 = exp(sigma12*t).*(exp(i*omegap12*t)*xmpo(1,2)); % mass 1

% displacements of mode 2 in principal coordinates

zp221 = exp(sigma21*t).*(exp(i*omegap21*t)*xmpo(3,3)); % mass 2
```

```

    zp222 = exp(sigma22*t).*(exp(i*omegap22*t)*xmpo(3,4)); % mass 2

% displacements of mode 3 in principal coordinates

    zp331 = exp(sigma31*t).*(exp(i*omegap31*t)*xmpo(5,5)); % mass 3
    zp332 = exp(sigma32*t).*(exp(i*omegap32*t)*xmpo(5,6)); % mass 3

```

10.7.5 Transformation to Physical Coordinates

The section of code below sets up the appropriate size matrices to enable back-transforming from principal to physical coordinates.

```

% calculate the motions of each mass for mode 2
% define matrix of displacements vs time for each eigenvector

    z221 = [zeros(1,length(t))
            zp221
            zeros(1,length(t))];

    z222 = [zeros(1,length(t))
            zp222
            zeros(1,length(t))];

% back-transform from principal to physical coordinates

    zmode21 = xn*z221;

    zmode22 = xn*z222;

    z1mode21 = zmode21(1,:);

    z2mode21 = zmode21(2,:);

    z3mode21 = zmode21(3,:);

    z1mode22 = zmode22(1,:);

    z2mode22 = zmode22(2,:);

    z3mode22 = zmode22(3,:);

% calculate the motions of each mass for mode 3
% define matrix of displacements vs time for each eigenvector

    z331 = [zeros(1,length(t))
            zeros(1,length(t))
            zp331];

    z332 = [zeros(1,length(t))
            zeros(1,length(t))
            zp332];

```

```

zmode31 = xn*z331;
zmode32 = xn*z332;

z1mode31 = zmode31(1,:);
z2mode31 = zmode31(2,:);
z3mode31 = zmode31(3,:);

z1mode32 = zmode32(1,:);
z2mode32 = zmode32(2,:);
z3mode32 = zmode32(3,:);

```

10.7.6 Plotting Results

The plotting commands for mode 2 are listed below; those for mode 3 have been eliminated for brevity.

```

% plot principal displacements of mode 2

plot(t,real(zp221),'k-',t,real(zp222),'k+-',t,imag(zp221),'k-',t,imag(zp222),'ko-')
title('principal real and imag disp for mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -1 1])
grid on

disp('execution paused to display figure, "enter" to continue'); pause

% plot physical disp of masses for mode 2

plot(t,real(z1mode21),'k-',t,real(z1mode22),'k+-',t,imag(z1mode21), ...
      'k-',t,imag(z1mode22),'ko-')
title('physical real and imag disp for mass 1, mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -0.5 0.5])
grid on

disp('execution paused to display figure, "enter" to continue'); pause

plot(t,real(z2mode21),'k-',t,real(z2mode22),'k+-',t,imag(z2mode21), ...
      'k-',t,imag(z2mode22),'ko-')
title('physical real and imag disp for mass 2, mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -0.5 0.5])
grid on

disp('execution paused to display figure, "enter" to continue'); pause

plot(t,real(z3mode21),'k-',t,real(z3mode22),'k+-',t,imag(z3mode21), ...

```

```

                                'k.-',t,imag(z3mode22),'ko-')
title('physical real and imag disp for mass 3, mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -0.5 0.5])
grid on

disp('execution paused to display figure, "enter" to continue'); pause

plot(t,real(z1mode21+z1mode22),'k-',t,real(z2mode21+z2mode22), ...
      'k+-',t,real(z3mode21+z3mode22),'k.-')
title('physical disp z1, z2, z3 mode 2')
legend('mass 1','mass 2','mass 3')
axis([0 max(t) -1 1])
grid on

disp('execution paused to display figure, "enter" to continue'); pause

% plot subplots for notes

% plot principal disp of mode 2

subplot(3,2,1)
plot(t,real(zp221),'k-',t,real(zp222),'k+-',t,imag(zp221),'k.-',t,imag(zp222),'ko-')
title('principal disp for mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -1 1])
grid on

% plot physical disp of masses for mode 2

subplot(3,2,3)
plot(t,real(z1mode21),'k-',t,real(z1mode22),'k+-',t,imag(z1mode21), ...
      'k.-',t,imag(z1mode22),'ko-')
title('physical real and imag disp for mass 1, mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -0.5 0.5])
grid on

subplot(3,2,4)
plot(t,real(z2mode21),'k-',t,real(z2mode22),'k+-',t,imag(z2mode21), ...
      'k.-',t,imag(z2mode22),'ko-')
title('physical real and imag disp for mass 2, mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -0.5 0.5])
grid on

subplot(3,2,5)
plot(t,real(z3mode21),'k-',t,real(z3mode22),'k+-',t,imag(z3mode21), ...
      'k.-',t,imag(z3mode22),'ko-')
title('physical real and imag disp for mass 3, mode 2')
legend('real','real','imag','imag')
axis([0 max(t) -0.5 0.5])
grid on

subplot(3,2,6)

```

```

plot(t,real(z1mode21+z1mode22),'k+-',t,real(z2mode21+z2mode22), ...
      'k.-',t,real(z3mode21+z3mode22),'ko-')
title('physical disp for z1, z2, z3 mode 2')
legend('mass 1','mass 2','mass 3')
axis([0 max(t)-1 1])
grid on

disp('execution paused to display figure, "enter" to continue'); pause

subplot(1,1,1)

```

10.7.7 Undamped / Proportionally Damped Argand Diagram, Mode 2

As in the Argand diagrams explained in Chapter 5, the two complex conjugate eigenvectors for each mode are plotted side by side. The direction of the rotation of the eigenvector is indicated by the arrow associated with the $e^{j\omega t}$ or $e^{-j\omega t}$ terms. The addition of the two counter-rotating complex eigenvectors for an arbitrary time “t” is shown in the middle and below the two individual eigenvector plots for each dof. The addition plot shows how the two imaginary components cancel each other out, leaving only the real portion of the motion.

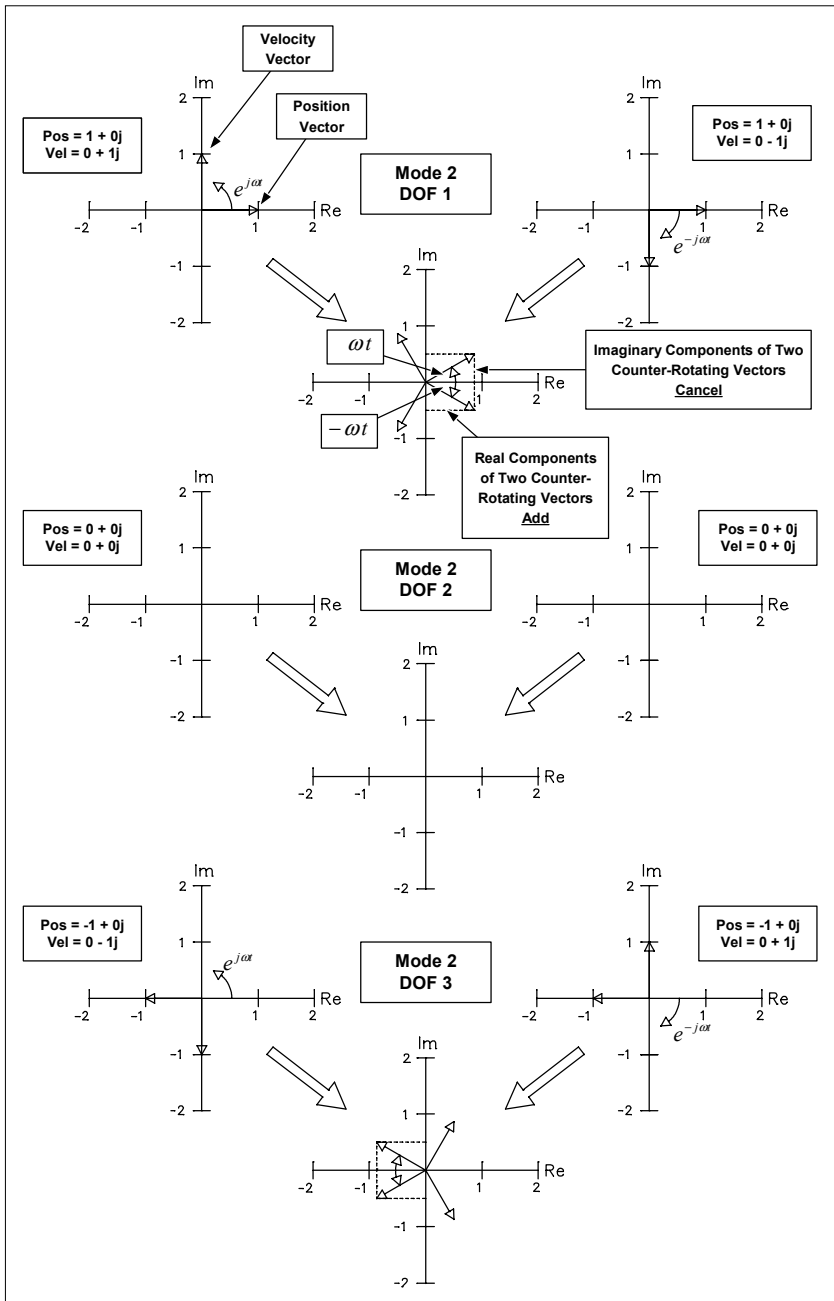


Figure 10.4: Argand diagram for undamped or proportionally damped system, mode 2.

10.7.8 Undamped / Proportionally Damped Argand Diagram, Mode 3

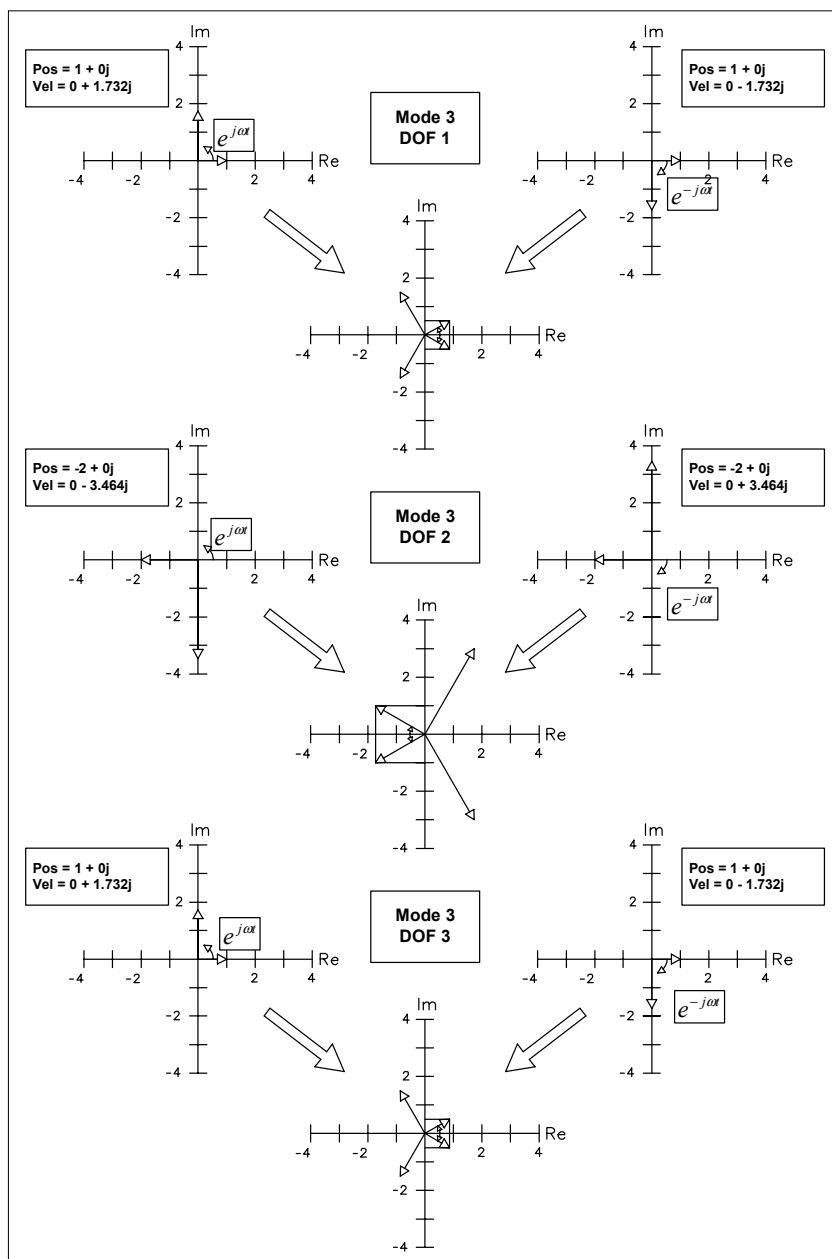


Figure 10.5: Argand diagram for undamped or proportionally damped system, mode 3.

10.7.9 Proportionally Damped Initial Condition Response, Mode 2

Figures 10.6 to 10.10 show the initial condition responses for mode 2 for proportional damping of 2%. Mode 2 is the mode where mass 2 is stationary and masses 1 and 3 are moving out of phase with each other with equal amplitude.

Figure 10.6 shows the real and imaginary components of the two complex eigenvector responses that make up mode 2. Note that the two imaginary components are out of phase and cancel each other while the two real components are overlaid and will add. Figures 10.7 to 10.9 show the real and imaginary components for each of the three masses. The motions of mass 2 are zero, while the motions of masses 1 and 3 are out of phase with each other, consistent with the shape of mode 2 in Figure 10.1. Figure 10.10 shows the physical displacements of the three masses versus time. The Argand diagram vectors for mode 2, Figure 10.4, can be matched with each figure for each degree of freedom.

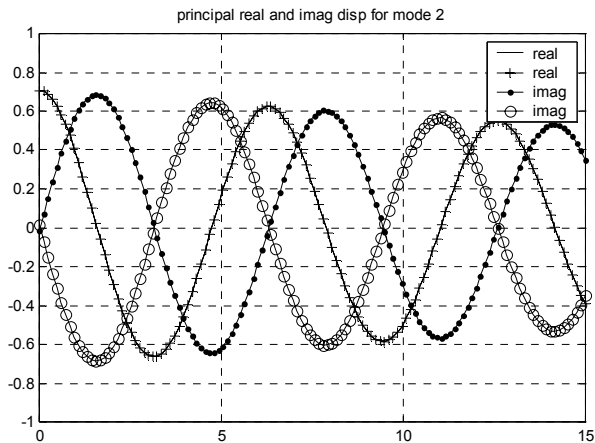


Figure 10.6: Principal real and imaginary displacements, mode 2.

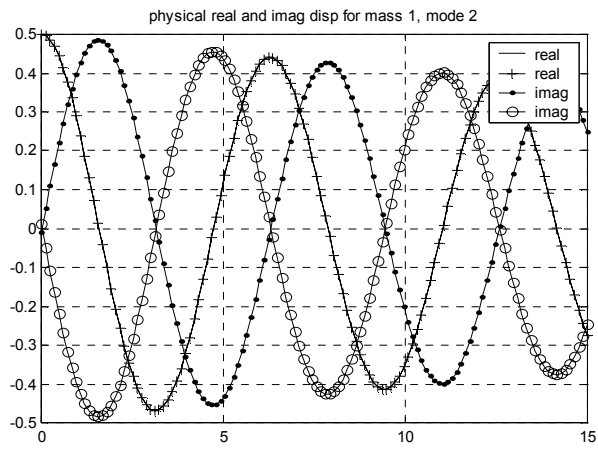


Figure 10.7: Physical real and imaginary displacements for mass 1, mode 2.

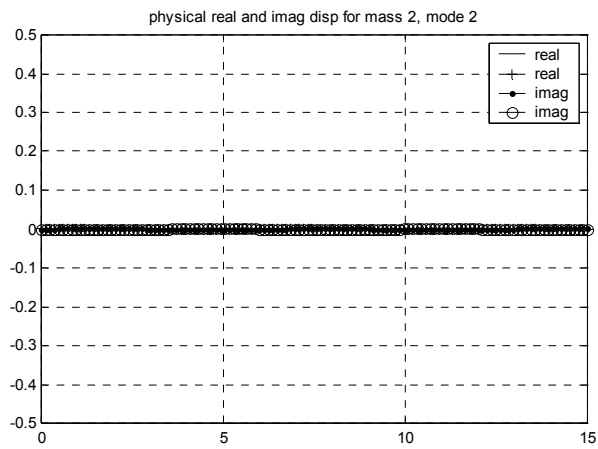


Figure 10.8: Physical real and imaginary displacements for mass 2, mode 2.

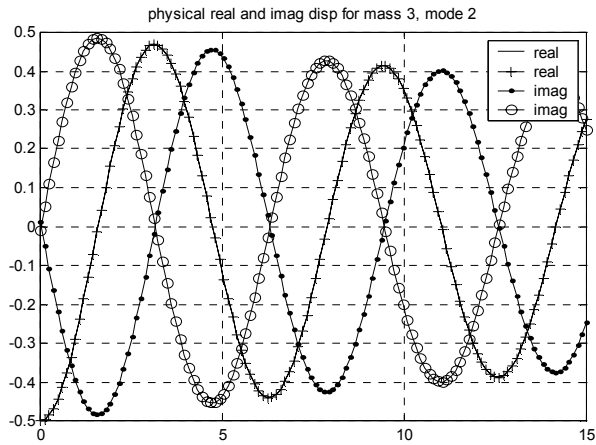


Figure 10.9: Physical real and imaginary displacements for mass 3, mode 2.

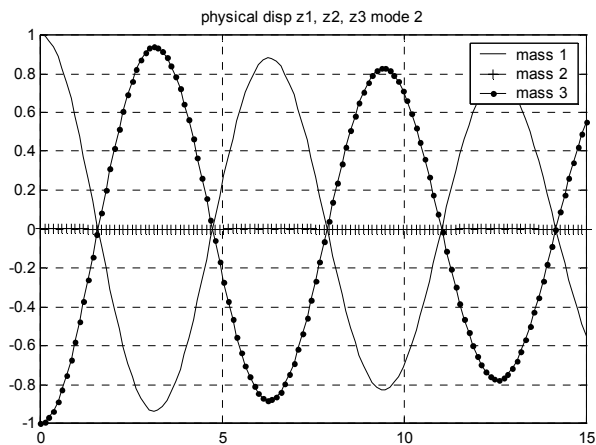


Figure 10.10: Physical displacements for masses 1, 2 and 3, mode 2.

10.7.10 Proportionally Damped Initial Condition Response, Mode 3

Figures 10.11 to 10.15 show the initial condition responses for mode 3 for 2% proportional damping, where mass 2 moves twice as far and out of phase with masses 1 and 3.

Figure 10.11 shows the real and imaginary components of the two complex eigenvector responses that make up mode 2. As in the previous section, note

that the two imaginary components are out of phase and cancel each other while the two real components are overlaid and will add. Figures 10.12 to 10.14 display the real and imaginary components for each of the three masses. Figure 10.15 shows the physical displacements of the three masses versus time. The Argand diagram vectors for mode 2, Figure 10.5, can be matched with each figure for each degree of freedom.

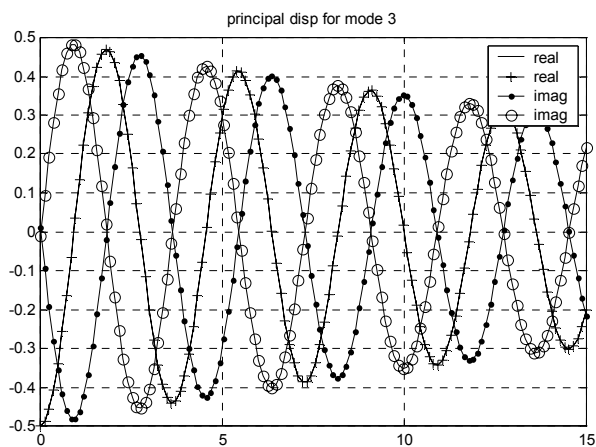


Figure 10.11: Principal real and imaginary displacements, mode 3.

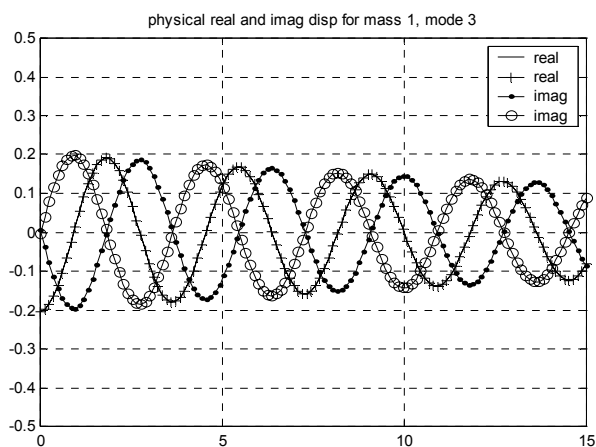


Figure 10.12: Physical real and imaginary displacements for mass 1, mode 3.

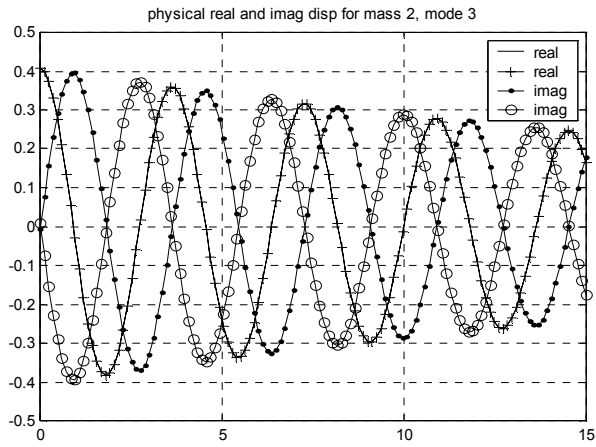


Figure 10.13: Physical real and imaginary displacements for mass 2, mode 3.

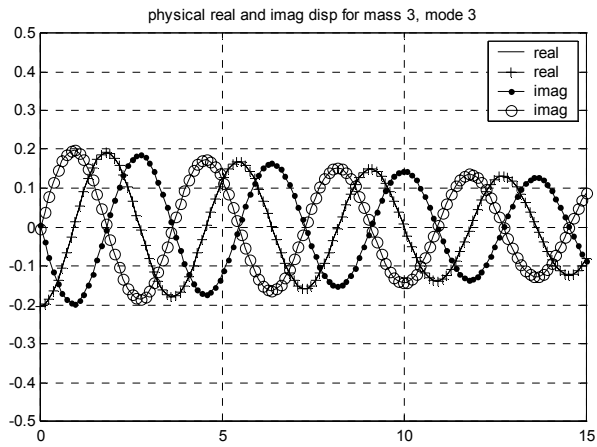


Figure 10.14: Physical real and imaginary displacements for mass 3, mode 3.

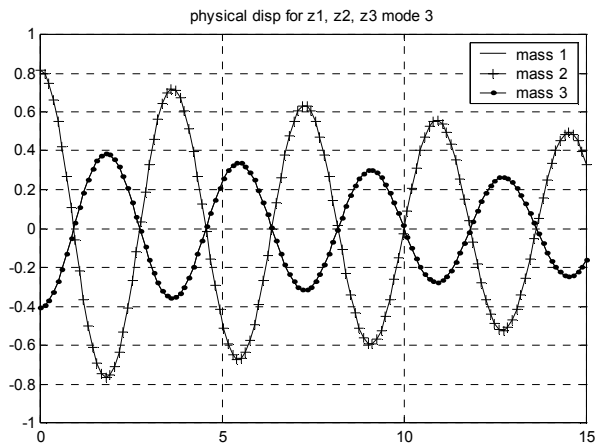


Figure 10.15: Physical real and imaginary displacements for masses 1,2 and 3, mode 3.

Problems

Note: All the problems refer to the two dof system shown in [Figure P2.2](#).

P10.1 Write the homogeneous equations of motion in state space form for the undamped two dof system with $m_1 = m_2 = m = 1$, $k_1 = k_2 = k = 1$. Set up the eigenvalue problem and expand the determinant to reveal the characteristic equation. Compare with the denominator terms from P2.2.

P10.2 Solve for the eigenvalues and eigenvectors in state space form. Compare with the results from P7.1. What is the relationship between the displacement and velocity eigenvector terms?

P10.3 (MATLAB) Modify the `tdofss_eig.m` code for the undamped two dof system with $m_1 = m_2 = m = 1$, $k_1 = k_2 = k = 1$. Print out the eigenvalue and eigenvector results and compare with the results from P10.2. What changes are required to the MATLAB eigenvectors to make them match the P10.2 results? After normalizing with respect to mass, confirm that the equations of motion consist of an identity mass matrix and a stiffness matrix with squares of the eigenvalues along the diagonal.

P10.4 Write the equations of motion in principal coordinates in state space form, knowing only the eigenvalues and eigenvectors, similar to (10.35). Use the displacements of mass 1 and mass 2 as outputs. Show how the output matrix C can be formulated to only require a single multiplication to give

outputs (Section 10.6.7.2). Identify the 2x2 submatrices which define the state equations of each mode. Are the individual modes uncoupled?

P10.5 (MATLAB) Modify the **tdof_prop_damped.m** code for the two dof system with $m_1 = m_2 = m = 1$, $k_1 = k_2 = k = 1$. Plot the eigenvalue locations in the s-plane for zero damping and for proportional damping of 2% (0.02). List the eigenvalues and eigenvectors for the undamped and proportional damping cases and note the differences. Plot the initial condition responses when started in initial conditions which match each of the two eigenvectors.

P10.6 Plot Argand diagrams for the undamped system.