CHAPTER 14

FINITE ELEMENTS: DYNAMICS

14.1 Introduction

The chapter starts out with discussions of various mass matrix formulations. The 6dof lumped mass example from Chapter 2 is used for the lumped mass matrix example. A two-element cantilever is used to develop the consistent mass example. Using the same technique as in the previous chapter, the global mass matrix is built up as an assemblage of element mass matrices. A method analogous to static condensation, Guyan reduction, is developed and used to reduce the size of the two-element cantilever problem. The cantilever is then solved for its eigenvalues by hand using Guyan reduction. The same cantilever is solved for eigenvalues and eigenvectors using MATLAB and results are compared to the hand calculations.

Following the two-element cantilever example, a second MATLAB code allows solving for eigenvalues and eigenvectors for a uniform cantilever beam with user-defined number of elements. The results of the MATLAB code are compared with the results from an ANSYS model for the same 10-element cantilever.

This 10-element cantilever will be the last eigenvalue analysis in the book using MATLAB. Further chapters will start with eigenvalue results from ANSYS models, which will be used to build state space MATLAB models. These MATLAB models are then used for frequency and time domain analyses. This chapter serves as a bridge between carrying out analyses completely in MATLAB and using ANSYS results as the starting point for state space MATLAB models. Hence, we will reintroduce ANSYS eigenvalue/eigenvector results and start becoming familiar with their form and interpretation.

14.2 Six dof Global Mass Matrix

The lumped mass matrix is simple to construct because there is only a single degree of freedom associated with each mass element. This leads to the 6x6 diagonal mass matrix below, which can be constructed in the same manner as the 6dof stiffness matrix in the previous chapter.

$$\mathbf{m}_g = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_6 \end{bmatrix} \tag{14.1}$$

14.3 Cantilever Dynamics

14.3.1 Overview – Mass Matrix Forms

In order to solve for the dynamics of the cantilever beam, we need to develop a mass matrix to complete the equations of motion. For a beam finite element, there are a number of different mass matrix formulations, each of which will be covered below:

1) Lumped mass, displacements only

2) Lumped mass, displacements and rotations both included

3) Consistent mass – distributed mass effect

14.3.2 Lumped Mass

Beam-element lumped parameter mass and inertia terms in the mass matrix relate point inertial loads to point accelerations and give only diagonal terms. Equation (14.2) below shows the lumped mass matrix including both displacements and rotations:

$$\mathbf{m}_l = \begin{bmatrix} \left(\frac{ml}{2}\right) & 0 & 0 & 0 \\ 0 & \left(\frac{ml^3}{24} + \frac{mlI_y}{2A}\right) & 0 & 0 \\ 0 & 0 & \left(\frac{ml}{2}\right) & 0 \\ 0 & 0 & 0 & \left(\frac{ml^3}{24} + \frac{mlI_y}{2A}\right) \end{bmatrix} \tag{14.2}$$

For the lumped mass for displacement terms only, the (2,2) and (4,4) terms in (14.2) would be set to zero. Notation is as follows: m is mass per unit length,

l is the element length, I_y is the cross-sectional moment of inertia about the y axis and A is the cross section area. This lumped mass formulation assumes a prismatic beam (same area and moment of inertia along the length) and effectively lumps half of the mass and inertia at each end (Archer 1963).

14.3.3 Consistent Mass

Lumped mass formulations were state of the art in structural dynamics until Archer's classic paper introduced the consistent mass matrix in 1963.

We will see in the development below that the consistent mass matrix for a beam element is a filled matrix. The filled matrix can be combined with other consistent mass matrices of other elements of the structure, in the same manner as the element stiffness matrices are combined, to yield the final global mass matrix.

The element consistent mass matrix for a prismatic beam is, with mass per unit length m and length l (Weaver 1990):

$$\mathbf{m}_e = \frac{ml}{420}\begin{bmatrix} 156 & 22l & 54 & -13l \\ 22l & 4l^2 & 13l & -3l^2 \\ 54 & 13l & 156 & -22l \\ -13l & -3l^2 & -22l & 4l^2 \end{bmatrix} \tag{14.3}$$

Figure 14.1 shows the unit accelerations of each of the four degrees of freedom which correspond to the four columns of the consistent mass matrix, analogous to the beam element stiffness description in Chapter 13.

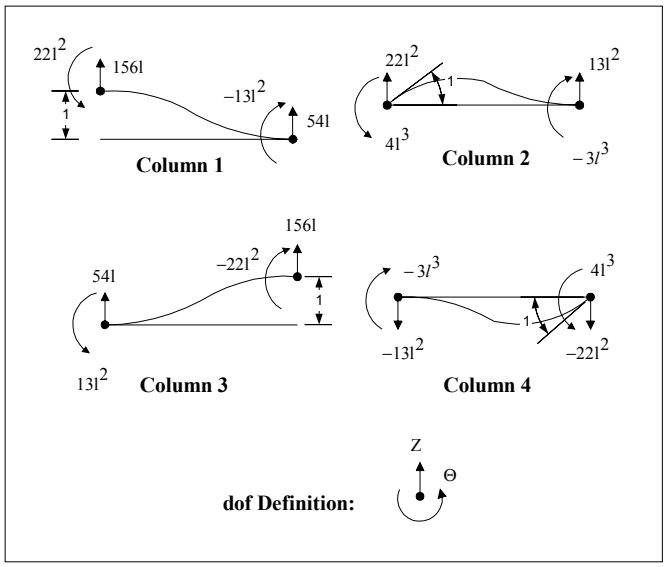

Figure 14.1: Beam element consistent mass matrix terms.

14.4 Dynamics of Two-Element Cantilever – Consistent Mass Matrix

We already have the global stiffness matrix for the two-element cantilever beam from (13.26):

$$\mathbf{k}_g = EI \begin{bmatrix} \frac{24}{l^3} & 0 & \frac{-12}{l^3} & \frac{6}{l^2} \\ 0 & \frac{8}{l} & \frac{-6}{l^2} & \frac{2}{l} \\ \frac{-12}{l^3} & \frac{-6}{l^2} & \frac{12}{l^3} & \frac{-6}{l^2} \\ \frac{6}{l^2} & \frac{2}{l} & \frac{-6}{l^2} & \frac{4}{l} \end{bmatrix} \tag{14.4}$$

The global mass matrix (using consistent mass) can also be built by combining the terms from each of the beam elements as follows:

$$\mathbf{m}_g = \frac{1}{420}\begin{bmatrix} 156m_1l_1 & 22m_1l_1^2 & 54m_1l_1 \\ 22m_1l_1^2 & 4m_1l_1^3 & 13m_1l_1^2 \\ 54m_1l_1 & 13m_1l_1^2 & (156m_1l_1 + 156m_2l_2) \\ -13m_1l_1^2 & -3m_1l_1^3 & (-22m_1l_1^2 + 22m_2l_2^2) \\ 0 & 0 & 54m_2l_2 \\ 0 & 0 & -13m_2l_2^2 \end{bmatrix}$$

$$\begin{bmatrix} -13m_1l_1^2 & 0 & 0 \\ -3m_1l_1^3 & 0 & 0 \\ (-22m_1l_1^2 + 22m_2l_2^2) & 54m_2l_2 & -13m_2l_2^2 \\ (4m_1l_1^3 + 4m_2l_2^3) & 13m_2l_2^2 & -3m_2l_2^3 \\ 13m_2l_2^2 & 156m_2l_2 & -22m_2l_2^2 \\ -3m_2l_2^3 & -22m_2l_2^2 & 4m_2l_2^3 \end{bmatrix} \tag{14.5}$$

Once again, assuming the two elements have the same properties and lengths, the global mass matrix becomes:

$$\mathbf{m}_g = \frac{1}{420}\begin{bmatrix} 156ml & 22ml^2 & 54ml & -13ml^2 & 0 & 0 \\ 22ml^2 & 4ml^3 & 13ml^2 & -3ml^3 & 0 & 0 \\ 54ml & 13ml^2 & 312ml & 0 & 54ml & -13ml^2 \\ -13ml^2 & -3ml^3 & 0 & 8ml^3 & 13ml^2 & -3ml^3 \\ 0 & 0 & 54ml & 13ml^2 & 156ml & -22ml^2 \\ 0 & 0 & -13ml^2 & -3ml^3 & -22ml^2 & 4ml^3 \end{bmatrix} \tag{14.6}$$

Taking into account the two constrained degrees of freedom at the built in end, we can eliminate the first two rows and columns:

$$\mathbf{m}_g = \frac{1}{420}\begin{bmatrix} 312ml & 0 & 54ml & -13ml^2 \\ 0 & 8ml^3 & 13ml^2 & -3ml^3 \\ 54ml & 13ml^2 & 156ml & -22ml^2 \\ -13ml^2 & -3ml^3 & -22ml^2 & 4ml^3 \end{bmatrix} \tag{14.7}$$

Having the mass and stiffness matrices allows us to solve the eigenvalue problem for the homogeneous equations of motion:

$$\mathbf{m}_g\ddot{\mathbf{z}} + \mathbf{k}_g\mathbf{z} = [0] \tag{14.8}$$

In order to solve the problem by hand, we will need to find several inverses, so we will again see if we can cut the 4x4 problem down to 2x2 size. We will now use Guyan reduction to reduce the size of the problem.

14.5 Guyan Reduction

Guyan reduction is a method of decreasing the number of degrees of freedom in a dynamics problem, similar to the process of static condensation in a statics problem. Unlike static condensation, however, Guyan reduction introduces errors due to the approximations made. The magnitude of the errors introduced depends upon the choice of degrees of freedom to be reduced, the dependent or slave degrees of freedom. The most popular choice of degrees of freedom to be reduced are translations of nodes with relatively lower masses and rotations of nodes with relatively lower mass moment of inertia. This leaves translations of relatively larger mass nodes and rotations of relatively larger mass moment of inertia nodes as the independent degrees of freedom. In a typical finite element problem, the analyst will define masters as degrees of freedom where forces/moment are applied, where displacements or rotations are required for output, or where known large masses/mass moments of inertia occur. The finite element program will then be allowed to choose an additional set of degrees of freedom and add them to the master set. Typically the program sorts along the diagonal of the mass matrix, adding degrees of freedom associated with the larger terms.

14.5.1 Guyan Reduction Derivation

Starting with the undamped equations of motion:

$$\mathbf{m}\ddot{\mathbf{z}} + \mathbf{k}\mathbf{z} = [0] \tag{14.9}$$

Rearranging and partitioning into displacements to be reduced, $\mathbf{z}_a$, and independent displacements, $\mathbf{z}_b$:

$$\begin{bmatrix} \mathbf{m}_{aa} & \mathbf{m}_{ab} \\ \mathbf{m}_{ba} & \mathbf{m}_{bb} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{z}}_a \\ \ddot{\mathbf{z}}_b \end{bmatrix} + \begin{bmatrix} \mathbf{k}_{aa} & \mathbf{k}_{ab} \\ \mathbf{k}_{ba} & \mathbf{k}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{z}_a \\ \mathbf{z}_b \end{bmatrix} = \begin{bmatrix} \mathbf{F}_a \\ \mathbf{F}_a \end{bmatrix} \tag{14.10}$$

Multiplying out the first matrix equation:

$$\mathbf{m}_{aa}\ddot{\mathbf{z}}_a + \mathbf{m}_{ab}\ddot{\mathbf{z}}_b + \mathbf{k}_{aa}\mathbf{z}_a + \mathbf{k}_{ab}\mathbf{z}_b = \mathbf{F}_a \tag{14.11}$$

Solving the above for $\mathbf{z}_a$:

$$\begin{aligned}\mathbf{z}_a &= \mathbf{k}_{aa}^{-1}\left(\mathbf{F}_a - \mathbf{k}_{ab}\mathbf{z}_b - \mathbf{m}_{aa}\ddot{\mathbf{z}}_a - \mathbf{m}_{ab}\ddot{\mathbf{z}}_b\right) \\ &= -\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab}\mathbf{z}_b + \mathbf{k}_{aa}^{-1}\left(\mathbf{F}_a - \mathbf{m}_{aa}\ddot{\mathbf{z}}_a - \mathbf{m}_{ab}\ddot{\mathbf{z}}_b\right)\end{aligned} \tag{14.12}$$

Instead of letting $\mathbf{z}_a$ depend upon the entire right-hand side of (14.13), the approximation of static equilibrium is introduced:

$$\mathbf{z}_a = -\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab}\mathbf{z}_b \tag{14.13}$$

Typically the choice of degrees of freedom to be reduced does not include any degrees of freedom to which forces are applied, thus $\mathbf{F}_a = 0$. The static equilibrium approximation basically sets the term in brackets in (14.12) to zero. Setting $\mathbf{F}_a = 0$ and using the second derivative of (14.13), we can see the form of $\mathbf{m}_{ab}$:

$$\begin{aligned}0 &= \mathbf{F}_a - \mathbf{m}_{aa}\ddot{\mathbf{z}}_a - \mathbf{m}_{ab}\ddot{\mathbf{z}}_b \\ &= -\mathbf{m}_{aa}\ddot{\mathbf{z}}_a - \mathbf{m}_{ab}\ddot{\mathbf{z}}_b \\ &= -\mathbf{m}_{aa}(-\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab}\ddot{\mathbf{z}}_b) - \mathbf{m}_{ab}\ddot{\mathbf{z}}_b \\ &= \mathbf{m}_{aa}\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab} - \mathbf{m}_{ab}\end{aligned} \tag{14.14a,b}$$

$$\mathbf{m}_{ab} = \mathbf{m}_{aa}\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab}$$

We assume that the $\mathbf{m}_{aa}\ddot{\mathbf{z}}_a$ terms are zero and that $\mathbf{m}_{aa}$ and $\mathbf{m}_{ab}$ are related as in (14.14b). The force transmission between the $\ddot{\mathbf{z}}_a$ and $\ddot{\mathbf{z}}_b$ degrees of freedom is related only to the stiffnesses as denoted in (14.14), hence the "static equilibrium" approximation.

Assuming (14.13) holds, the displacement vector $\mathbf{z}$ can be written in terms of $\mathbf{z}_b$ only:

$$\mathbf{z} = \begin{bmatrix}\mathbf{z}_a \\ \mathbf{z}_b\end{bmatrix} = \begin{bmatrix}-\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab} \\ \mathbf{I}\end{bmatrix}\mathbf{z}_b = \begin{bmatrix}\mathbf{T}_{ab} \\ \mathbf{I}\end{bmatrix}\mathbf{z}_b = \mathbf{T}\mathbf{z}_b \tag{14.15}$$

where:

$$\mathbf{T}_{ab} = -\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab} \tag{14.16}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{ab} \\ \mathbf{I} \end{bmatrix} \tag{14.17}$$

Substitution of (14.14), with derivatives, into (14.9) yields:

$$\mathbf{mT}\ddot{\mathbf{z}}_b + \mathbf{kT}\mathbf{k}_b = \mathbf{F} \tag{14.18}$$

Equation (14.18) still contains (a + b) degrees of freedom, so premultiplication by $\mathbf{T}^T$ is required to reduce to (b) degrees of freedom and to return symmetry to the reduced mass and stiffness matrices:

$$\left(\mathbf{T}^T\mathbf{mT}\right)\ddot{\mathbf{z}}_b + \left(\mathbf{T}^T\mathbf{kT}\right)\mathbf{z}_b = \mathbf{T}^T\mathbf{F} \tag{14.19}$$

Rewriting in a more compact form:

$$\mathbf{m}^*_{bb}\ddot{\mathbf{z}}_b + \mathbf{k}^*_{bb}\mathbf{z}_b = \mathbf{F}^*_b \tag{14.20}$$

Equation (14.20) is the final reduced equation of motion which can be solved for the displacements of type b. Displacements of type a (assuming static equilibrium) can then be solved for using (14.13).

$\mathbf{k}^*_{bb}$ can be shown to be the same as that derived in the static condensation Section 13.4.1, (13.39):

$$\begin{aligned}
\mathbf{k}^*_{bb} &= \begin{bmatrix} \mathbf{T}^T_{ab} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{k}_{aa} & \mathbf{k}_{ab} \\ \mathbf{k}_{ba} & \mathbf{k}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{T}_{ab} \\ \mathbf{I} \end{bmatrix} \\
&= \begin{bmatrix} \left(\mathbf{T}^T_{ab}\mathbf{k}_{aa} + \mathbf{k}_{ba}\right) & \left(\mathbf{T}^T_{ab}\mathbf{k}_{ab} + \mathbf{k}_{bb}\right) \end{bmatrix} \begin{bmatrix} \mathbf{T}_{ab} \\ \mathbf{I} \end{bmatrix} \\
&= \mathbf{T}^T_{ab}\mathbf{k}_{aa}\mathbf{T}_{ab} + \mathbf{k}_{ba}\mathbf{T}_{ab} + \mathbf{T}^T_{ab}\mathbf{k}_{ab} + \mathbf{k}_{bb} \\
&= \mathbf{k}_{ba}\mathbf{k}^{-1}_{aa}\mathbf{k}_{aa}\mathbf{k}^{-1}_{aa}\mathbf{k}_{ab} - \mathbf{k}_{ba}\mathbf{k}^{-1}_{aa}\mathbf{k}_{ab} - \mathbf{k}_{ba}\mathbf{k}^{-1}_{aa}\mathbf{k}_{ab} + \mathbf{k}_{bb} \\
&= \mathbf{k}_{bb} - \mathbf{k}_{ba}\mathbf{k}^{-1}_{aa}\mathbf{k}_{ab}
\end{aligned} \tag{14.21}$$

14.5.2 Two-Element Cantilever Eigenvalues Closed Form Solution Using Guyan Reduction

Repeating the rearranged global stiffness matrix from the static run, (13.45):

$$\mathbf{k}_g = EI\begin{bmatrix} \frac{8}{l} & \frac{2}{l} & 0 & \frac{-6}{l^2} \\ \frac{2}{l} & \frac{4}{l} & \frac{6}{l^2} & \frac{-6}{l^2} \\ 0 & \frac{6}{l^2} & \frac{24}{l^3} & \frac{-12}{l^3} \\ \frac{-6}{l^2} & \frac{-6}{l^2} & \frac{-12}{l^3} & \frac{12}{l^3} \end{bmatrix} \tag{14.22}$$

Breaking out and identifying the four submatrices of dependent (a) and independent (b) degrees of freedom:

$$\mathbf{k}_{aa} = \frac{EI}{l}\begin{bmatrix} 8 & 2 \\ 2 & 4 \end{bmatrix} \qquad \mathbf{k}_{ab} = \frac{EI}{l^2}\begin{bmatrix} 0 & -6 \\ 6 & -6 \end{bmatrix}$$

$$\mathbf{k}_{ba} = \frac{EI}{l^2}\begin{bmatrix} 0 & 6 \\ -6 & -6 \end{bmatrix} \qquad \mathbf{k}_{bb} = \frac{EI}{l^3}\begin{bmatrix} 24 & -12 \\ -12 & 12 \end{bmatrix} \tag{14.23a-d}$$

Finding the inverse of k_{aa} :

$$\mathbf{k}_{aa}^{-1} = \frac{1}{14EI}\begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix} \tag{14.24}$$

$$-\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab} = \frac{-1}{14l}\begin{bmatrix} -6 & -6 \\ 24 & -18 \end{bmatrix} \tag{14.25}$$

$$\mathbf{k}_{ba}\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab} = \frac{EI}{14l^3}\begin{bmatrix} 144 & -108 \\ -108 & 144 \end{bmatrix} \tag{14.26}$$

$$\mathbf{k}_{bb}^{*} = \mathbf{k}_{bb} - \mathbf{k}_{ba}\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab}$$

$$= \frac{EI}{14l^3}\left\{\begin{bmatrix} 336 & -168 \\ -168 & 168 \end{bmatrix} - \begin{bmatrix} 144 & -108 \\ -108 & 144 \end{bmatrix}\right\}$$

$$= \frac{EI}{14l^3}\begin{bmatrix} 192 & -60 \\ -60 & 24 \end{bmatrix}$$

(14.27)

The transformation matrix T is given by:

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{ab} \\ \mathbf{I} \end{bmatrix} = \begin{bmatrix} -\mathbf{k}_{aa}^{-1}\mathbf{k}_{ab} \\ \mathbf{I} \end{bmatrix} = \begin{bmatrix} \frac{6}{14l} & \frac{6}{14l} \\ \frac{-24}{14l} & \frac{18}{14l} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{14.28}$$

The mass matrix now needs to be rearranged into "a" and "b" submatrices and then transformed to $\mathbf{m}_{bb}^{*}$:

$$\mathbf{m}_g = \frac{1}{420}\begin{bmatrix} 312ml & 0 & 54ml & -13ml^2 \\ 0 & 8ml^3 & 13ml^2 & -3ml^3 \\ 54ml & 13ml^2 & 156ml & -22ml^2 \\ -13ml^2 & -3ml^3 & -22ml^2 & 4ml^3 \end{bmatrix} \tag{14.29}$$

Rearranging rows 1 to 3, 2 to 1, 3 to 4 and 4 to 2:

$$\mathbf{m}_g = \frac{1}{420}\begin{bmatrix} 0 & 8ml^3 & 13ml^2 & -3ml^3 \\ -13ml^2 & -3ml^3 & -22ml^2 & 4ml^3 \\ 312ml & 0 & 54ml & -13ml^2 \\ 54ml & 13ml^2 & 156ml & -22ml^2 \end{bmatrix} \tag{14.30}$$

Rearranging columns 1 to 3, 2 to 1, 3 to 4 and 4 to 2:

$$\mathbf{m}_g = \frac{m}{420}\begin{bmatrix} 8l^3 & -3l^3 & 0 & 13l^2 \\ -3l^3 & 4l^3 & -13l^2 & -22l^2 \\ 0 & -13l^2 & 312l & 54l \\ 13l^2 & -22l^2 & 54l & 156l \end{bmatrix} \tag{14.31}$$

Separating into submatrices:

$$\mathbf{m}_{aa} = \frac{ml^3}{420}\begin{bmatrix} 8 & -3 \\ -3 & 4 \end{bmatrix} \qquad \mathbf{m}_{ab} = \frac{ml^2}{420}\begin{bmatrix} 0 & 13 \\ -13 & -22 \end{bmatrix}$$

$$\mathbf{m}_{ba} = \frac{ml^2}{420}\begin{bmatrix} 0 & -13 \\ 13 & -22 \end{bmatrix} \qquad \mathbf{m}_{bb} = \frac{ml}{420}\begin{bmatrix} 312 & 54 \\ 54 & 156 \end{bmatrix} \tag{14.32a-d}$$

Calculating $\mathbf{m}^*_{bb}$:

$$\mathbf{m}^*_{bb} = \mathbf{T}^T\mathbf{m}\mathbf{T} \tag{14.33}$$

Carrying out the multiplications:

$$\mathbf{m}^*_{bb} = ml\begin{bmatrix} \frac{1528}{1715} & \frac{241}{1372} \\ \frac{241}{1372} & \frac{471}{1715} \end{bmatrix} \tag{14.34}$$

14.6 Eigenvalues of Reduced Equations for Two-Element Cantilever, State Space Form

The second order reduced equation of motion is shown in (14.35), (14.36), using the 2x2 stiffness matrix from static condensation, (13.50). We will now generate the state space form of the second order reduced equations. It is useful to see how to convert a second order set of differential equations with a filled (not diagonal) mass matrix to state space form. Once we have the equations of motion in state space form, we will use a symbolic algebra program to solve for the eigenvalues.

$$\mathbf{m}^*_{bb}\ddot{\mathbf{z}}_b + \mathbf{k}^*_{bb}\mathbf{z}_b = [0] \tag{14.35}$$

$$ml\begin{bmatrix} \frac{1528}{1715} & \frac{241}{1372} \\ \frac{241}{1372} & \frac{471}{1715} \end{bmatrix}\begin{bmatrix} \ddot{z}_{b1} \\ \ddot{z}_{b2} \end{bmatrix}+\frac{EI}{14l^3}\begin{bmatrix} 192 & -60 \\ -60 & 24 \end{bmatrix}\begin{bmatrix} z_{b1} \\ z_{b2} \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad (14.36)$$

z_{b1} and z_{b2} are the first two reduced degrees of freedom, the displacements of nodes 2 and 3.

Normally, we would solve each of the equations of motion for the highest derivative and then convert to state space form, but we cannot do that here because the mass matrix is filled, meaning that there is more than one second derivative in each equation. To get around this problem, we will first convert the equation to state space form. We will then take the inverse of the mass matrix and premultiply, leaving only the identity matrix to multiply with the derivative vector.

Converting to state space form, where x_1 and x_2 are displacement and velocity of node 2 and x_3 and x_4 are the displacement and velocity of node 3, respectively:

$$\mathbf{m}_{ss}\dot{\mathbf{x}}+\mathbf{k}_{ss}\mathbf{x}=[0] \qquad (14.37)$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1528ml}{1715} & 0 & \frac{241ml}{1372} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{241ml}{1372} & 0 & \frac{471ml}{1715} \end{bmatrix}\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix}+\begin{bmatrix} 0 & -1 & 0 & 0 \\ \frac{192EI}{14l^3} & 0 & \frac{-60EI}{14l^3} & 0 \\ 0 & 0 & 0 & -1 \\ \frac{-60EI}{14l^3} & 0 & \frac{24EI}{14l^3} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(14.38)

Note that the "1" terms are on the diagonal in the mass matrix and the "-1" terms are off diagonal in the stiffness matrix. Taking the inverse of $\mathbf{m}_{ss}$:

$$\mathbf{m}_{ss}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \dfrac{263760}{205367ml} & 0 & \dfrac{-168700}{205367ml} \\ 0 & 0 & 1 & 0 \\ 0 & \dfrac{-168700}{205367ml} & 0 & \dfrac{855680}{205367ml} \end{bmatrix} \tag{14.39}$$

Premultiplying the equation of motion by $\mathbf{m}_{ss}^{-1}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 & 0 \\ \dfrac{4340280EI}{205367ml^4} & 0 & \dfrac{-1419600EI}{205367ml^4} & 0 \\ 0 & 0 & 0 & -1 \\ \dfrac{-5980800EI}{205367ml^4} & 0 & \dfrac{2189880EI}{205367ml^4} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{14.40}$$

Rewriting without the identity matrix:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 & 0 \\ \dfrac{4340280EI}{205367ml^4} & 0 & \dfrac{-1419600EI}{205367ml^4} & 0 \\ 0 & 0 & 0 & -1 \\ \dfrac{-5980800EI}{205367ml^4} & 0 & \dfrac{2189880EI}{205367ml^4} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{14.41}$$

Converting to standard state space form, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{-4340280EI}{205367ml^4} & 0 & \dfrac{1419600EI}{205367ml^4} & 0 \\ 0 & 0 & 0 & 1 \\ \dfrac{5980800EI}{205367ml^4} & 0 & \dfrac{-2189880EI}{205367ml^4} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{14.42}$$

Using a symbolic algebra program to solve for the eigenvalues:

$$f_{1,2} = \left(\frac{1}{2\pi}\right)\left(\frac{2}{205367}\right)\frac{\sqrt{43127070}\sqrt{EIm(3887 \pm 20\sqrt{34178})}}{ml^2} \qquad (14.43)$$

14.7 MATLAB Code cant_2el_guyan.m – Two-element Cantilever Eigenvalues/Eigenvectors

14.7.1 Code Description

The MATLAB code **cant_2el_guyan.m** solves for the eigenvalues and eigenvectors of a two-element steel cantilever with dimensions of 0.2 x 2 x 20mm. The code does the following, where each time MATLAB calculates a result it is compared to the hand-calculated result:

1) builds mass and stiffness matrices element by element
2) deletes degrees of freedom associated with constrained left-hand end
3) reorders the matrices and performs Guyan reduction
4) converts to state space form
5) calculates eigenvalues/eigenvectors

The code for **cant_2el_guyan.m** is not listed as similar code is used in **cantbeam_guyan.m**, which is listed below.

14.7.2 Code Results

Substituting for E, I, m and l in (14.43) as shown in the code results in eigenvalues of 398.55 and 2521.1 Hz. The first two eigenvalues for a 10-element model using ANSYS (following section) are calculated to be 397.86 and 2493.2 Hz, giving differences between the two-element and 10-element beams of 0.17% and +1.11%, respectively. The differences between the two-element and theoretical values are +0.1697% and +0.0095%, respectively. Archer's consistent mass paper stated that in order to calculate accurate eigenvalues using consistent mass we only needed one more element than the number of accurate modes desired. In this case we found the frequency of the first mode very accurately using only two elements, and the second mode was only off by 1.11%, even with the errors inherent in the Guyan reduction method.

14.8 MATLAB Code cantbeam_guyan.m – User-Defined Cantilever Eigenvalues/Eigenvectors

This MATLAB code solves for the eigenvalues and eigenvectors of a cantilever with user-defined dimensions, material properties, number of elements and number of mode shapes to plot. The code is similar to that in **cant_2el_guyan.m** except that Guyan reduction is an option for this code. If Guyan reduction is chosen, all rotations are reduced, leaving only translations as master degrees of freedom. The code is listed below, but is not broken down and commented because the comments integrated with the code should be sufficient.

In order to compare results with the ANSYS run below, a 10-element beam with the following properties is used: width = 2mm, thickness = 0.2, length = 20mm, modulus = $190e^{6}$ mN/mm^{2}, density = $7.83e^{-6}$ Kg/mm^{3}.

14.9 ANSYS Code cantbeam.inp, Code Description

The ANSYS code solves for the eigenvalues and eigenvectors of the same beam as cantbeam_guyan.m.

14.10 MATLAB cantbeam_guyan.m / ANSYS cantbeam.inp Results Summary

14.10.1 10-Element Beam Frequency Comparison

The Table 14.1 shows the eigenvalues from the 10-element ANSYS and MATLAB runs, both with Guyan reduction, along with theoretical values calculated using the MATLAB code cantbeam_ss_freq_craig.m (Chang 1969). The errors for the first five modes are quite small, with the maximum error (for the ninth mode) being only 6.5%.

Mode No.	MATLAB Cantbeam_ guyan.m	ANSYS Cantbeam.in p	Theoretical	Percent Error, Cantbeam_guyan.m and Theoretical
1	397.88	397.86	397.874572279	-0.0001
2	2493.6	2493.2	2493.437382146	-0.0051
3	6984.5	6982.2	6981.696870181	-0.0408
4	13703	13696	13681.339375292	-0.1646
5	22727	22705	22616.234284744	-0.4887
6	34194	34145	33784.737867762	-1.2113
7	48420	48234	47186.94828572	-2.6126
8	65831	65657	62822.86012645	-4.7893
9	85987	85697	80692.473674351	-6.5619
10	104570	101392	100795.788914948	-3.7445

Table 14.1: 10-element beam frequency comparisons.

14.10.2 20-Element Beam Mode Shape Plots, Modes 1 to 5

Instead of plotting the mode shapes for the 10-element model, we will use a 20-element model to give better resolution and smoother plots. The first five mode shape plots are shown in Figures 14.2 through 14.6 below. Note that for the third and fifth modes the displacements of the middle node are quite small relative to the maximum 1.0. In other words, there is a "node" of the mode near the midpoint of the beam. This meaning for "node" of a mode is not that of a finite element "node," but is a location along the beam where displacement goes to zero for that mode of vibration.

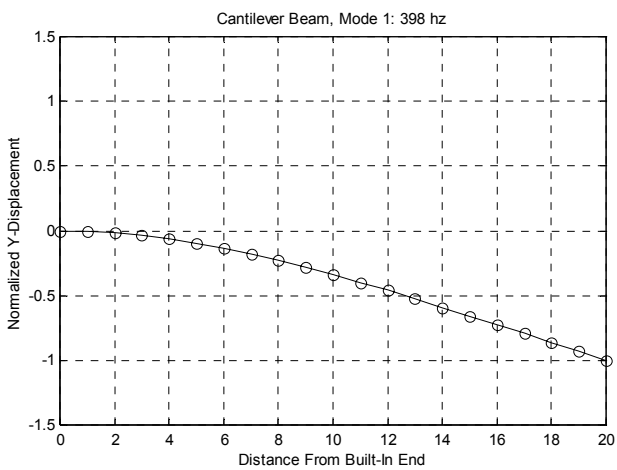

Figure 14.2: Cantilever beam first mode.

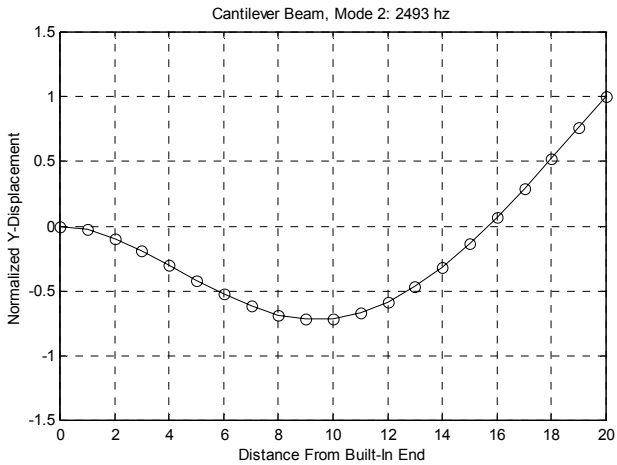

Figure 14.3: Cantilever beam second mode.

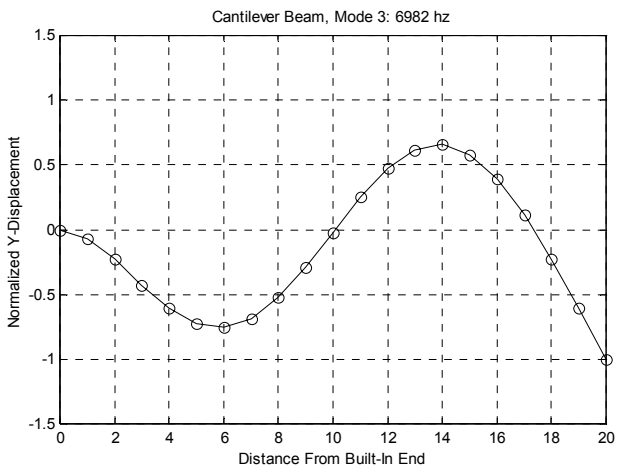

Figure 14.4: Cantilever beam third mode. Note "node" near the beam middle.

We are focusing on "nodes" located near the middle of the beam because in the next chapter we will solve for the frequency responses of a cantilever with a force at the center and output displacement at the tip. We will see that modes with small eigenvector entries for input or output (or both) degrees of freedom are able to be removed from the model, as they contribute little to the input or output of the system.

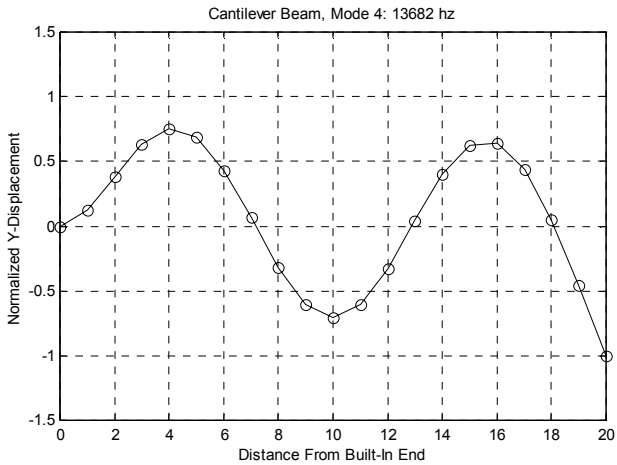

Figure 14.5: Cantilever beam fourth mode.

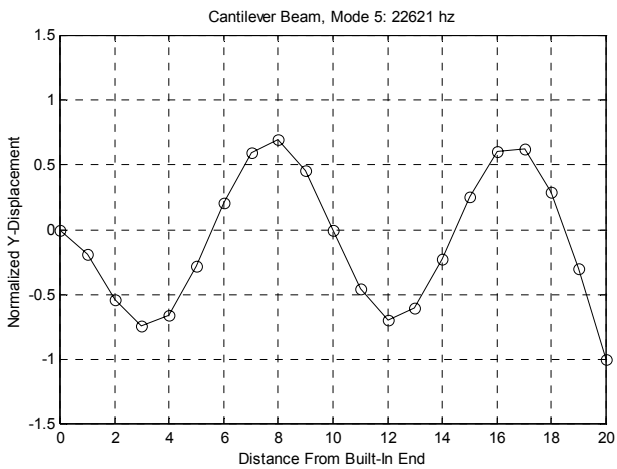

Figure 14.6: Cantilever beam fifth mode. Note the "node" near the midpoint of the beam, and two additional "nodes" to the left and right of the midpoint.

The 10 eigenvectors from the 10-element **cantbeam_guyan.m**, normalized to unity, are shown in Table 14.2. The displacement entry for the built-in left-hand end of the beam is not shown, the 10 rows represent the nodes from left to right, starting with the second node from the end.

Mode: 1	2	3	4	5	6	7	8	9	10
-0.0168	-0.0926	-0.2280	0.3841	-0.5331	-0.6485	0.7129	0.7310	0.7418	-0.6239
-0.0639	-0.3010	-0.6042	0.7519	-0.6535	-0.3274	-0.1055	-0.4942	-0.7714	0.7719
-0.1365	-0.5261	-0.7558	0.4324	0.2109	0.6574	-0.5480	0.0107	0.6458	-0.9023
-0.2299	-0.6834	-0.5256	-0.3153	0.6906	0.1048	0.6100	0.4831	-0.3565	0.9797
-0.3395	-0.7136	**-0.0195**	-0.7053	**-0.0028**	-0.6931	**-0.0029**	-0.6863	**-0.0222**	-1.0000
-0.4611	-0.5894	0.4737	-0.3249	-0.6948	0.1125	-0.6070	0.4771	0.3953	0.9618
-0.5909	-0.3170	0.6571	0.3971	-0.2215	0.6607	0.5534	0.0186	-0.6692	-0.8674
-0.7255	0.0701	0.3945	0.6411	0.5965	-0.3025	0.1160	-0.5089	0.7788	0.7247
-0.8624	0.5238	-0.2288	0.0504	0.2884	-0.4706	-0.5885	0.6466	-0.6636	-0.5252
-1.0000	1.0000	-1.0000	-1.0000	-1.0000	1.0000	1.0000	-1.0000	1.0000	0.7913

Table 14.2: 10-element beam eigenvectors normalized to unity. Note small values for third, fifth, seventh and ninth mode displacements for midpoint node, in bold type.

The presence of a "node" of a mode can be seen numerically for the 10-element MATLAB model by looking at the fifth row (midpoint of beam) of the eigenvector listing in Table 14.2 and noting the small values for the third, fifth, seventh and ninth modes, highlighted in bold type. Getting a good mental picture of the relationship between the plotted mode shape and the eigenvector listing is quite useful. We will see in the next chapter that the small value of node displacements for certain modes of vibration will mean that for certain transfer functions the modes are less important to include in the reduced (smaller number of states used) state space model, and therefore, can be eliminated.

For eigenvector comparison with the ANSYS results, which are normalized with respect to mass instead of unity, the first two eigenvectors for the 10-element MATLAB beam model, are shown below. Compare with the "UZ" columns in the ANSYS listing below.

4.2387	-23.4098
14.1402	-76.0842
34.4892	-132.9666
58.0918	-172.7285
85.7975	-180.3585
116.5287	-148.9709
149.3145	-80.1210
183.3282	17.7069
217.9284	132.3727
252.7000	252.7326

Table 14.3: MATLAB 10-element beam model, first and second eigenvectors normalized with respect to mass.

A listing for the first two modes from the ANSYS code **cantbeam.eig** is shown below. The listing displays the title, resonant frequency (eigenvalue) and a listing of eigenvector entries for each degree of freedom. Even though we used Guyan reduction on the ANSYS model, ANSYS back-calculates the eigenvector values of the reduced dof's so there are eigenvector values for both the UZ and ROTY degrees of freedom below. Since we constrained all the degrees of freedom except the displacement in the z-direction and rotation about the y axis, all other degree of freedom entries for the eigenvectors are zero.

```
*DO LOOP ON PARAMETER= I       FROM  1.0000     TO  10.000     BY  1.0000


 USE LOAD STEP     1  SUBSTEP     1  FOR LOAD CASE 0

 SET COMMAND GOT LOAD STEP=     1  SUBSTEP=     1  CUMULATIVE ITERATION=
1
  TIME/FREQUENCY=  397.86
TITLE=  cantbeam.inp, 0.2 thick x 2 wide x 20mm long steel cantilever beam, 10

 PRINT DOF  NODAL SOLUTION PER NODE

  ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  FREQ=   397.86      LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN GLOBAL COORDINATES

 NODE   UX        UY        UZ        ROTX      ROTY      ROTZ
    1   0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    2   0.0000    0.0000    4.2385    0.0000   -4.1366    0.0000
    3   0.0000    0.0000   16.140     0.0000   -7.6631    0.0000
    4   0.0000    0.0000   34.488     0.0000   -10.586    0.0000
    5   0.0000    0.0000   58.090     0.0000   -12.920    0.0000
    6   0.0000    0.0000   85.796     0.0000   -14.695    0.0000
    7   0.0000    0.0000  116.53      0.0000   -15.954    0.0000
    8   0.0000    0.0000  149.31      0.0000   -16.761    0.0000
    9   0.0000    0.0000  183.32      0.0000   -17.198    0.0000
   10   0.0000    0.0000  217.92      0.0000   -17.366    0.0000
   11   0.0000    0.0000  252.70      0.0000   -17.396    0.0000

MAXIMUM ABSOLUTE VALUES
NODE         0         0        11         0        11         0
VALUE    0.0000    0.0000    252.70    0.0000   -17.396    0.0000

*ENDDO  INDEX= I

  ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     2
```

FREQ= **2493.2** LOAD CASE= 0

THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN GLOBAL COORDINATES

NODE	UX	UY	UZ	ROTX	ROTY	ROTZ
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	**-23.405**	0.0000	21.188	0.0000
3	0.0000	0.0000	**-76.071**	0.0000	29.354	0.0000
4	0.0000	0.0000	**-132.95**	0.0000	25.705	0.0000
5	0.0000	0.0000	**-172.71**	0.0000	12.776	0.0000
6	0.0000	0.0000	**-180.34**	0.0000	-5.7217	0.0000
7	0.0000	0.0000	**-148.96**	0.0000	-25.506	0.0000
8	0.0000	0.0000	**-80.124**	0.0000	-42.575	0.0000
9	0.0000	0.0000	**17.689**	0.0000	-54.169	0.0000
10	0.0000	0.0000	**132.34**	0.0000	-59.449	0.0000
11	0.0000	0.0000	**252.69**	0.0000	-60.537	0.0000

MAXIMUM ABSOLUTE VALUES

NODE	0	0	11	0	11	0
VALUE	0.0000	0.0000	252.69	0.0000	-60.537	0.0000

14.11 MATLAB Code cantbeam_guyan.m Listing

```
        echo off
%       cantbeam_guyan.m    cantilever beam finite element program,
%       selectable number of elements.  Solves for eigenvalues and
%       eigenvectors of a cantilever with user-defined dimensions,
%       material properties, number of elements and number of mode shapes
%       to plot.  Guyan reduction is an option.  A 10 element beam is used
%       as an example.  Default beam is 2mm wide by 20mm long by 0.2mm thick.

        clf;

        clear all;

        inp = input('Input "1" to enter beam dimensions, "Enter" to use default ...  ');

        if (isempty(inp))
                inp = 0;
        else
        end

        if  inp == 0

                wbeam = 2.0
                tbeam = 0.2
                lbeam = 20.0
                E = 190e6
                density = 7.83e-6

        else

%       input size of beam and material
```

```
        wbeam = input('Input width of beam, default 2mm, ...  ');

        if (isempty(wbeam))
                wbeam = 2.0;
        else
        end

        tbeam = input('Input thickness of beam, default 0.2mm, ...  ');

        if (isempty(tbeam))
                tbeam = 0.2;
        else
        end

        lbeam = input('Input length of beam, default 20mm, ...  ');

        if (isempty(lbeam))
                lbeam = 20.0;
        else
        end

        E = input('Input modulus of material, mN/mm^2, default stainless steel 190e6 ...  ');

        if (isempty(E))
                E = 190e6;
        else
        end

        density = input('Input density of material, Kg/mm^3, default stainless steel 7.83e-6 ...
');

        if (isempty(density))
                density = 7.83e-6;
        else
        end

        end

%       input number of elements

        num_elements = input('Input number of elements for beam, minimum 2, default 10 ...
');

        if (isempty(num_elements))
                num_elements = 10;
        else
        end

%       define whether or not to do Guyan Reduction

        guyan = input('enter "1" to do Guyan elimination of rotations, …
                        "enter" to not do Guyan ...  ');

        if (isempty(guyan))
```

```
                guyan = 0;
        else
        end

        if  guyan == 0

                num_plot_max = 2*num_elements;

                num_plot_default = num_elements;

        else

                num_plot_max = num_elements;

                num_plot_default = num_elements;

        end

        num_plot = input(['enter the number of modes to plot, max', …
                num2str(num_plot_max),', default ',num2str(num_plot_default),' ...  ']);

        if (isempty(num_plot))
                num_plot = 9;
        else
        end

%       define length of each element, uniform lengths

        l = lbeam/num_elements;

%       define length vector for plotting, right-to-left numbering

        lvec = 0:l:lbeam;

%       define the node numbers

        n = 1:num_elements+1;

%       number the nodes for the elements

        node1 = 1:num_elements;

        node2 = 2:num_elements+1;

%       size the stiffness and mass matrices to have 2 times the number of nodes
%       to allow for translation and rotation dof's for each node, including built-
%       in end

        max_node1 = max(node1);

        max_node2 = max(node2);

        max_node_used = max([max_node1 max_node2]);

        mnu = max_node_used;
```

```
        k = zeros(2*mnu);

        m = zeros(2*mnu);

%       now build up the global stiffness and consistent mass matrices, element by element


%       calculate I, area and mass per unit length of beam

        I = wbeam*tbeam^3/12;

        area = wbeam*tbeam;

        mpl = density*area;

        for  i = 1:num_elements

                dof1 = 2*node1(i)-1;
                dof2 = 2*node1(i);
                dof3 = 2*node2(i)-1;
                dof4 = 2*node2(i);

                k(dof1,dof1) = k(dof1,dof1)+(12*E*I/l^3);
                k(dof2,dof1) = k(dof2,dof1)+(6*E*I/l^2);
                k(dof3,dof1) = k(dof3,dof1)+(-12*E*I/l^3);
                k(dof4,dof1) = k(dof4,dof1)+(6*E*I/l^2);

                k(dof1,dof2) = k(dof1,dof2)+(6*E*I/l^2);
                k(dof2,dof2) = k(dof2,dof2)+(4*E*I/l);
                k(dof3,dof2) = k(dof3,dof2)+(-6*E*I/l^2);
                k(dof4,dof2) = k(dof4,dof2)+(2*E*I/l);

                k(dof1,dof3) = k(dof1,dof3)+(-12*E*I/l^3);
                k(dof2,dof3) = k(dof2,dof3)+(-6*E*I/l^2);
                k(dof3,dof3) = k(dof3,dof3)+(12*E*I/l^3);
                k(dof4,dof3) = k(dof4,dof3)+(-6*E*I/l^2);

                k(dof1,dof4) = k(dof1,dof4)+(6*E*I/l^2);
                k(dof2,dof4) = k(dof2,dof4)+(2*E*I/l);
                k(dof3,dof4) = k(dof3,dof4)+(-6*E*I/l^2);
                k(dof4,dof4) = k(dof4,dof4)+(4*E*I/l);

                m(dof1,dof1) = m(dof1,dof1)+(mpl/420)*(156*l);
                m(dof2,dof1) = m(dof2,dof1)+(mpl/420)*(22*l^2);
                m(dof3,dof1) = m(dof3,dof1)+(mpl/420)*(54*l);
                m(dof4,dof1) = m(dof4,dof1)+(mpl/420)*(-13*l^2);

                m(dof1,dof2) = m(dof1,dof2)+(mpl/420)*(22*l^2);
                m(dof2,dof2) = m(dof2,dof2)+(mpl/420)*(4*l^3);
                m(dof3,dof2) = m(dof3,dof2)+(mpl/420)*(13*l^2);
                m(dof4,dof2) = m(dof4,dof2)+(mpl/420)*(-3*l^3);

                m(dof1,dof3) = m(dof1,dof3)+(mpl/420)*(54*l);
                m(dof2,dof3) = m(dof2,dof3)+(mpl/420)*(13*l^2);
```

```
                        m(dof3,dof3) = m(dof3,dof3)+(mpl/420)*(156*l);
                        m(dof4,dof3) = m(dof4,dof3)+(mpl/420)*(-22*l^2);

                        m(dof1,dof4) = m(dof1,dof4)+(mpl/420)*(-13*l^2);
                        m(dof2,dof4) = m(dof2,dof4)+(mpl/420)*(-3*l^3);
                        m(dof3,dof4) = m(dof3,dof4)+(mpl/420)*(-22*l^2);
                        m(dof4,dof4) = m(dof4,dof4)+(mpl/420)*(4*l^3);

            end

%           now that stiffness and mass matrices are defined for all dof's, including
%           constrained dof's, need to delete rows and columns of the matrices that
%           correspond to constrained dof's, in the left-to-right case, the first two
%           rows and columns

            k(1:2,:) = [];              % translation/rotation of node 1
            k(:,1:2) = [];

            m(1:2,:) = [];
            m(:,1:2) = [];

            if  guyan == 1

%           Guyan Reduction - reduce out the rotation dof's, leaving displacement dof's
%           re-order the matrices

%           re-order the columns of k

            kr = zeros(2*(mnu-1));

            krr = zeros(2*(mnu-1));

%   rearrange columns, rotation and then displacement dof's

            mkrcolcnt = 0;

            for  mkcolcnt = 2:2:2*(mnu-1)

                        mkrcolcnt = mkrcolcnt + 1;

                        kr(:,mkrcolcnt) = k(:,mkcolcnt);

                        mr(:,mkrcolcnt) = m(:,mkcolcnt);

            end

            mkrcolcnt = num_elements;

            for  mkcolcnt = 1:2:2*(mnu-1)

                        mkrcolcnt = mkrcolcnt + 1;

                        kr(:,mkrcolcnt) = k(:,mkcolcnt);

                        mr(:,mkrcolcnt) = m(:,mkcolcnt);
```

```
        end
%   rearrange rows, rotation and then displacement dof's
        mkrrowcnt = 0;
        for  mkrowcnt = 2:2:2*(mnu-1)
                mkrrowcnt = mkrrowcnt + 1;
                krr(mkrrowcnt,:) = kr(mkrowcnt,:);
                mrr(mkrrowcnt,:) = mr(mkrowcnt,:);
        end
        mkrrowcnt = num_elements;
        for  mkrowcnt = 1:2:2*(mnu-1)
                mkrrowcnt = mkrrowcnt + 1;
                krr(mkrrowcnt,:) = kr(mkrowcnt,:);
                mrr(mkrrowcnt,:) = mr(mkrowcnt,:);
        end
%       define sub-matrices and transformation matrix T
        kaa = krr(1:num_elements,1:num_elements);
        kab = krr(1:num_elements,num_elements+1:2*num_elements);
        T = [-inv(kaa)*kab
                        eye(num_elements,num_elements)]
%       calculate reduced mass and stiffness matrices
        kbb = T'*krr*T
        mbb = T'*mrr*T
        else
        kbb = k;
        mbb = m;
        end
%       define the number of dof for state-space version, 2 times dof left after
%       removing constrained dof's
```

```
        [dof,dof] = size(kbb);
%       define the sizes of mass and stiffness matrices for state-space
        ssdof = 2*dof;
        aud = zeros(ssdof);                 % creates a ssdof x ssdof null matrix
%       divide the negative of the stiffness matrix by the mass matrix
        ksm = inv(mbb)*(-kbb);
%       now expand to state space size
%       fill out unit values in mass and stiffness matrices
        for  row = 1:2:ssdof
                aud(row,row+1) = 1;
        end
%       fill out mass and stiffness terms from m and k
        for  row = 2:2:ssdof
                for  col = 2:2:ssdof
                        aud(row,col-1) = ksm(row/2,col/2);
                end
        end
%       calculate the eigenvalues/eigenvectors of the undamped matrix for plotting
%       and for calculating the damping matrix c
        [evec1,evalu] = eig(aud);
        evalud = diag(evalu);
        evaludhz = evalud/(2*pi);
        num_modes = length(evalud)/2;
%       now reorder the eigenvalues and eigenvectors from low to high freq
        [evalorder,indexhz] = sort(abs((evalud)));
        for  cnt = 1:length(evalud)
                eval(cnt,1) = evalud(indexhz(cnt));
                evalhzr(cnt,1) = round(evaludhz(indexhz(cnt)));
                evec(:,cnt) = evec1(:,indexhz(cnt));
```

```
        end
%       now check for any imaginary eigenvectors and convert to real
        for  cnt = 1:length(evalud)
                if  (imag(evec(1,cnt)) & imag(evec(3,cnt)) & imag(evec(5,cnt))) ~= 0
                        evec(:,cnt) = imag(evec(:,cnt));
                else
                end
        end
        if  guyan == 0
%       now separate the displacement and rotations in the eigenvectors
%       for plotting mode shapes
        evec_disp = zeros(ceil(dof/2),ssdof);
        rownew = 0;
        for  row = 1:4:ssdof
                rownew = rownew+1;
                evec_disp(rownew,:) = evec(row,:);
        end
        evec_rotation = zeros(ceil(dof/2),ssdof);
        rownew = 0;
        for  row = 3:4:ssdof
                rownew = rownew+1;
                evec_rotation(rownew,:) = evec(row,:);
        end
        else
        evec_disp = zeros(ceil(dof/4),ssdof);
        rownew = 0;
        for  row = 1:2:ssdof
                rownew = rownew+1;
```

```
                    evec_disp(rownew,:) = evec(row,:);

            end

            end

%           normalize the displacement eigenvectors wrt one for plotting

            for  col = 1:ssdof

                    evec_disp(:,col) = evec_disp(:,col)/max(abs(real(evec_disp(:,col))));

                    if  evec_disp(floor(dof/2),col) >= 0

                            evec_disp(:,col) = -evec_disp(:,col);

                    else
                    end

            end

%           list eigenvalues, hz

            format long             e

            evaludhz_list = sort(evaludhz(1:2:2*num_modes))

            format short

%           list displacement (not velocity) eigenvectors

            evec_disp(:,1:2:2*num_plot)

            if  guyan == 0

%           plot mode shapes

            for  mode_cnt = 1:num_plot

                    evec_cnt = 2*mode_cnt -1;

                    plot(lvec,[0; evec_disp(:,evec_cnt)],'ko-')
                    title(['Cantilever Beam, Mode ', ...
                            num2str(mode_cnt),': ',num2str(abs(evalhzr(evec_cnt))),' hz']);
                    xlabel('Distance From Built-In End')
                    ylabel('Normalized Y-Displacement')
                    axis([0 lbeam -1.5 1.5])
                    grid on

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

            end

            else
```

```
%       plot mode shapes, Guyan Reduced

        for  mode_cnt = 1:num_plot

                evec_cnt = 2*mode_cnt -1;

                plot(lvec,[0; evec_disp(:,evec_cnt)],'ko-')
                title(['Cantilever Beam, Mode ', ...
                        num2str(mode_cnt),': ',num2str(abs(evalhzr(evec_cnt))),' hz']);
                xlabel('Distance From Built-In End')
                ylabel('Normalized Y-Displacement')
                axis([0 lbeam -1.5 1.5])
                grid on

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

        end

        end

%       normalization with respect to mass on a filled (not diagonal) mass matrix

%       calculate the displacement (displacement and rotation) eigenvectors
%       to be used for the modal model eigenvectors

        xm = zeros(dof);

        col = 0;

        for  mode = 1:2:ssdof

                col = col + 1;

                row = 0;

                for          ndof = 1:2:ssdof

                        row = row + 1;

                        xm(row,col) = evec(ndof,mode);

                end

        end

%       normalize with respect to mass

        for  mode = 1:dof

                xn(:,mode) = xm(:,mode)/sqrt(xm(:,mode)'*mbb*xm(:,mode));

        end

%       calculate the normalized mass and stiffness matrices for checking
```

```
        mm = xn'*mbb*xn;

        km = xn'*kbb*xn;

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

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

        row = 0;

        for cnt = 1:2:ssdof

                row = row + 1;

                evalrad(row) = abs((eval(cnt)));

        end

        [p evalrad']/(2*pi)

        evalhz = evalrad/(2*pi);

        semilogy(evalhz)
        title('Resonant Frequencies, Hz')
        xlabel('Mode Number')
        ylabel('Frequency, hz')
        grid
        disp('execution paused to display figure, "enter" to continue'); pause
```

14.12 ANSYS Code cantbeam.inp Listing

```
/title, cantbeam.inp, 0.2 thick x 2 wide x 20mm long steel cantilever beam, 10 elements

/prep7

et,1,4                  ! element type for beam

! steel

ex,1,190e6                      ! mN/mm^2
dens,1,7.83e-6                  ! kg/mm^3
nuxy,1,.293

! real value to define beam characteristics

r,1,0.4,0.1333,0.0013333,0.2,2          ! area, Izz, Iyy, TKz, TKy

! define plotting characteristics

/view,1,1,-1,1      ! iso view
/angle,1,-60        ! iso view
/pnum,mat,1         ! color by material
```

```
/num,1          ! numbers off
/type,1,0       ! hidden plot
/pbc,all,1      ! show all boundary conditions

csys,0                                  ! define global coordinate system

! nodes

n,1,0,0,0                       ! left-hand node
n,11,20,0,0                             ! right-hand node

fill,1,11                       ! interior nodes

nplo

! elements

type,1
mat,1
real,1
e,1,2
egen,10,1,-1

! constrain left-hand end

d,1,all,0                       ! constrain node 1, all dof's

! constrain all but uz and roty for all other nodes to allow only those dof's
! this will give 10 nodes, node 2 through node 11, each with 2 dof, giving a total of 20 dof
! can calculate a maximum of 20 eigenvalues if don't use Guyan reduction to reduce size of
! eigenvalue problem, maximum of 10 eigenvalues if use Guyan reduction

nall
nsel,s,node,,2,11
d,all,ux
d,all,uy
d,all,rotx
d,all,rotz

allsel
nplo
eplo

! ******************  eigenvalue run  ******************

fini        ! fini just in case not in begin

/solu       ! enters the solution processor, needs to be here to do editing below

allsel      ! default selects all items of specified entity type, typically nodes, elements

nsel,s,node,,2,11
m,all,uz

antype,modal,new
```

```
modopt,reduc,10        ! method - reduced Householder, number of modes to extract
expass,off             ! key = off, no expansion pass, key = on, do expansion
mxpand,10,,,no         ! nummodes to expand
total,10,1             ! total masters, 10 to be used, exclude rotational dofs

allsel

solve                  ! starts the solution of one load step of a solution sequence, modal here

fini

! plot first mode

/post1

set,1,1

pldi,1


! ******************  output frequencies  **********************

/output,cantbeam,frq        ! write out frequency list to ascii file .frq

set,list

/output,term                                          ! returns output to terminal

! *****************  output eigenvectors  *****************

! define nodes for output:  forces applied or output displacements

nall
!nsel,s,node,,11         ! cantilever tip

/output,cantbeam,eig        ! write out eigenvectors to ascii file .eig

*do,i,1,10
            set,,i
            prdisp
*enddo

/output,term
******************  plot modes  ******************

! pldi plots

/show,cantbeam,grp,0
allsel

/view,1,,-1,,           ! side view for plotting
/angle,1,0
/auto

*do,i,1,10
```

```
            set,1,i
   pldi
*enddo

/show,term
```

Problem

P14.1 Modify the **cantbeam_guyan.m** code to allow variable material and geometry properties along the beam by converting the following scalar quantities into user defined vector quantities: wbeam, tbeam, E, density.

Run the modified code for a 20mm long beam with the twice the default values for the left half of the beam and the default parameters for the right-hand side. Plot eigenvalues in hz versus mode number. Plot the first five mode shapes.