Numerical Simulation of Vibrations of Mechanical Structures

Adaku Uchendu

A Senior Thesis submitted to the faculty of
University of Maryland Baltimore County
in partial fulfillment of the requirements for the degree of

Mathematics, BS

Bedřich Sousedík, Advisor

Department of Mathematics and Statistics

University of Maryland Baltimore County

November 2017

ABSTRACT

Numerical Simulation of Vibrations of Mechanical Structures

Adaku Uchendu
Department of Mathematics and Statistics, UMBC
Mathematics, BS

We develop an implementation of finite element method to simulate vibrations of mechanical structures. Specifically, we use a 2D frame model and corresponding stiffness, mass and damping matrices to set up a system of ordinary differential equations, which is solved in Matlab. We also consider uncertainties in the model parameters by taking the Young's modulus as a random variable. We use Monte Carlo simulation, and the effect of uncertainties is studied by numerical experiments.

Keywords: Mechanical Structures, Mass matrix, Stiffness Matrix, Damping Matrix, Finite Element Method, Degrees of freedom (dof)

ACKNOWLEDGMENTS

The work was supported by the National Science Foundation (award DMS-1521563) and the University of Maryland Baltimore County, Undergraduate Research Award (URA). I would like to thank Dr. Bedrich Sousedik for giving me the opportunity to both learn and conduct this research under his invaluable tutelage. Also, I would like to my Parents for always supporting my research career. Lastly, I would like to thank God.

Contents

Chapter 1

Introduction

The volume of earthquake engineering research has increased in proportion to the severity and frequency of earthquakes around the world. The destruction of civil and mechanical structures caused by these earthquakes fostered large interest in the study of earthquakes. As a result of the influx of active researchers, development of new structures and retrofitting of already existing structures to withstand earthquakes and wind has grown. This is where structural control design becomes important because without it, it would be impossible to create structures capable of withstanding earthquakes. Since during earthquakes, structures emulate the behavior of springs by their movement, the structure was modeled as coupled springs.

Structural control can be defined as the effective measures put in place to ensure that a structure is performing as it should. Structural design is the procedure by which structures are inspected for their strength, stability and rigidity. These three qualities ensure that a structure is efficient for what it was built for. An application of this research can be seen in the Taipei 101, an architectural beauty in Taiwan with a gigantic sphere at the top of the building, balanced by steel cables that swings in the reverse direction of the swaying building during an earthquake, consequently, dissipating the energy and vibrational consequences caused by earthquakes and typhoons. In addition, it is worth noting that the findings of this research are based on mathematical models and not real life buildings or structures. This is for the engineers to take the results and replicate it on real life mechanical structures in ways like the Tapiei 101.

In the article [1] the researchers use a program called FRAME, a finite element and optimal structural control program. It is used to generate a dynamic finite element model of a two-dimensional structure that acts like a mechanical structure, as a procedure to study earthquakes. The 2-D model is simulated in MATLAB to create a moving picture of the frame. This is known as a MATLAB movie and it shows a structure that swerves to the right and left continuously for a given time, thus emulating a structure during an earthquake. Since the force the earthquake exerts on a structure cannot always be fully cushioned, the whole point of this research is to at least subtract some of the force to avoid collapse of buildings when force is too much. I would be using the FRAME model in my research to emulate a structure.

Next, while conducting this research study, the first step was to derive a mathematical model of the structure to simulate in MATLAB. Since this structure is a large system with vibrational

properties, it was only fitting that I used differential equations and matrix algebra to model. This will be discussed in detail in the next chapters. Thus, the next chapters will discuss the steps taken in this research to achieve the utimate goal. Also, they will discuss the basics of mechanical vibrations which involves a discussion of two examples, an introduction to a Finite Element Method application of a frame, the derivation of a more complex frame, its simulation in MATLAB the monte carlo simulation of this frame and lastly, the conclusions obtained from all these analysis.

Finally, the ultimate goal of this research is to provide structural engineers with the mathematical model and measurements needed to make an efficient structural design to mitigate the effects of earthquakes and wind on structures. However, it is important to note that achieving a 100% efficient structure is nearly impossible since the exact earthquake magnitude is unknown until after the earthquake occurs. Therefore, this work solely operates on estimations which are really good but not perfect.

Chapter 2

Basics of Mechanical Vibrations

This chapter will discuss two examples of mechanical vibration systems. The examples to be explored are; trajectory movie and coupled strings. The trajectory movie example captures the oscillation of a mechanical structure by making a moving picture in MATLAB. The, second example, coupled strings models the motion of the spring with newton's second law of motion and plots the solution.

2.1 Trajectory movie

A MATLAB movie can be described as a gif or a moving picture created in MATLAB. This can be done by generating a sequence of figures, saving it and displaying them coherently using the appropraite fuctions which could be either VideoWriter, movie2avi, movie and some other movie functions. A skill like this is useful in visualizing the behavior of vibrating structures such as any mechanical vibration systems. Thus, for this exercise, the MATLAB code in [2] was modified by adding a movie function to the script. The initial MATLAB code solves a differential equation and plots the solution. See figure 2.1 for the solution plot. However, the improved code allows the user

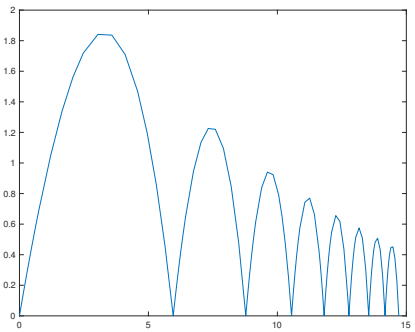

Figure 2.1 Coupled springs diagram

to view the motion on the curves of the plot. Now, first loop obtains the number of bounces from the number of peaks in the figure. See first loop below.

```
for n = 1:max_number_bounces
    max_number_bounces = 8;
    [tbounce,wbounce] = ode45(@odefunc,[tstart,tstop],w0,options);

    % This appends the solution for the nth bounce
    % to the end of the full solution
    n_steps = length(tbounce); % Extracts # of steps in new solution
    % This appends the new times to end of the existing ones
    times = [times;tbounce(1:n_steps)];
    % This appends new solution to end of y
    sols = [sols;wbounce(1:n_steps,:)];

    % This sets the initial conditions for the next bounce
    w0 = wbounce(n_steps,:); % Initial condition is sol at end of bounce
    w0(4) = -w0(4); % But we change the direction of vertical velocity
    tstart = tbounce(n_steps);
end
```

This loop runs eight times, obtains the necessary variables for each frame and saves them in a matrice, *sols*. Then, the next loop plots the eight different solutions for the differential equations obtained and saved in *sols* in the first loop. See second loop below.

```
A = sols(:,1);B = sols(:,2);
axis_length = [15 2];
for t=1:numel(A)
    plot(A(t),B(t),'ro','MarkerSize',20,'MarkerFaceColor','b');
    axis([0,axis_length(1),0,axis_length(2)]);
    M(t) = getframe;
end
```

The second loop is exactly where the movie is created. The line *M(t) = getframe* captures the frames and saves them to $M(t)$, a function of time which displays the MATLABmovie when it is called. The next script below is known as the testing script. It calls the function that displays the movie. See script below.

```
%M = movie_pplot_vector_ode(V0,theta,c,max_number_bounces)
M = movie_pplot_vector_ode(10,45,0.1,8);
movie(M,2,3);
```

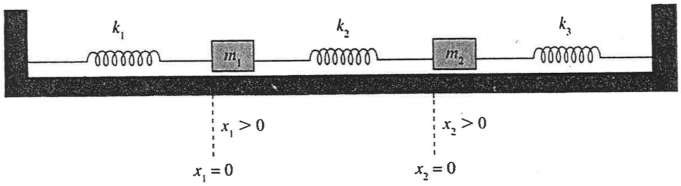

Figure 2.2 Coupled springs plot

2.2 Coupled Springs

Springs are mechanical structures which possess vibrational ability. The behavior of springs can be applied to other mechanical structures which is why understanding the behavior of springs is important.

Problem Statement: Two equal masses $m_1 = m_2 = m$ are attached to three springs, each having the same spring constant $k_1 = k_2 = k_3 = k$, where the two outside springs are attached to walls. The masses slide in a straight line on a frictionless surface. The system is set in motion by holding the left mass in its equilibrium position while at the same time pulling the right mass to the right of its equilibrium a distance of d. Find the subsequent motion of the masses. This can be found in Example 3 of page 378 in [4]. See figure 2.2 to visualize problem statement.

The goal of this exercise is to derive the solution of the differential equation and plot it. The first step is to set up a system of differential equations. From the problem statement and figure 2.2, we get that the system of equations is as follows:

$$\begin{aligned} m_1\ddot{x_1} &= -k_1x_1 + k_2(x_2 - x_1), \\ m_2\ddot{x_2} &= -k_2(x_2 - x_1) - k_3x_2, \end{aligned} \tag{2.1}$$

with initial conditions:

$$\begin{aligned} x_1(0) = 0,\ \dot{x_1}(0) = 0, \\ x_2(0) = 0,\ \dot{x_2}(0) = 0. \end{aligned}$$

Equation (2.1) is modeled by Newton's second law of motion:

$$m\ddot{x} + c\ddot{x} + kx = 0, \tag{2.2}$$

where m, c and k are the mass, damping and stiffness constant of problem. And x is the displacement. However, for this problem there is no damping, so $c = 0$. Then, the equation is refined as:

$$m\ddot{x} + kx = 0. \tag{2.3}$$

Now, solving for $\ddot{x}$, the result is:

$$\ddot{x} = -\frac{k}{m}x. \tag{2.4}$$

Since there are two systems of equations in (2.1), the next step is to insert it in a matrix for simplicity to obtain:

$$\ddot{x} = \begin{bmatrix} \ddot{x_1} \\ \ddot{x_2} \end{bmatrix}, \; M = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}. \tag{2.5}$$

Solving for $\ddot{x}$ in system of equations in (2.1):

$$\begin{aligned} \ddot{x_1} &= -\frac{(k_1+k_2)}{m_1}x_1 + \frac{k_2}{m_1}x_2, \\ \ddot{x_2} &= \frac{k_2}{m_2}x_1 - \frac{(k_2+k_3)}{m_2}x_2 \end{aligned} \tag{2.6}$$

The K matrix can now be derived to be:

$$Kx = \begin{bmatrix} -(k_1+k_2) & k_2 \\ k_2 & -(k_2+k_3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{2.7}$$

The next step is to find $\ddot{x}$. This is achieved by manipulating equations (2.5), to get:

$$\ddot{x} = -M^{-1}Kx = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}^{-1} \begin{bmatrix} -(k_1+k_2) & k_2 \\ k_2 & -(k_2+k_3) \end{bmatrix}, \tag{2.8}$$

which can be further simplified as:

$$\ddot{x} = \begin{bmatrix} -\frac{(k_1+k_2)}{m_1} & 0 \\ 0 & -\frac{(k_2+k_3)}{m_2} \end{bmatrix} \begin{bmatrix} \ddot{x_1} \\ \ddot{x_2} \end{bmatrix}. \tag{2.9}$$

Since $m_1 = m_2 = m$ and $k_1 = k_2 = k_3 = k$, let us assume that $k = m = 1$ for simplicity. Below is the code that computes the solution to the problem using the informtaion derived.

```
function xdot = spring3(t,x)
xdot = zeros(4,1);  %Matrix for the equations
k1 = 1;
k2 = 1;
k3 = 1;
m1 = 1;
m2 = 1;
xdot(1) = x(2);
xdot(2) = -(k1/m1)*x(1)+(k2/m1)*(x(3)-x(1));
xdot(3) = x(4);
xdot(4) = -(k2/m2)*(x(3)-x(1))-(k3/m2)*x(3);
```

And this code is tested in the script below.

```
function test_spring3(time)  % test with time=20

[t,x] = ode45('spring3',[0 time],[0;0;2;0]);
plot(t,x(:,1),t,x(:,3))
legend('xdot(1)','xdot(3)')
```

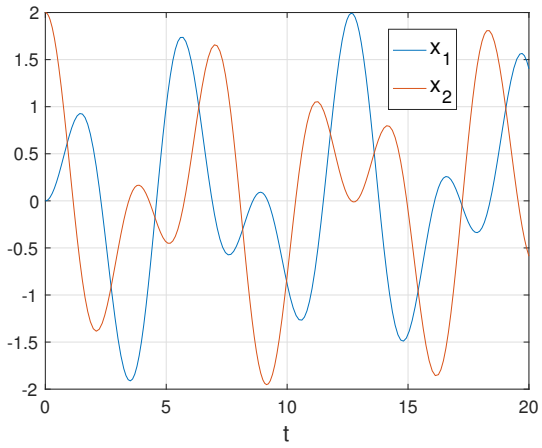

Figure 2.3 Solution of coupled springs problem (x_1, x_2).

The plots generated by the scripts are the same as the ones provided in [4] which confirms that figure 2.3 does in fact contain the correct plots.

Chapter 3

Frame Finite Element Method

The *Finite Element Method* is a powerful numerical method used in solving Engineering and Physics problems. It is used in problems like structural analysis such as the problem proposed in this chapter. The Finite Element Method works by dividing a problem (i.e frame in this case) into smaller parts which are known as finite elements. These finite elements are then modeled with differential equations individually and assembled together as a whole. We will be applying the finite element method to the exercise problem below.

Exercise Problem: Find the natural frequencies of an L-shaped frame that is made of two beams of length of $1m$ each. Both beams have cross-sections of $0.01m$ by $0.01m$. The elastic modulus is $100GPa$. The beam has mass density of $1000\frac{kg}{m^3}$. Use 10 elements. This problem is from [3].

For easy understanding, there will be an assumption that the frame is a 2-Dimensional object. Since the frame is considered to have rotational, transverse and axial movement, the frame is said to have 3 degrees of freedom (dof). These dof are represented as functions such that $v(x), u(x)$, and $\theta(x)$ are transverse deflection, axial deflection and rotational displacement respectively. Here, the goal is to derive the mass and stiffness matrix in order to ultimately find the natural frequency of the frame. This is achieved by implementing the finite element method on the frame structure.

The natural frequency of a structure is the distribution of it's innate energy of which it oscillates with. Knowing the natural frequency of a frame has many advantages, some of which are; it allows for the derivation of the appropriate damping force needed by a structure to withstand vibrations that maybe caused by earthquakes or other natural disasters. Therefore, due to such applications, the final step of the solution to the problem statement is to derive the natural frequency of the L-shaped frame. We achieve this using the eigenvalue analysis method because the frame has more than one degree of freedom. Thus, in taking the eigenvalue of the equation of motion of the frame, we obtain the following equation:

$$\det[K - \omega_n^2 M] = 0, \tag{3.1}$$

where K and M are the stiffness and mass matrices, respectively. And ω_n^2 is the natural frequency of the frame with respect to the nth degree of freedom. Equation (3.1) will be further understood in the next sections.

3.1 Stiffness Matrix

This section focuses on the derivation of the stiffness matrix on Newton's second law of motion model introduced above. It is already known that the frame has three nodal variables or dof, which are the following functions: $v(x), u(x)$, and $\theta(x)$ that represent the transverse deflection, axial deflection and rotational displacement, respectively.

For simplicity, this section will begin by observing the finite element method for a frame with two dof. The dof for this frame would be transverse and rotational deflection with functions: $v(x)$ and $\theta(x)$, respectively. Since every element has two nodes and each node has 2 dof, it implies that each element has 4 dof. This means that to account for all these dof per element, the transverse function would be a third degree polynomial expressed as:

$$v(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \tag{3.2}$$

Which is derived by the fourth integration of $\frac{\delta^4 v}{\delta x^4} = 0$, an equation of equilibrium for a beam element in the unloaded region. Next, considering the rotation deflection, $\theta(x)$ which is a the slope of the transverse deflection, $\theta = \frac{dv}{dx}$, it can be further expressed as:

$$\theta(x) = a_1 + 2a_2 x + 3a_3 x^2. \tag{3.3}$$

To account for everything, the boundary equations for equations (3.2) and (3.3) are:

$$\begin{aligned} v(x) = v_1,\ \theta(x) = \theta_1, \\ v(L) = v_2,\ \theta(L) = \theta_2. \end{aligned} \tag{3.4}$$

This is because the frame has a starting point, 0 and an end point, L for each node in an element. Then, applying the boundary conditions on the equations, the result is:

$$\begin{aligned} v(0) &: a_0 = v_1 \\ \theta(0) &: a_1 = \theta_1 \\ v(l) &: a_0 + a_1 l + a_2 l^2 + a_3 l^3 = v_2 \\ \theta(l) &: a_1 + 2a_2 l^2 + 3a_3 l^2 = \theta_2. \end{aligned} \tag{3.5}$$

After Substituting equation (3.5) into (3.2), the following is obtained:

$$v(x) = v_1 + x\theta_1 - \frac{3x^2}{L^2}v_1 - \frac{2x^2}{L}\theta_1 + \frac{3x^2}{L^2}v_2 - \frac{x^2}{L}\theta_2 + \frac{2x^3}{L^3}v_1 + \frac{x^3}{L^2}\theta_1 - \frac{2x^3}{L^3}v_2 + \frac{x^3}{L^2}\theta_2. \tag{3.6}$$

However, after much simplification, the function is:

$$\left(1 - \frac{3x^2}{L^2} + \frac{2x^3}{L^3}\right)v_1 + \left(x - \frac{2x^2}{L} + \frac{x^3}{L^2}\right)\theta_1 + \left(\frac{3x^2}{L^2} - \frac{2x^3}{L^3}\right)v_2 + \left(\frac{x^3}{L^2} - \frac{x^2}{L}\right)\theta_2. \tag{3.7}$$

Finally, the transverse deflection function can then be further expressed as:

$$v(x) = H_1(x)v_1 + H_2(x)\theta_1 + H_3(x)v_2 + H_4(x)\theta_2, \tag{3.8}$$

where:

$$\begin{aligned} H_1(x) &= 1 - \frac{3x^2}{l^2} + \frac{2x^3}{l^3}, \\ H_2(x) &= x - \frac{2x^2}{l} + \frac{x^3}{l^2}, \\ H_3(x) &= \frac{3x^2}{l^2} - \frac{2x^3}{l^3}, \\ H_4(x) &= \frac{-x^2}{l} + \frac{x^3}{l^2}. \end{aligned} \tag{3.9}$$

$H(x)$ is known as Hermite shape functions. They are used to ensure that the deflection and slope are continuous between elements. Since, these Hermite functions are clearly differentiable, it implies that they are continuous and therefore also means that $v(x)$ and $\theta(x)$ are continuous between neighboring elements. These functions represents the 4 degrees-of-freedom for each element. After, much derivations and calculations, it can be concluded that the formula for constructing an element stiffness matrix according to the beam theory is:

$$[K^e] = \int_0^l [B]^T EI[B]\, dx. \tag{3.10}$$

where,

$$[B] = [H_1'' \; H_2'' \; H_3'' \; H_4'']. \tag{3.11}$$

However, after deriving the *Euler-Bernoulli Beam* equation, the moment and shear force function, are:

$$\begin{aligned} m(x) &= EI\, \frac{d^2v(x)}{dx^2}, \\ V(x) &= EI\, \frac{d^3v(x)}{dx^3}, \end{aligned} \tag{3.12}$$

with boundary conditions:

$$\begin{aligned} V(0) = F_{1,y},\ m(0) = m_1, \\ V(L) = F_{2,y},\ m(L) = m_2 \end{aligned} \tag{3.13}$$

Now, applying the boundary equations in (3.13) to equations (3.12), the result is:

$$\begin{aligned} F_{1,y} = V(0) &= \frac{EI}{L^3}(12v_1 + 6L\theta_1 - 12v_2 + 6L\theta_2), \\ m_1 = -m(0) &= \frac{EI}{L^3}(6Lv_1 + 4L^2\theta_1 - 6Lv_2 + 2L^2\theta_2), \\ F_{2,y} = -V(L) &= \frac{EI}{L^3}(-12v_1 - 6L\theta_1 + 12v_2 - 6L\theta_2), \\ m_2 = m(L) &= \frac{EI}{L^3}(6Lv_1 + 2L^2\theta_1 - 6Lv_2 + 4L^2\theta_2) \end{aligned} \tag{3.14}$$

Then, inserting these equations in (3.14) in a matrix for easy visualization, since there are 4 systems of equation, the result is:

$$[K^e] = \begin{Bmatrix} F_{1,y} \\ m_1 \\ F_{2,y} \\ m_2 \end{Bmatrix} = \frac{EI}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix} \begin{Bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{Bmatrix}. \tag{3.15}$$

Now that the mass and stiffness matrix for a 2 dof frame is known, the same formula can be applied to find the mass and stiffness for a frame with 3 dof. This entails accounting for the axial deflection for each node in an element. The axial stiffness matrix of a beam with a constant cross-sectional area is:

$$k_{axial} = \frac{AE}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{3.16}$$

k_{axial} is the appropriate axial stiffness matrix for the frame since it has a constant cross-sectional area. Then, combining the axial stiffness matrix with the 2 dof frame matrix in equation (3.15), the stiffness matrix for the L-shaped beam is:

$$[K^e] = \frac{E}{l^3} \begin{bmatrix} Al^2 & 0 & 0 & -Al^2 & 0 & 0 \\ 0 & 12l & 6Il & 0 & -12l & 6Il \\ 0 & 6Il & 4Il^2 & 0 & -6Il & 2Il^2 \\ -Al^2 & 0 & 0 & Al^2 & 0 & 0 \\ 0 & -12Il & -6Il & 0 & 12I & -6Il \\ 0 & 6Il & 2Il^2 & 0 & -6Il & 4Il^2 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{Bmatrix}. \tag{3.17}$$

Finally, to construct the gloabl matrix, the co-ordinate system for the element stiffness matrix is changed and assembled together with overlapping to avoid repetition to create the 33 by 33 matrix.

3.2 Mass Matrix

Similar to the stiffness matrix, the mass matrix can be derived using a consistent mass matrix which is as shown below.

$$[M^e] = \int_0^l \rho A [N]^T [N] \, dx \tag{3.18}$$

where N is the matrix of the appropriate shape functions. ρ is the mass density and A is the cross-sectional area. This matrix is derived from the kinetic energy in the element which is simlar to how stiffness is derived from the strain energy in the element. By using the consistent mass matrix both translational (or linear) momentum and rotational momentum is conserved. Shown below is the derivation of the mass matrix from it's kinetic energy equation. Using the hermite shape functions

$$H = [H_1 \; H_2 \; H_3 \; H_4], \tag{3.19}$$

the consistent mass matrix shown in equation (3.20) for the 2 dof (v and θ) can be evaluated. With $N = H$, the mass matrix equation can be modified as follows:

$$M^e = \int_0^l \rho A[H]^T[H]\, dx. \tag{3.20}$$

After substiting the variables into the equation and calculating the mass matrix using the equation above, the result is:

$$[M^e] = \begin{bmatrix} 156 & 22l & 54 & -13l \\ 22l & 4l^2 & 13l & -3l^2 \\ 54 & 13l & 156 & -22l \\ -13l & -3l^2 & -22l & 4l^2 \end{bmatrix}. \tag{3.21}$$

However, since the known axial mass matrix is:

$$M_{axial} = \begin{bmatrix} 140 & 70 \\ 70 & 140 \end{bmatrix}, \tag{3.22}$$

using the process of assembling the 2 dof mass matrix, M^e with the axial displacement matrix, M_{axial}, the mass matrix for the 3 dof frame is:

$$[M^e] = \frac{\rho Al}{420} \begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & 22l & 0 & 54 & -13l \\ 0 & 22l & 4l^2 & 0 & 13l & -3l^2 \\ 70 & 0 & 0 & 140 & 0 & 0 \\ 0 & 54 & 13l & 0 & 156 & -22l \end{bmatrix}. \tag{3.23}$$

The system matrix is constructed by changing the co-ordinate using the following matrix:

$$\begin{Bmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{Bmatrix} = \begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & s & 0 \\ 0 & 0 & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \bar{u_1} \\ \bar{v_1} \\ \bar{\theta_1} \\ \bar{u_2} \\ \bar{v_2} \\ \bar{\theta_2} \end{Bmatrix}, \tag{3.24}$$

where $c = \cos(\phi)$ and $s = \sin(\phi)$. Equation (3.24) can be further expessed as:

$$\{d^e\} = [T]\{\bar{d^e}\}. \tag{3.25}$$

Using $[T^e]$ to transform the co-ordinate and then overlapping the element mass matrices, the result is a 33 by 33 matrix which is as a result of multiplying the degrees of freedom to the number of nodes.

3.3 Matlab code Analysis

This section focuses on the MATLAB code that implements all the information provided and explained in this chapter. Therefore, first, the important variables which is used all over the MATLAB code is introduced.

```
nel=10;                  % number of elements
nnel=2;                  % number of nodes per element
ndof=3;                  % number of dofs per node
nnode=(nnel-1)*nel+1;    % total number of nodes in system
sdof=nnode*ndof;         % total system dofs
```

These variables are needed for the overall analysis of the mass and stiffness matrices. Next, derive the co-ordinate values of the nodes in terms of the global axis, (x,y). After deriving these values put it in 1 by 11 matrix which represents the 11 nodes. The code below performs these derivations and calculation as follows:

```
% ga means global axis

x(1)=0;     y(1)=0;      % x, y coord. values of node 1 in terms of the ga
x(2)=0;     y(2)=0.2;    % x, y coord. values of node 2 in terms of the ga
x(3)=0;     y(3)=0.4;    % x, y coord. values of node 3 in terms of the ga
x(4)=0;     y(4)=0.6;    % x, y coord. values of node 4 in terms of the ga
x(5)=0;     y(5)=0.8;    % x, y coord. values of node 5 in terms of the ga
x(6)=0;     y(6)=1;      % x, y coord. values of node 6 in terms of the ga
x(7)=0.2;   y(7)=1;      % x, y coord. values of node 7 in terms of the ga
x(8)=0.4;   y(8)=1;      % x, y coord. values of node 8 in terms of the ga
x(9)=0.6;   y(9)=1;      % x, y coord. values of node 9 in terms of the ga
x(10)=0.8; y(10)=1;      % x, y coord. values of node 10 in terms of the ga
x(11)=1;    y(11)=1;     % x, y coord. values of node 11 in terms of the ga
```

Next, define another set of variables that will be used throughout the derivation process. Below is the code that contains these new variables. The variables are part of the formula needed in the numerical calculation of the mass and stiffness matrices.

```
el=100*10^9;          % elastic modulus (material property)
area=0.0001;          % cross-sectional area
xi=8.3333*10^(-10); % moment of inertia of cross-section
rho=1000;             % mass density per volume (material property)
```

Now, define the boundary conditions. There are 3 boundary conditions, one for each degree of freedom. Below are the boundary conditions.

```
bcdof(1)=1;           % transverse deflection at node 1 is constrained
bcdof(2)=2;           % axial displacement at node 1 is constrained
```

```
bcdof(3)=3;          % slope at node 1 is constrained
```

Thus, all necessary variables for the calculation have been defined. Now for the calculations, below the system stiffness and mass matrices are assembled.

```
kk=zeros(sdof,sdof);       % initialization of system stiffness matrix
mm=zeros(sdof,sdof);       % initialization of system mass matrix
index=zeros(nel*ndof,1);   % initialization of index vector
```

The *index* variable intializes a 30 by 1 vector of zeros which serves as a place holder for the 6 dof per element in the frame. Below is the code where most of the calculations are made.

```
for iel=1:nel    % loop for the total number of elements

index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with element

node1=iel;      % starting node number for element 'iel'
node2=iel+1;    % ending node number for element 'iel'

x1=x(node1); y1=y(node1); % x and y coordinate values of 'node1'
x2=x(node2); y2=y(node2); % x and y coordinate values of 'node2'

leng=sqrt((x2-x1)^2+(y2-y1)^2); % length of element 'iel'

if (x2-x1)==0;  % compute the angle between the local and global axes
beta=pi/2;
else
beta=atan((y2-y1)/(x2-x1));
end

[k,m]=feframe2(el,xi,leng,area,rho,beta,1); % compute element stiffness matrix

kk=feasmbl1(kk,k,index); % assemble element matrices into system matrix

mm=feasmbl1(mm,m,index); % assemble element mass matrices into system matrix

end
```

The goal of this loop is to obtain the 6 dof in an element of the system. This changes the *index* matrix into a 1 by 6 matrix for each element in the system. Also, this matrix changes 10 times to represent each element of the system. It defines $node1$ and $node2$ for the two nodes in an element. Next, it uses the position of $node1$ and $node2$ to determine the value of the rotational deflection function, one of the degrees of freedom. After that, this line: $[k,m] = feframe2(el,xi,leng,area,rho,beta,1)$ uses the function $feframe2$ to derive the element stiffness and mass matrix. This function uses the element matrices derived in sections 3.1 and 3.2 and substitutes the constant variables defined both in the problem statement and the code. Then, the next function, $feasmbl1$ assembles the system mass and stiffness matrix by taking in the previously defined system matrices, the element matrices

and the *index* matrix. It is a loop that runs 6 times and assigns each index element to a diagonal entry of the system matrix 6 times and then changes the element of the system matrix by adding it to the diagonal value of the element matrix. Below I have added the $feasmbl1$ function for more clarification of how the system matrix is built.

```
edof=nnel*ndof;

edof = length(index);
for i=1:edof
    ii=index(i);
        for j=1:edof
            jj=index(j);
                kk(ii,jj)=kk(ii,jj)+k(i,j);
        end
    end
end
```

Finally, below is the last piece of code needed to obtain the natural frequency of the frame.

```
[kn,mn]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions

fsol=eig(kn,mn);   % solve the matrix equation and print
fsol=sqrt(fsol)
```

The first line applies the boundary condition on the system matrices, and then takes their eigenvalue. This is because to obtain the natural frequency of an object, the normal procedure to follow is to use the eigenvalue analysis. And then, the last line takes the square root of the eigenvalue to obtain the desired result which is the natural frequency of a frame.

Chapter 4

Mechanical Structure

This chapter focuses on the introduction of a mechanical structure which applies the finite element method to it. The mechanical structure in question here is more complex than the ones discussed in the previous chapters. The model used as the mechanical structure in this chapter is known as FRAME. This FRAME model is a general purpose finite element and optimal structural control program [3]. The model is used to generate the two dimensional frame finite element structure model which is used for testing how a building behaves during an earthquake. See figure 4.1 for frame model. It is a finite element discretization of a frame which subsitutes a building for the purposes of this work. The next step is to derive the frame structure. Thus, to begin, first note some important variables that are known based on the nature of the problem. This structure is a typical six degrees of freedom frame using the finite element method. This means that the degrees

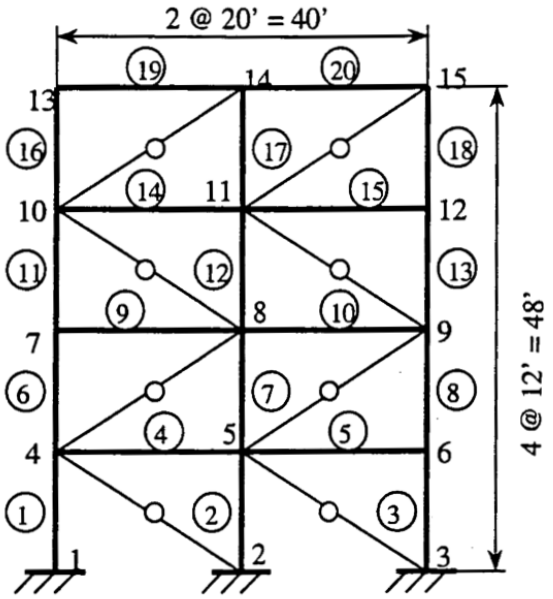

Figure 4.1 Frame Finite Element

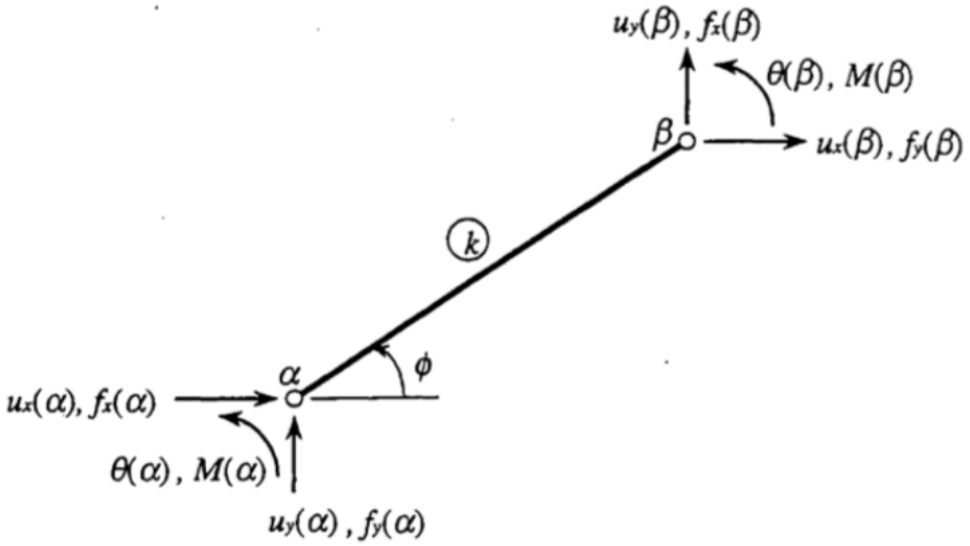

Figure 4.2 Frame Element Degrees of Freedom

of freedom are represented as the following functions:

$$x(i) = \begin{Bmatrix} u_x(i) \\ u_y(i) \\ \theta(i) \end{Bmatrix}, \; f(i) = \begin{Bmatrix} f_x(i) \\ f_y(i) \\ m(i) \end{Bmatrix}, \tag{4.1}$$

where $x(i)$ is the displacement vector, $f(i)$ is the force vector and i represents the nodal points. And the boundary conditions are:

$$\ddot{u_x} = a_x, \; \ddot{u_y} = a_y, \; \theta = 0,$$

where a_x, a_y are the earthquake ground accelerations in the horizontal, and vertical directions respectively [3]. Since the magnitude of the horizontal, x motion is usually greater than the vertical, y motion, they are known as:

$$a \equiv \{a_x \; a_y\} \sim \{a_x \; 0\}. \tag{4.2}$$

See figure 4.2 for degrees of freedom. Using the second law of motion to model the motion of the structure, the result is:

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = Bu(t) + Ef(t), \tag{4.3}$$

where M, C, K are the mass, damping and stiffness matrices, respectively. $x(t)$ is the displacement vector and $Bu(t) + Ef(t)$ are the external variables acting on the Frame. These external variables are; control force and excitation location matrix, respectively. For the purposes of the derivation, let $Bu + Ef = F(t)$. Next, solve for $x(t)$, the displacement of the structure which in turn helps us

find the level and rate of vibrations of this structure. To do this, multiply the equation by M^{-1} to get $\ddot{x}$ alone. Then, function becomes:

$$\ddot{x}(t) + M^{-1}C\dot{x}(t) + M^{-1}Kx(t) = M^{-1}F(t). \tag{4.4}$$

Next, collect like terms and get:

$$\ddot{x}(t) = -M^{-1}C\dot{x}(t) - M^{-1}Kx(t) + M^{-1}F(t). \tag{4.5}$$

Let A be the system matrix which implies $Ax(t) = -M^{-1}C\dot{x}(t) - M^{-1}Kx(t)$. Then, the function is modified as:

$$\ddot{x}(t) = Ax(t) + F(t), \tag{4.6}$$

which can be further simplified as:

$$\ddot{x}(t) = Ax + Bu + Ea. \tag{4.7}$$

Using the change of variables technique, let $\dot{x}(t) = z(t)$ which implies; $\ddot{x}(t) = \dot{z}(t)$. Therefore, the result is:

$$\dot{z}(t) = Az + Bu + Ea. \tag{4.8}$$

Finally, A, B and E can be represented as:

$$z(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}; \; A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \tag{4.9}$$

$$B = \begin{bmatrix} 0 \\ M^{-1}\beta \end{bmatrix}; \; E = \begin{bmatrix} 0 \\ M^{-1}\varepsilon, \end{bmatrix} \tag{4.10}$$

with inital conditions $z(0) = 0$.

Chapter 5

Derivation of the Mechanical Structure

This Chapter, discusses the math model derived in Chapter 4, how to derive the mass, stiffness and damping matrix for this frame in figure 4.1. Using newton's second law of motion as a model, the motion of the frame can be expressed as:

$$M\ddot{x} + C\dot{x} + Kx = Bu + Ef,$$

where M, C, K are the mass, damping and stiffness matrices, respectively. And $Bu + Ef$ are the external variables acting on the Frame. Let $F(t) = Bu + Ef$, so the result is:

$$M\ddot{x} + C\dot{x} + Kx = F(t).$$

The goal in this chapter is to derive the matrices M, C, and K. From the figure 4.1, it can be observed that the frame is more complex than the one discussed in Chapter 2. This frame is made of beams, columns and truss members. Therefore, when applying the derivation techniques acquired from chapter 2 in this chapter there will be a slight modification. Equation (4.1) shows us the generalized form of the displacement and force vector as functions of the nodal points of the frame. Let α and β be the nodal points such that the displacement and force vector can be represented as:

$$x = \begin{Bmatrix} u_x(\alpha) \\ u_y(\alpha) \\ \theta(\alpha) \\ u_x(\beta) \\ u_y(\beta) \\ \theta(\beta) \end{Bmatrix}, \; f = \begin{Bmatrix} f_x(\alpha) \\ f_y(\alpha) \\ m(\alpha) \\ f_x(\beta) \\ f_y(\beta) \\ m(\beta) \end{Bmatrix}. \tag{5.1}$$

Now, let k denote the element of the frame, such that $M(k), K(k)$ and $C(k)$ are the mass, stiffness and damping matrix of the kth element of the frame, respectively. In chapter 2, matrices of a 2 dof frame is defined first and then the 3 dof frame. The same process will be applied in this derivation. However, since these two frames are different there matrices are very different since the frame considered in this chapter is complex. This is further discussed in [7]. Now, let us condider the

2 degrees of freedom matrix derived in chapter 2. Since, the frame is not common there will be a focus on the Yang's suggestion in [7] of a stiffness and mass matrix of an uncommon frame. Therefore, after rigorous readings it is confirmed that the matrices in [3] are in fact the correct matrices for the frame.

Thus, the mass matrix may be represented as:

$$M(k) = \frac{\rho AL}{420}\begin{bmatrix} 156s^2 & -156sc & -22Ls & 54s^2 & -54sc & 13Ls \\ -156sc & 156c^2 & 22Lc & -54sc & 54c^2 & -13Lc \\ -22Ls & 22Lc & 4L^2 & -13Ls & 13Lc & -3L^2 \\ 54s^2 & -54sc & -13Ls & 156s^2 & -156sc & 22Ls \\ -54sc & 54c^2 & 13Lc & -156sc & 156c^2 & -22Lc \\ 13Ls & -13Lc & -3L^2 & 22Ls & -22Lc & 4L^2 \end{bmatrix}, \tag{5.2}$$

and the stiffness matrix:

$$K(k) = \frac{EI}{L}\begin{bmatrix} 12s^2/L^2 & -12sc/L^2 & -6s/L & -12s^2/L^2 & 12sc/L^2 & -6s/L \\ -12sc/L^2 & 12c^2/L^2 & 6c/L & 12sc/L^2 & -12c^2/L^2 & 6c/L \\ -6s/L & 6c/L & 4 & 6c/L & -6c/L & 2 \\ -12s^2/L^2 & 12sc/L^2 & 6s/L & 12s^2/L^2 & -12sc/L^2 & 6s/L \\ 12sc/L^2 & -12c^2/L^2 & -6c/L & -12sc/L^2 & 12c^2/L^2 & -6c/L \\ -6s/L & 6c/L & 2 & 6s/L & -6c/L & 4 \end{bmatrix}, \tag{5.3}$$

Next, derive the damping matrix and obtain:

$$C(k) = \frac{vI}{L}\begin{bmatrix} 12s^2/L^2 & -12sc/L^2 & -6s/L & -12s^2/L^2 & 12sc/L^2 & -6s/L \\ -12sc/L^2 & 12c^2/L^2 & 6c/L & 12sc/L^2 & -12c^2/L^2 & 6c/L \\ -6s/L & 6c/L & 4 & 6c/L & -6c/L & 2 \\ -12s^2/L^2 & 12sc/L^2 & 6s/L & 12s^2/L^2 & -12sc/L^2 & 6s/L \\ 12sc/L^2 & -12c^2/L^2 & -6c/L & -12sc/L^2 & 12c^2/L^2 & -6c/L \\ -6s/L & 6c/L & 2 & 6s/L & -6c/L & 4 \end{bmatrix}, \tag{5.4}$$

where $c = \cos\phi$, and $s = \sin\phi$. Moving the beams in the frame requires a change of coordinate which is why c and s are in the matrices. For instance rotating the frame will cause a sinuosidal motion which these change of coordinates accounts for.

Chapter 6

Numerical Simulation of the Mechanical Structure

This chapter will discuss the MATLAB script that integrates all the discussions from the previous chapters. However, it mostly focuses on integrating chapters 4 and 5. It uses the derived mass, stiffness, damping matrices and other variables to create a MATLAB movie that captures the motion of the frame's vibrations. Below is an introduction to some of the variables in the script.

```
coeffs = 'test';
% coeffs = 'paper'; % vibration invisible

switch (coeffs)
    case 'paper'
        E = 205463.82; % MPa
        A = 75483.72;    % mm^2
        I = 25e8;    % mm^4
        rho = 1;  %lb/in^3
        nu = 500;   %psi-sec
    case 'test'
        E = 200; %205463.82; % MPa
        A = 1; %75483.72;    % mm^2
        I = 1; %25e8;    % mm^4
        rho = 1;  %lb/in^3
        nu = 1; %500;   %psi-sec
end

EA = E*A;
EI = E*I;
rhoA = rho*A;
nuI = nu*I;
nuA = nu*A;
```

We use the first case because the vibration can be seen. The second case vibrates at a rate that is close to invisible to the human eyes so the the first case is used. Next, discretization of the frame

is presented.

```
%Generation of coordinates and connectivities
numberElements = 20;
nodeCoordinates = [ 0 16 32 0 16 32 0 16 32 0 16 32 0 16 32; ...
                    0 0 0 12 12 12 24 24 24 36 36 36 48 48 48; ...
                    zeros(1,15) ];
dofCoordinates = nodeCoordinates'; dofCoordinates = dofCoordinates(:);

elementNodes = [ 1 4;
                 2 5;
                 3 6;
                 4 5;
                 5 6;
                 4 7;
                 5 8;
                 6 9;
                 7 8;
                 8 9;
                 7 10;
                 8 11;
                 9 12;
                10 11;
                11 12;
                10 13;
                11 14;
                12 15;
                13 14;
                14 15];

prescribedDof = [1 2 3 16 17 18 31 32 33]';
numberNodes = size(nodeCoordinates,2);
xx = nodeCoordinates(1,:);
yy = nodeCoordinates(2,:);

ndof = 3*numberNodes;
```

Now, the script below shows a set up of the stiffness matrix and the assembly of its global matrix.

```
% Stiffness Matrix
stiffness = zeros(ndof,ndof);
% computation of the system stiffness matrix
for e = 1:numberElements
    % elementDof: element degrees of freedom (Dof)
    indice = elementNodes(e,:);
    elementDof = [ indice indice+numberNodes indice+2*numberNodes] ;
    nn = length(indice);
    xa = xx(indice(2))-xx(indice(1));
```

```
    ya = yy(indice(2))-yy(indice(1));
    length_element = sqrt(xa*xa+ya*ya);
    cosa = xa/length_element;
    sena = ya/length_element;
    ll = length_element;
    L = [ cosa*eye(2) sena*eye(2) zeros(2);
         -sena*eye(2) cosa*eye(2) zeros(2);
          zeros(2,4) eye(2) ];
    oneu  = [ 1 -1;-1 1 ];
    oneu2 = [ 1 -1;1 -1 ];
    oneu3 = [ 1 1;-1 -1 ];
    oneu4 = [ 4 2;2 4 ];
    k1 = [ EA/ll*oneu zeros(2,4);
           zeros(2) 12*EI/ll^3*oneu 6*EI/ll^2*oneu3;
           zeros(2) 6*EI/ll^2*oneu2 EI/ll*oneu4 ];
    stiffness(elementDof,elementDof) = ...
        stiffness(elementDof,elementDof)+L'*k1*L;
end
```

This loop runs from 1 to 20, assembling the stiffness matrix by element. The *indice* variable obtains the nodes for each element in the frame. Then these nodes are inserted into $elementDof$, the matrix for the dofs per element. Since, there are 3 dofs per element, it assembles a 1 by 6 matrix. Next it finds the distance between each node and uses it to find the angles, *cosa* and *sena*. Then, these angles are used find the L matrix, which is used to change the co-ordinate system of the stiffness matrix for the assembly of the global stiffness matrix. We define the axial displacement matrix, and hence assemble both the element and global stiffness matrix.
Next, assemble the mass and damping matrix. Note, that these matrices are assembled as the stiffness matrix. See code below.

```
% Mass Matrix
massmtrx = zeros(ndof);
for e = 1:numberElements
    indice = elementNodes(e,:);
    elementDof = [ indice indice+numberNodes indice+2*numberNodes] ;
    nn = length(indice);
    xa = xx(indice(2))-xx(indice(1));
    ya = yy(indice(2))-yy(indice(1));
    length_element = sqrt(xa*xa+ya*ya);
    cosa = xa/length_element;
    sina = ya/length_element;
    ll = length_element;
    oneu  = [ 156 54;54 156 ];
    oneu2 = [ -22 13;-13 22 ];
    oneu3 = [ -22 -13;13 22 ];
    oneu4 = [ 4 -3;-3 4 ];
    oneu5 = [ 2 1;1 2 ];
    M = rhoA*ll/420*[oneu*sina^2 -oneu*sina*cosa oneu2*ll*sina;
                     -oneu*sina*cosa oneu*cosa^2 -oneu2*ll*cosa;
```

```
                    oneu3*ll*sina -oneu3*ll*cosa oneu4*ll^2] + ...
                    rhoA*ll/6*[oneu5*cosa^2 oneu5*sina*cosa zeros(2,2);
                    oneu5*sina*cosa oneu5*sina^2 zeros(2,2);
                    zeros(2,6) ];
    massmtrx(elementDof,elementDof) = massmtrx(elementDof,elementDof) + M;
end

% Damping
damping = zeros(ndof);
for e = 1:numberElements
    indice = elementNodes(e,:);
    elementDof = [ indice indice+numberNodes indice+2*numberNodes ];
    nn = length(indice);
    xa = xx(indice(2))-xx(indice(1));
    ya = yy(indice(2))-yy(indice(1));
    length_element = sqrt(xa*xa+ya*ya);
    cosa = xa/length_element;
    sina = ya/length_element;
    ll = length_element;
    oneu  = [ 1 -1;-1 1 ];
    oneu2 = [ -1 -1;1 1 ];
    oneu3 = [ -1 1;-1 1 ];
    oneu4 = [ 4 2;2 4 ];
    C = nuI/ll*[oneu*12*cosa^2/ll^2 -oneu*sina*cosa/ll^2 oneu2*6*sina/ll;
               -oneu*12*sina*cosa/ll^2 oneu*12*cosa^2/ll^2 -oneu2*6*cosa/ll;
                oneu3*6*sina/ll -oneu3*6*cosa/ll oneu4]+...
                nuA/ll*[oneu*cosa^2 oneu*sina*cosa zeros(2,2);
                oneu*sina*cosa oneu*sina^2 zeros(2,2);
                zeros(2,6) ];
    damping(elementDof,elementDof) = damping(elementDof,elementDof) + C;
end
```

Now, that all the matrices are defined, the next step is to set it up in the equation model and solve for x in the equation of motion. However, in the model used, x is Z for this frame, so script below solves for Z.

```
activeDof = setdiff(transpose([1:ndof]),[prescribedDof]);
lethActDof = length(activeDof);
SM = stiffness(activeDof,activeDof);
MM = massmtrx(activeDof,activeDof);
DM = damping(activeDof,activeDof);
epsilon = ones(lethActDof,1);
EE = 0.1*[zeros(lethActDof,1);MM\epsilon];
sss = length(activeDof);
sss3 = sss/3;
factZ = [zeros(sss,sss) eye(sss,sss);-MM\SM -MM\DM];
[T,Z] = solve_vibration(factZ,EE,sss3);
positions = zeros(length(T),ndof);
```

```
for t = 1:length(T)
    positions(t,prescribedDof) = dofCoordinates(prescribedDof);
    for i = 1:length(activeDof)
        positions(t,activeDof(i)) = dofCoordinates(activeDof(i)) + Z(t,i);
    end
end
```

The next piece of code achieves the ultimate goal of creating the movie to display the frame and visualize it's vibrations. See below.

```
vidObj = VideoWriter('peaks.avi');
open(vidObj);

figure
t1 = 1;
t2 = length(T(:,1));
dt = 32;
for tt = t1:dt:t2
    plot(positions(tt,1:numberNodes)',
         positions(tt,numberNodes+1:2*numberNodes)',
         'ro');
    axis([-10,40,-15,65]);

    % Write each frame to the file.
    currFrame = getframe;
    writeVideo(vidObj,currFrame);
end

% Close the file.
close(vidObj);

return % end of function
```

This is where the magic happens. It is at this place that the movie is created and all the plots are captured for the length of time the loop iterates. This captured are then played which causes a moving picture, the MATLAB movie. The movie is saved as an avi file called peaks.

Chapter 7

Stochastic Vibrations of Mechanical Structures

This chapter investigates the vibrations of a specific nodal point, 14. To test for an interesting behavior in node 14 (center of the roof) of figure 4.1, monte carlo's formula was adopted for the random premutation of vibrations. Monte carlo simulation is an approximation technique that involves the generation of random variables from a sample to model the unpredictability of known results like earthquake vibrations. This is useful because earthquake excititations are not unknown until after the fact and this calculation simulates the unpredictable behavior of earthquake excititations. We implemented the model in MATLAB and used *ode*45 solver. Due to adaptive time-stepping, for Monte Carlo simulation we interpolated the results in post-processing to constant time intervals.

Therefore, to achieve this, we considered 10% variability of the Young's modulus E, and used Monte Carlo simulation with 10^4 samples. Specifically, we randomly sampled E from a uniform distribution in the range $190-210\,\text{psi}$, and we simulated the motion of the planar structure in the time interval $[0,600]\,\text{s}$. The horizontal displacement of node 14 is shown in figure 7.1. The mean displacement is given by the periodic forcing, and we see that the width of the band given by standard deviation of the displacement increases with time.

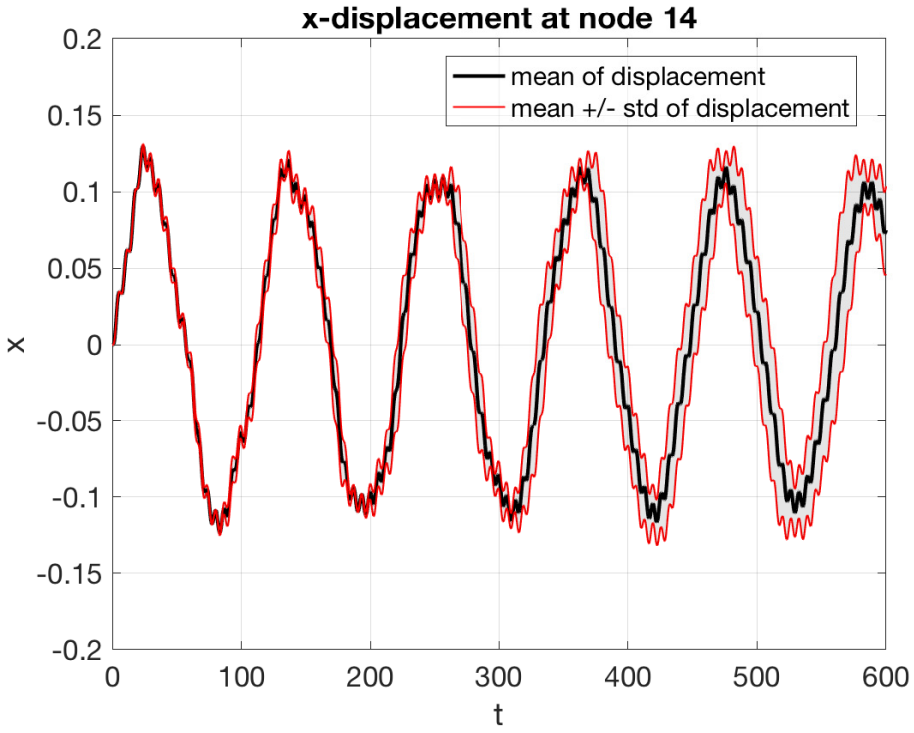

Figure 7.1 Permutation of the horizontal displacement of node14

Chapter 8

Conclusion and Future work

We learned the basics of finite elements and MATLAB programming. Based on our knowledge of elementary differential equations and numerical analysis, we derived and implemented models of vibrations for several mechanical structures. Finally, we also applied our codes in Monte Carlo simulation.

The future research will focus on implementation of active structural control and use of realistic earthquake data for forcing. The goal is to test design, reliability and efficiency of possible structural control mechanisms, both under certainty and uncertainty.

Bibliography

[1] James D. Lee and Siyuan Shen, *Structural Control Algorithms In Earthquake Resistant Design*, Journal of Earthquake Engineering **4** (2000), no. 1, 67–96.

[2] A.F. Bower, *Dynamics and Vibrations MATLAB tutorial*, School of Engineering Brown University.

[3] Kwon and Hyochoong Bang Young W, *The Finite Element Method using Matlab*, second, CRC Press, New York, 2000.

[4] Stanley J. Farlow, *An Introduction to Differential Equations and their Applications*, McGraw-Hill, Inc., New York, 1994.

[5] Oz H.R, *Calculation of the natural frequencies of a beam-mass system using finite element menthod*, Mathematical & Computational Applications **5** (2000), 67–75.

[6] *Real Eigenvalue Analysis*. Chapter 3.

[7] T. Y. Yang, *Finite Element Structural Analysis*, Prentice-Hall International series in Civil Engineering and Engineering Mechanics, New Jersey, 1986.