MQP LDA-1905

Dynamic Analysis and Design: Programming and Applying the Finite Element Method

Jonathan Palczynski

A Major Qualifying Project

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of

the requirements for the degree of

Bachelor of Science

in

Civil Engineering

December 2019

Abstract

Structures in areas prone to earthquakes present unique design challenges and require adherence to detailed seismic design procedures. This project investigates the static and dynamic analysis methods prescribed by ASCE 7-16 for structures in areas prone to earthquakes by completing the structural design of a triangle-shaped five-story building in San Francisco, CA. Three primary seismic analysis methods stemming from ASCE 7-16 are presented namely, the equivalent lateral force analysis method, linear time-history analysis method, and modal response spectrum analysis method. Additionally, computer programs based on the finite element method were developed in Fortran and MATLAB to perform the structural analysis for static and dynamic loading conditions.

Acknowledgments

I have had a unique and enjoyable experience working with Dr. Leonard D. Albano over the past year on this project. Dr. Albano regularly shared his diverse knowledge of structural engineering, and always listened to and provided meaningful feedback on my many ideas. For these reasons and more, I must express my sincerest gratitude to him.

I must also thank Khant Win Htet for his contributions to this project. Htet developed the gravity loads to apply to the structure and gathered the RSMeans cost data for a detailed cost estimate.

Capstone Design Statement

To comply with the accreditation requirements established by the Accreditation Board for Engineering and Technology (ABET), the Department of Civil and Environmental Engineering at Worcester Polytechnic Institute (WPI) requires all Major Qualifying Projects (MQP) to include a Capstone Design Experience that incorporates engineering standards and realistic design constraints. To fulfill this requirement, this MQP completed the structural design of a triangle-shaped, five-story, mixed-use office and residential building subject to seismic loads in San Francisco, CA. The project used ASCE 7-16 seismic analysis methods, namely the equivalent lateral force analysis method, linear time-history analysis method, and modal response spectrum analysis method. Additionally, computer programs based on the finite element method were developed in MATLAB to complete the seismic analysis. Throughout the completion of these items, economic, health and safety, ethical, constructability, and social design constraints were addressed. These constraints are described below.

Economic

A cost estimate of the structural design was completed and broken down into individual components, such as beams, columns, and connections. Different beam and column sizes, column arrangements, and flooring systems were considered to determine the most cost-effective design. Additionally, different seismic force-resisting systems were considered. The recommendation for the final structure considered the estimated cost.

Health and Safety

The structural design followed all applicable specifications and codes to ensure the health and safety of the public. Codes and specifications included ASCE 7-16 [11], AISC Specification for structural steel building [28], and ACI Specification for reinforced concrete [16]. Additionally, because the building supports offices and residential spaces in a location subject seismic loads, it was necessary to assign a proper risk category. Assigning an appropriate risk category ensured the structure was designed to be safe enough for human occupancy.

To comply with health and safety constraints, computer programs based on the finite element method were developed in Fortran and MATLAB to perform the seismic analysis. Software packages such as Ansys, ETABS, and Autodesk Robot allow users to easily perform complex structural analysis procedures with little to no understanding as to how they work. Work that was once performed by engineers with masters and doctoral degrees is now being performed by technicians and engineers possessing only undergraduate degrees. Because of this, results are often accepted blindly or interpreted wrongly [9] [26]. Therefore, for the health and safety of the public, engineers using software to perform seismic analysis must have a fundamental understanding of what they are doing. Blindly accepting or misinterpreting results may lead to a total structural failure in the future.

During the project, ETABS could have been used to avoid spending time gaining a basic understanding of the seismic analysis methods. However, this would go against the health and safety constraints, so the project sought to develop a basic understanding of the seismic analysis methods. An understanding of the methods was gained during the development of structural analysis programs. During the development of the programs, all steps of the equivalent lateral force analysis, linear time-history analysis, and modal response spectrum analysis were examined. It also required examining various types of finite elements, investigating the construction of finite element models, and other considerations.

Ethical

During the design process, the code of ethics provided by the National Society of Professional Engineers (NSPE) was followed. The building is in an area subject to seismic loads, and because of this, design procedures beyond the scope of those required for a traditional static analysis had to be completed. While this required additional work, and ultimately yielded heightened material costs, compromising the overall quality and safety of the structure for cost efficiency was avoided.

Constructability

Constructability was prioritized during the development of the building design. Efficient constructability included using standard and readily available beam and column sizes, and repetitive column spacing.

Social

While this project was mainly aimed at designing the structural skeleton of the building, the proposed building is planned to be used for a mixed-use office and residential space. The proposed

building comprises small- and medium-sized office spaces, small shop spaces, and residential apartments for different types of social backgrounds. Column locations accounted for the spatial layout to ensure that adequate and functional offices and apartments could be implemented into the building.

Sustainability

Sustainable structures are designed to have minimal deterioration from adverse events such as earthquakes. In the event of deterioration, a sustainable structure can recover quickly; that is, the structure is *highly resilient* to deterioration. The structure in this project was designed to withstand loads during extreme events without undergoing total structural failure. During an extreme event, the yielding of structural members could occur, and members may need to be replaced. Replacing members is significantly quicker than replacing an entire structure that has undergone total structural failure. This scenario attests to the high resilience of the structure to deterioration.

Professional Licensure Statement

When designing a product, such as an electronic device or living space, it is necessary and essential to consider the health and safety of the public. Engineers are required to train rigorously to a level where they are deemed competent enough to design and review a product. For civil engineers, designing or constructing a building comes with a significant amount of risk, and the necessary training requires years of academic knowledge and work experience. To mitigate this risk, the National Society of Professional Engineers (NSPE) specifies a standard for becoming a licensed professional engineer (PE).

To become a PE, one must finish a four-year engineering program from an accredited university, pass the Fundamentals of Engineering (FE) exam, complete four years of work experience under a PE, and pass the PE exam. Once an engineer becomes a PE, they have more authority as well as responsibility. Because a PE can design, review, and approve projects, following ethical guidelines is a prime responsibility.

This capstone design project is strictly related to the professional practice of structural engineers. The project includes structural member sizing, structural analysis, and preliminary design drawings. The final structural design would need the approval of a professional engineer if the project ever came to reality.

Contents

Note: Additional appendices are attached as a separate file.

List of Figures

List of Tables

1. Introduction

Earthquakes are a source of enormous amounts of destruction in various locations all around the world. The 2010 Haiti earthquake caused approximately 316,000 deaths and destroyed hundreds of thousands of homes. In the United States, the 1994 Northridge earthquake killed 57 and caused an estimated $40 billion in damage [29]. To minimize monetary damage and loss of life, structures in areas prone to earthquakes must withstand them without undergoing total structural failure. However, this can be an impossible task since future earthquakes cannot be predicted with 100% certainty. Even so, unique seismic structural analysis methods have been created for these areas. These analysis methods include the equivalent lateral force analysis method, linear time-history analysis method, and modal response spectrum analysis method.

The project goal aimed to learn about the aforementioned analysis methods through the design of a triangle-shaped building for gravity and seismic loads. The project consisted of three areas culminating in the design of the triangle-shaped building. First, structural analysis computer programs based on the finite element method for static and dynamic loading conditions were developed in Fortran and MATLAB. Following this, the structural analysis and design of the building was completed through the application of the fully developed computer programs. Upon completion of the structural analysis and design, a cost estimate of the building was prepared using RSMeans data.

It must be noted, that while the MATLAB code included in this report will produce figures with identical data to the ones presented herein, the data generated by MATLAB was post-processed using TikZ and PGF for ease of manipulation and graphical display within this report. TikZ and PGF are languages for producing vector graphics.

2. Background

This chapter presents an overview of the background information needed to understand the significant aspects of the project. Section 2.1 explains the overall structural design process and the engineering standards that were used. Section 2.2 discusses the main components of the finite element method for structural analysis. Section 2.3 presents the seismic analysis methods used, namely the equivalent lateral force analysis method, linear time-history analysis method, and modal response spectrum analysis method.

2.1 Structural Design and Standards

The structural design of a building is divided into two parts, the superstructure design and the foundation design. The superstructure design consists of the column layout and framing system. The column layout must account for the ease of use and functionality of the spatial layout. The spatial layout is the floor plan, and the arrangement of furniture, cubicles, counters, equipment, and other items located within the floor plan. If columns are placed solely to increase the structural integrity of a building with no consideration for the spatial layout, the output may be an unusable building. The superstructure design also consists of sizing the beams, columns, and framing connections. The foundation design consists of sizing footings, the foundation wall dimensions, and other components of the foundation. The superstructure design and foundation design follow different design standards depending on the location of a building.

The American Society of Civil Engineering Specification 7-16 (ASCE 7-16) is a common building design standard used in the United States. ASCE 7-16 specifies methods for calculating design loads to apply to a structure. Design loads include gravity loads, live loads, wind loads, seismic loads, and snow loads, among others. For seismic design, it is vital to follow ASCE 7-16 as it contains updated specifications regarding seismic loading criteria and seismic design methods. Many locations have transferred to or have deadlines set to transfer from ASCE 7-10 to ASCE 7-16. Compared to ASCE 7-10, ASCE 7-16 contains safer, and consequently, stricter updated methods. For example, the seismic design loads for tall structures with a Site Class D soil classification have an additional increase of up to 50%. ASCE 7-16 also removed the provision that allowed for an 18% reduction in seismic design loads for tall structures. Updated ground motion maps and soil coefficients have also contributed to the increase in seismic design loads.

2.2 The Finite Element Method

The finite element method is a procedure for approximating differential equations that govern a variety of engineering problems. An approximate solution is obtained by first discretizing a continuum system into a finite set of points (nodes) connected by *finite elements*. The continuous (field) variable in the differential equation is then approximated in terms of the nodal values of the finite element through interpolation (shape) functions. It is then possible to integrate the differential equation over the element leading to matrices that represent different properties of the element. In static structural mechanics problems based on the displacement method, elements have stiffness properties, expanding to mass and damping properties in dynamic problems [20] [27] [34].

Since the finite element method is approximate, an error can be introduced into the solution. Increasing the number of finite elements and using higher-order shape functions are two ways for convergence of the finite element solution to the analytical solution (this report considers the analytical solution to be the continuous solution). Increasing the number of finite elements to converge upon the exact solution is known as *mesh refinement*. An example of the second method, higher-order shape functions, is found in the commercial finite element software package Ansys. Ansys BEAM188 elements have options for linear, quadratic, and cubic shape functions [21]. A study on the convergence of the finite element solution to the analytical solution is presented in Section 4.3. The study was performed to understand how to create a finite element model that converges to the analytical solution.

2.3 Seismic Analysis Methods

There are three primary seismic analysis methods prescribed in ASCE 7-16. To begin, the seismic response of a structure can be classified as deterministic or non-deterministic. A deterministic response is when the seismic loading characteristics, such as the point of application and magnitude, are known. A non-deterministic response is when the loading characteristics are not known and are defined by some probability, i.e., a random vibration analysis [30]. ASCE 7-16 specifies only deterministic analysis methods: the equivalent lateral force analysis method, linear time-history analysis method, non-linear time-history analysis method, and modal response spectrum analysis method. This project did not consider non-linear analysis, and the following sections cover the equivalent lateral force analysis, linear time-history analysis, and modal response spectrum analysis.

2.3.1 Equivalent Lateral Force Analysis

The simplest seismic analysis method, the equivalent lateral force analysis, is discussed first. Equivalent lateral force analysis is often used for the preliminary design of a structure, and in many cases, can be used for the final design provided the structure does not have many irregularities. The end

goal is to obtain a set of static design forces to apply to the structure. Design forces are used to generate the seismic response, i.e., the displacements and stresses that would occur when a structure is undergoing an earthquake. Design forces are necessary since earthquakes have no direct contact with structures; the effects felt by a structure are generated by inertia. Thus, the design forces are *equivalent* to the inertial forces felt by a structure during an earthquake. Once the earthquake design forces are found, combination with other ASCE 7-16 loads, such as gravity loads, is possible.

Design forces are calculated as a seismic base shear to distribute along the height and width of a structure. The seismic base shear is first used to determine story forces to distribute based on the seismic weight of each story. The story forces are then distributed horizontally at each story relative to the lateral stiffness of the vertical resisting elements and the diaphragm. The seismic base shear V is calculated as

$$V = C_s W \tag{2.1}$$

where W is the effective seismic weight of the structure and C_s is a seismic coefficient dependent on the fundamental period of a structure. The effective seismic weight considers the weight of the structure, along with objects tied to the dynamic properties of the structure. For example, dead loads within a structure will contribute to the inertial forces felt by the structure and thus are considered part of the effective seismic weight. On the other hand, live loads are generally not considered, assumed to be flying through the air during an earthquake and not contributing to inertial forces felt by the structure. The seismic coefficient is used to account for possible material non-linearity and to provide improved performance in high-risk structures [11].

The seismic base shear is based on the simplification of a structure to a single-degree-of-freedom system with a fundamental mode containing 100% mass participation. This simplification can lead to erroneous results for irregular shaped structures; hence, ASCE 7-16 only allows this analysis method for regular-shaped structures.

2.3.2 Linear Time-History Analysis

Linear time-history analysis is used to examine the dynamic response of a structure over the entirety of a seismic event. Figures 2.1, 2.2, and 2.3 show the time-history data from the July 5, 2019, Ridgecrest earthquake. The time interval in Figures 2.1, 2.2, and 2.3 was chosen to display the peak ground motion. It is worth mentioning how to obtain this data and other strong-motion data from all around the world. The data presented in Figures 2.1, 2.2, and 2.3 was obtained from the Center for Engineering Strong Motion Data (CESMD). The CESMD publishes strong-motion seismic data from monitoring stations all around the world to an open-access database. This data, in particular, was recorded by the "Christmas Canyon China Lake" ground station. The CESMD is a collaborative effort between the California Geological Survey, U.S. Geological Survey (USGS), and the Advanced National Seismic System (ANSS). One feature of the database is the ability

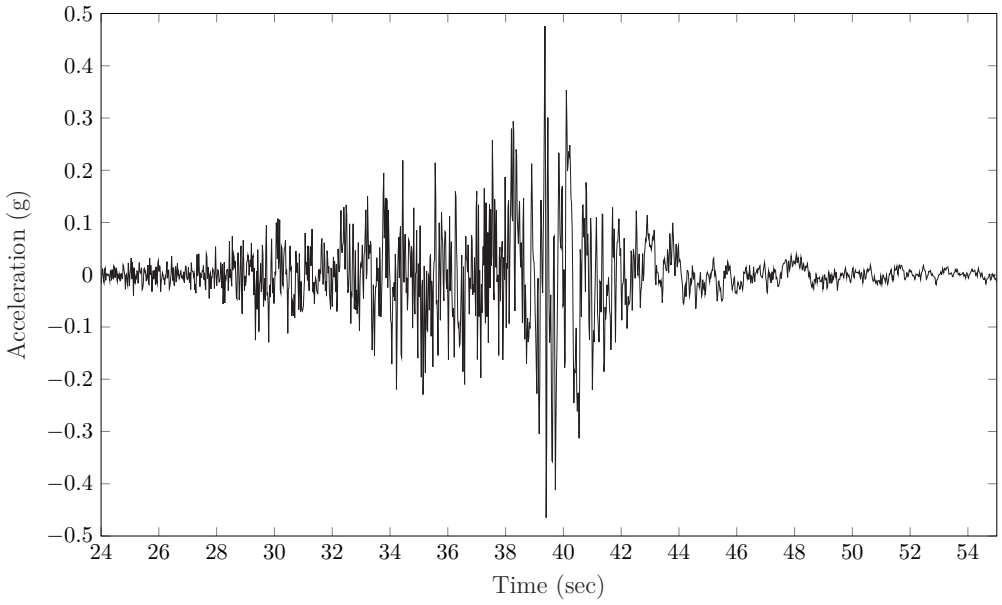

Figure 2.1: 5 July 2019 Ridgecrest Earthquake Acceleration Time-History

to search based on the properties of the monitoring station. This is useful since, in addition to ground monitoring stations, stations are attached to buildings, bridges, dams, and other structures. Searches can be performed for monitoring stations attached to steel high-rise buildings, wooden low-rise buildings, and a variety of different combinations. Certain monitoring stations automatically upload strong-motion data to the database, so it is sometimes possible to obtain the data minutes to hours after an earthquake has occurred. The raw data for Figures 2.1, 2.2, and 2.3 is located in Appendix F, as it was also used for the linear time-history analysis presented later.

Once the time-history data is known, the relative displacements, exact relative velocities, and absolute accelerations of a structure at each point in time can be found. It should be established that this report treats structures as discrete systems, and the following methods apply to discrete systems. For a structure idealized as a single-degree-of-freedom system subject to a ground acceleration $\ddot{\mathbf{u}}_g(t)$, the relative displacement $\mathbf{u}(t)$ of the system, assuming zero initial conditions, can be solved for with the Duhamel integral:

$$\mathbf{u}\left(t\right) = \frac{-1}{\omega\sqrt{1-\zeta^2}} \int_0^t \ddot{\mathbf{u}}_g\left(\tau\right) e^{-\zeta\omega(t-\tau)} \sin\left(\omega\sqrt{1-\zeta^2}\left(t-\tau\right)\right) d\tau, \tag{2.2}$$

where ζ is the damping ratio, and ω is the natural frequency. The Duhamel integral can also be applied to proportionally damped linear multi-degree-of-freedom systems through the modal superposition method. In the modal superposition method, the equations of motion are decoupled by expressing them in terms of an alternative coordinate system. Once the equations of motion are

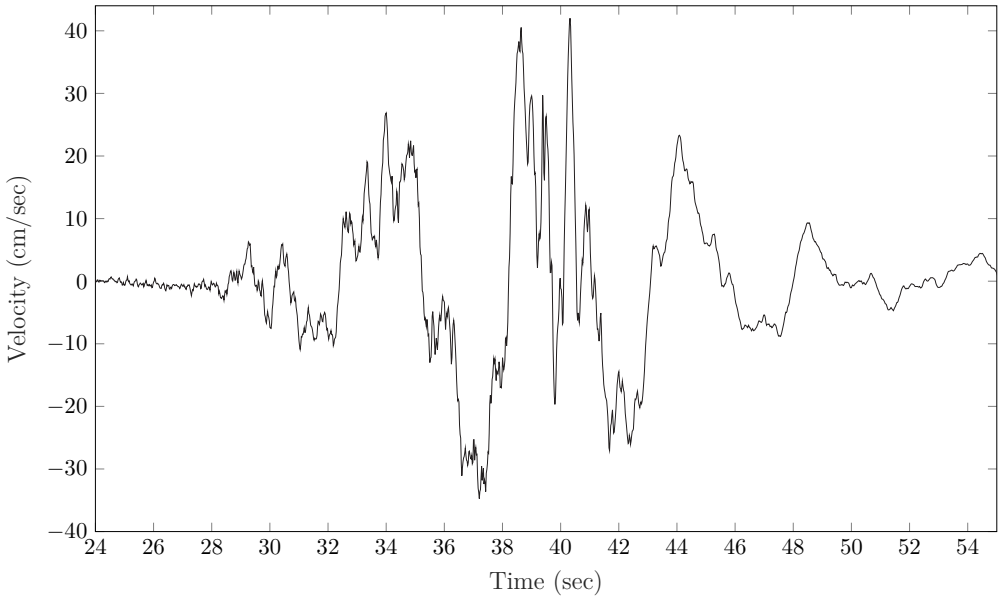

Figure 2.2: 5 July 2019 Ridgecrest Earthquake Velocity Time-History

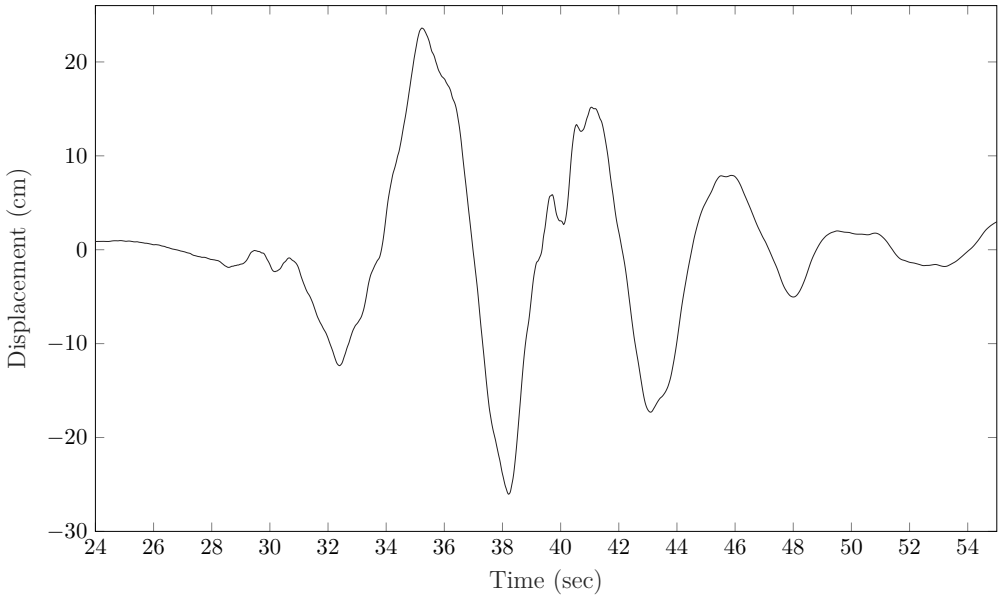

Figure 2.3: 5 July 2019 Ridgecrest Earthquake Displacement Time-History

decoupled, a formerly single system with i degrees of freedom can be analyzed as i single-degree-of-freedom systems, each corresponding to a mode of the structure [24] [13] [30].

While the Duhamel integral and modal superposition method can be used in the case of simple loading, it is not possible to generate a single mathematical function to define the ground acceleration when the acceleration data resembles that of Figure 2.1. For this reason, a variety of step-by-step numerical integration methods for direct integration of the equations of motion (to find the relative displacements, exact relative velocities, and absolute accelerations) exist. These include the Central Difference, Houbolt, Newmark, and Wilson-θ methods and apply to both single- and multi-degree-of-freedom systems [35]. Table 2.1 shows the generalized Newmark method for direct integration of the equations of motion.

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + (1-\gamma)(\Delta t)\,\ddot{\mathbf{u}}_n + \frac{\gamma}{\beta}\frac{1}{\Delta t}\left[\mathbf{u}_{n+1} - \mathbf{u}_n - (\Delta t)\,\dot{\mathbf{u}}_n - \left(\frac{1}{2}-\beta\right)\ddot{\mathbf{u}}_n (\Delta t)^2\right] \tag{2.3}$$

$$\ddot{\mathbf{u}}_{n+1} = \frac{1}{\beta(\Delta t)^2}\left[\mathbf{u}_{n+1} - \mathbf{u}_n + (\Delta t)\,\dot{\mathbf{u}}_n + \left(\frac{1}{2}-\beta\right)(\Delta t)^2\,\ddot{\mathbf{u}}_n\right] \tag{2.4}$$

Table 2.1: Newmark Method

Initial Step	
	1. Input the system stiffness $\mathbf{K}$, mass $\mathbf{M}$, and damping $\mathbf{C}$
	2. Select a time step Δt, and the parameters γ and β
	3. Input the initial displacement $\mathbf{u}_0$ and initial velocity $\dot{\mathbf{u}}_0$
	4. Calculate the initial acceleration: $\ddot{\mathbf{u}}_0 = \mathbf{M}^{-1}(\mathbf{F}_0 - \mathbf{C}\dot{\mathbf{u}}_0 - \mathbf{K}\mathbf{u}_0)$
	5. Calculate the constants
	$a_0 = {}^1/\left(\beta(\Delta t)^2\right)$ $\quad a_1 = ({}^\gamma/_\beta)({}^1/_{\Delta t})$ $\quad a_2 = ({}^1/_\gamma)\,a_1$
	$a_3 = ({}^1/_2 - {}^\beta)/_\beta$ $\quad a_4 = {}^\gamma/_\beta - 1$ $\quad a_5 = \left(\frac{1}{2}-\beta\right)\left(\frac{\gamma\Delta t}{\beta}\right) + (\gamma-1)\,\Delta t$
	$a_6 = \Delta t(1-\gamma)$ $\quad a_7 = \gamma\Delta t$
	6. Form the effective stiffness: $\hat{\mathbf{K}} = \mathbf{K} + a_0\mathbf{M} + a_1\mathbf{C}$
At Each Step	
	1. Calculate $\hat{\mathbf{F}}_{n+1} = \mathbf{M}(a_0\mathbf{u}_n + a_2\dot{\mathbf{u}}_n + a_3\ddot{\mathbf{u}}_n) + \mathbf{C}(a_1 u_n + a_4\dot{\mathbf{u}}_n + a_5\ddot{\mathbf{u}}_n) + \mathbf{F}_{n+1}$
	2. Solve $\hat{\mathbf{K}}\mathbf{u}_{n+1} = \hat{\mathbf{F}}_{n+1}$
	3. Calculate $\dot{\mathbf{u}}_{n+1}$ and $\ddot{\mathbf{u}}_{n+1}$ with Eqs. (2.3) and (2.4)

The Newmark method can be used in a seismic analysis problem by defining an effective time-dependent loading $\mathbf{F}(t)$, where

$$\mathbf{F}(t) = -\mathbf{M}\ddot{\mathbf{u}}_g(t) \tag{2.5}$$

with $\mathbf{M}$ as the system mass. Eq. (2.5) is then used in the Newmark method presented in Table 2.1.

Once the relative displacements of the structure are found, design forces can be generated to combine with other ASCE 7-16 load combinations. The concept of design forces is the same as described in Section 2.3.1, except now design forces that generate specific displacements at various time-steps can be calculated and applied to the structure. For a single- or multi-degree-of-freedom system, design forces for any point in time $\mathbf{F}_d(t)$ can be generated by

$$\mathbf{F}_d(t) = \mathbf{K}\mathbf{u}(t) \tag{2.6}$$

where $\mathbf{K}$ is the system stiffness.

2.3.3 Modal Response Spectrum Analysis

Modal response spectrum analysis is used to estimate the maximum possible response of a structure during a seismic event and, from this information, generate design forces to apply to the structure. Because of this, modal response spectrum analysis is only concerned with the largest displacement, largest velocity, and largest acceleration experienced by a structure during an earthquake. These are known as the *spectral displacement*, *spectral velocity*, and *spectral acceleration*.

A response spectrum can be constructed by subjecting a series of single-degree-of-freedom systems with varying natural periods to the time-history of an earthquake and recording the spectral values experienced by each single-degree-of-freedom system. The response spectrum is the plot of all the spectral values versus all the natural periods. In this case, the response spectrum is considered calculated [30]. The Duhamel integral or step-by-step integration methods could be used to calculate it. Figure 2.4 shows the calculated spectral acceleration response spectrum for the acceleration time-history in Figure 2.1, considering various damping ratios. The Newmark integration method was used for the calculation of Figure 2.4, and the computer code written for this is included in Appendix A. A calculated response spectrum can be useful when comparing structures of varying fundamental periods. For example, the spectral displacements experienced by a set of structures (with known fundamental periods) during a specific ground motion can be easily found from the response spectrum. A designer can use this information to gain an understanding of how various building designs perform during an earthquake without needing to complete a lengthy analysis.

The alternative to a calculated response spectrum is a design response spectrum. A design response spectrum is formed based on a combination of past earthquakes to better model what a future earthquake could entail. Compared to a jagged calculated response spectrum, design response spectrums are often smooth, as shown in Figure 2.5. Figure 2.5 shows a design response spectrum formed from the ASCE 7-16 procedure. The following is a summary of how earthquake design forces are calculated once the response spectrum is known.

The first step is to perform modal analysis. Modal analysis examines the response of a structure when it is in free vibration; that is, neglection of damping effects and excitation by external forces.

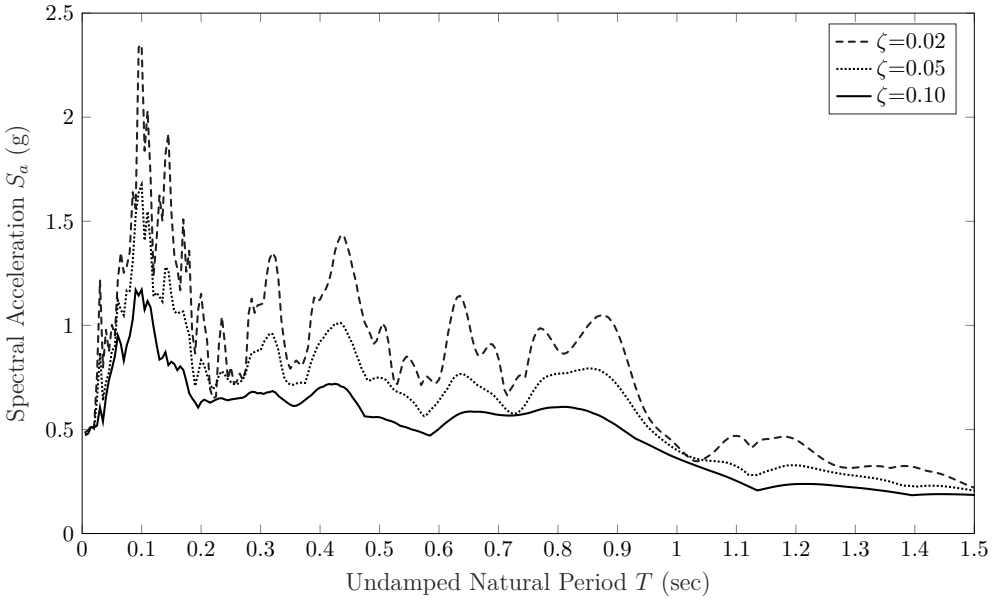

Figure 2.4: Calculated Response Spectrum for Various Damping Ratios

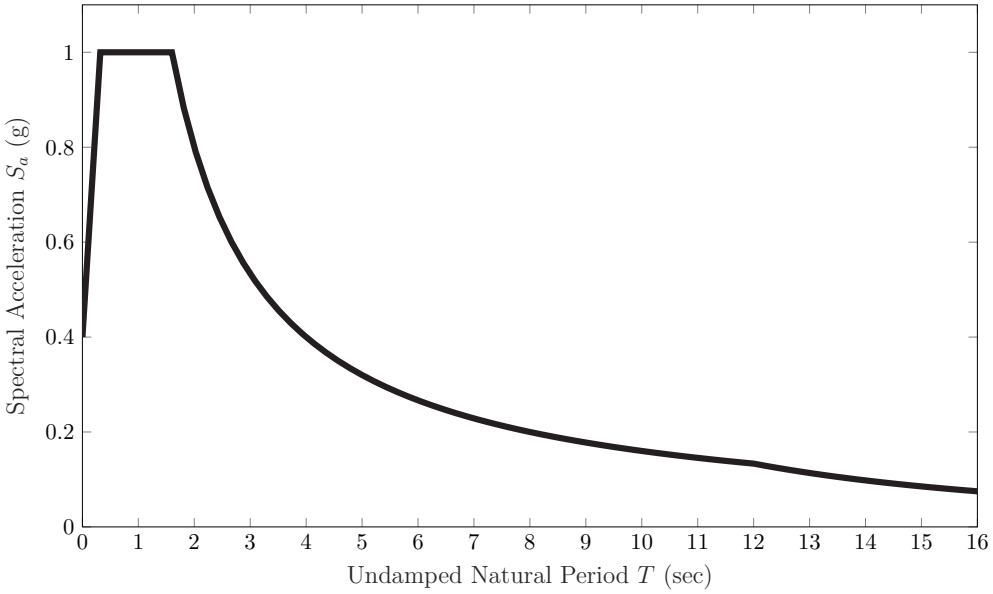

Figure 2.5: ASCE 7-16 Design Response Spectrum

Inducing a structure into free vibration can be done by imparting an arbitrary displacement at some point on the structure and then releasing it. Once in free vibration, the structure will oscillate indefinitely. The frequency of this oscillation is known as the natural frequency of the structure. The deformed shape of the structure when oscillating at this natural frequency is known as its mode shape; hence, modal analysis [32]. There are infinitely many natural frequencies and mode shapes of a continuum structure; however, once the continuum structure is discretized into a finite set of points for analysis, the number of modes computed is equal to the total number of free degrees of freedom of the system.

As an example, Figure 2.6 shows the first four mode shapes of a Euler-Bernoulli (E-B) cantilever beam generated using the finite element method. The cantilever beam in Figure 2.6 is discretized into six elements for a total of seven nodes. Each node has two free degrees of freedom possible (a translation and rotation). Only the end node is fixed, leading to 12 free degrees of freedom total, also meaning that the system is only capable of demonstrating 12 modes. For clarification, Figure 2.6 uses the point normalization convention for displaying the mode shapes. The arbitrary modal displacements are set equal to one at the free end. The deformed shapes of each mode are then scaled accordingly. The computer program written to generate Figure 2.6 is included in Appendix A.

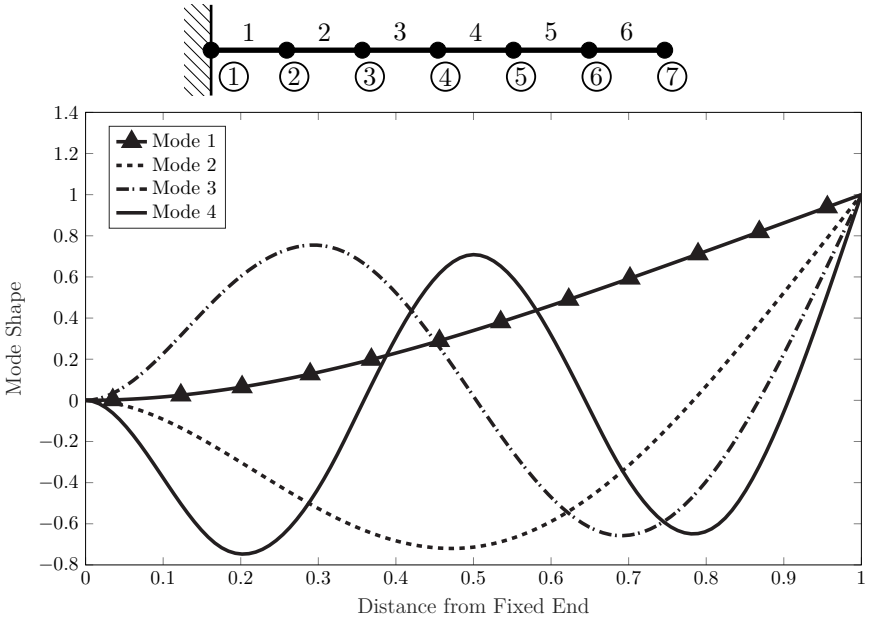

Figure 2.6: Primary Mode Shapes of Cantilever Beam

Modal analysis starts with the governing equation of free vibration for a single- or multi-degree-of-

freedom system:

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{0} \tag{2.7}$$

For a structure with i degrees of freedom, at mode i, the displacement $\mathbf{u}(t)_i$ varies harmonically with time and can be modeled as a sinusoidal function

$$\mathbf{u}(t)_i = \mathbf{\Phi}_i \sin(\omega_i t + \varphi_i) \tag{2.8}$$

where ω_i is the angular frequency, φ_i is the phase angle, and $\mathbf{\Phi}_i$ is the amplitude of free vibration. The amplitude of free vibration vector $\mathbf{\Phi}_i$ is the mode shape of the structure at a given frequency ω_i. Substitution of Eq. (2.8) and the second time derivative of Eq. (2.8) into Eq. (2.7), and removing the $\sin(\omega_i t + \varphi_i)$ term (since the maximum value is one), reduces Eq. (2.7) to the generalized eigenvalue problem:

$$\mathbf{K}\mathbf{\Phi} = \omega^2 \mathbf{M}\mathbf{\Phi} \tag{2.9}$$

Solving the eigenvalue problem amounts to finding the roots (eigenvalues) of the characteristic polynomial and then calculating the eigenvectors. From this analysis, the eigenvalues are the squares of the natural frequencies, and the eigenvectors are the corresponding mode shapes. Solving the eigenvalue problem can be impossible for large systems without the use of numerical methods. Once Eq. (2.9) is reduced to a standard eigenvalue problem (generally possible by completing a Cholesky factorization of the mass matrix [27]); the resulting matrix is often symmetric, and in most cases, a large sparse symmetric matrix. A variety of numerical methods exist to solve this problem, including the the symmetric Jacobi method, symmetric QR method, and symmetric Lanczos method [18] [14] [27].

After the modal analysis is complete, the next step is to calculate the modal displacement vector $\mathbf{u}_i$ for each mode i, where

$$\mathbf{u}_i = \mathbf{\Phi}_i \Gamma_i (S_d)_i \tag{2.10}$$

with Γ_i as the modal participation factor, and $(S_d)_i$ is the spectral displacement corresponding to the natural period of mode i found from the response spectrum. Eq. (2.10) is also considered the process of transforming modal coordinates into physical coordinates. The modal participation factor Γ_i measures how strongly a mode contributes to the response of a structure when the structure is subject to excitation in a specific direction and is calculated with

$$\Gamma_i = \frac{\mathbf{\Phi}_i^T \mathbf{M}\,\mathbf{I}}{\mathbf{\Phi}_i^T \mathbf{M} \mathbf{\Phi}_i} \tag{2.11}$$

where $\mathbf{I}$ is the unit vector. The maximum design forces $(\mathbf{F}_d)_i$ are then calculated for each mode i with

$$(\mathbf{F}_d)_i = \mathbf{K}\mathbf{u}_i \tag{2.12}$$

Eq. (2.12) generates the largest forces since Eq. (2.10) calculates the largest possible displacements. Since it is unlikely that the maximum modal responses for all of the modes will co-occur, a

statistical combination of the modal responses is used to calculate a single modal response. There are different combination methods, including the square root sum of the squares (SRSS) method and the complete quadratic combination (CQC) method [23].

As before, once the seismic design forces are calculated, combination with other ASCE 7-16 loads is possible for application to the structure and member sizing.

3. Methodology

This chapter outlines the objectives of the project and the actions taken to complete each objective. The objectives were broken down loosely based on a systems engineering approach. The application of the systems engineering approach started by completing a hierarchical decomposition of the project (problem) into a series of objectives (sub-problems) [8] [22]: Objectives 1-3. Upon completion of this first-order decomposition, a decomposition within each objective was completed (resulting in second-order decomposed items). By doing this, the project was broken down into a series of logical steps. Completion of the project required completion of the objectives, and completion of the objectives required completion of each second-order decomposed item within the objectives. Further, Objectives 1-3 required sequential completion and each second-order decomposed item within the objectives required sequential completion.

Objective 1: Develop Finite Element Programs
Objective 2: Perform Structural Analysis and Design
Objective 3: Prepare Cost Estimate

The second-order decomposed items of each objective are explained in Sections 3.1 and 3.2 Following Chapter 3, Chapters 4 and 5 correspond to Objectives 1 and 2, and the subsections within each chapter correspond to the second-order decomposed items wherein the results of the completed items are presented.

3.1 Objective 1: Develop Finite Element Programs

Objective 1 was decomposed into four steps: (i) develop an understanding of the finite element method, (ii) define a structure for the finite element programs and write programs, (iii) perform tests on the convergence of the finite element solution, and (iv) create and complete test cases for code verification.

1. Develop an Understanding of the Finite Element Method:
Before the development of the finite element programs, a sufficient understanding of the finite element method was needed for programming the element matrices. There were two possible sequences for approaching this objective, where the sequence refers to the order in which different

types of finite elements were to be learned.

Approach 1: Rod Elements, Beam Elements, Membrane Elements, Shell Elements
Approach 2: Rod Elements, Membrane Elements, Beam Elements, Shell Elements

Approach 1 followed a progression from one-dimensional (1-D) rod and beam elements to two-dimensional (2-D) membrane and shell elements. Approach 2 followed a sequence from elements with solely in-plane deformation (rod and membrane) to elements with both in-plane and out-of-plane deformation (beam and shell). The difficulty of Approach 1 could have manifested with the addition of rotational degrees of freedom as it increases the complexity of member formulation. However, Approach 2 placed the elements needed to model and analyze the building structure last. At a minimum, beam elements were required for a structural model and the structural analysis. Due to this, Approach 1 was chosen. Resources for developing an understanding of the finite element method included [20].

2. Define a Structure for the Finite Element Programs and Write Programs:
Once an understanding of finite elements was gained, the finite element programs were developed. Resources for understanding how to structure and write finite element programs included books [34] [27] [10] and the documentation for NASTRAN (**NA**SA **Str**uctural **An**alysis). NASTRAN is a finite element analysis program initially developed for the National Aeronautics and Space Administration (NASA) in the late 1960s under U.S. government funding for the aerospace industry [3]. Examination of NASTRAN was made possible starting in 2001 when NASA, in collaboration with the Open Channel Foundation (OSC), began releasing the NASTRAN source code and documentation into the public domain [12]. The NASTRAN Programmer's Manual [5], NASTRAN User's Manual [4], and The NASTRAN Theoretical Manual [6] were used for gaining an understanding of the structuring of finite element programs.

Parallel to this, a programming language needed to be chosen for coding the finite element programs. Two options were examined: Fortran and MATLAB. MATLAB is more prominently used in academia, although an understanding of both these languages was desired, resulting in programs being written in both Fortran and MATLAB. The final structural analysis was performed using MATLAB, as this report was completed in an academic setting. Additionally, the various test cases, figures, and studies included in the report were developed in MATLAB, allowing readers to verify the results quickly. The Fortran code developed was still included for readers who may be interested in this and for completeness.

3. Perform Tests on the Convergence of the Finite Element Solution:
As mentioned in Chapter 2, two ways for convergence of the finite element solution are (i) continuous discretization of the finite element model, and (ii) the use of higher-order shape functions. Different types of finite elements were considered for this test. Ultimately a rod and Timoshenko beam

element were chosen. The Timoshenko beam element was chosen for study (i). Timoshenko beam elements use linear shape functions compared to the Hermitian shape functions of a Euler-Bernoulli beam element [20]. When comparing the finite element solution to the analytical solution, the distinction between linear elements and the analytical solution was clear. The rod element was chosen for the investigation of the higher-order shape functions. It was found that the Lagrangian polynomial interpolation formula could be used for the simple generation of higher-order shape functions [17].

4. Create and Complete Test Cases for Code Verification:
Test cases were chosen to highlight the different capabilities of the programs developed. Static and dynamic test cases for beam, shell, and membrane elements were chosen. Test cases were compared with the output of a structural analysis software package, an analytical solution, or a textbook solution. This allowed for a variety of test cases to be performed.

3.2 Objective 2: Perform Structural Analysis and Design

Objective 2 was decomposed into four steps: (i) propose a building design, (ii) develop a structural model, (iii) determine gravity loads and perform preliminary member sizing, and (iv) complete ASCE 7-16 seismic analysis methods.

1. Propose a Building Design:
The proposed building design required following the ASCE 7-16 linear time-history analysis method and the modal response spectrum analysis method. As mentioned in Chapter 2, ASCE 7-16 generally allows for the final design to be based on the equivalent lateral force analysis method provided there are no irregularities. To bypass this, various irregular building designs that would require following the dynamic analysis methods were examined. After proposing a building design, a framing system was chosen. ASCE 7-16 was used as a guideline for selecting an appropriate framing system for a building subject to seismic loading.

2. Develop a Structural Model:
The structural model was developed by subdividing the structure into nodes and assigning element connectivity. Scripts were written for this, as it would not be possible to do manually. The subdivision of the model was split into three parts for storing data. Node locations and element connectivity for joists, girders, and columns were stored distinctly. Storing this data distinctly simplified the process of assigning loads and extracting member forces.

3. Determine Gravity Loads and Perform Preliminary Member Sizing:
Gravity loads were determined by [15] using ASCE 7-16. The governing load combination was chosen from the ASCE 7-16 LRFD load combinations. Scripts were written for assigning gravity

loads to the structure, as it would not be possible to do manually. Once the gravity loads had been defined in the computer model, a linear static analysis using only the gravity loads was performed. From the member force data, member sizes were assigned following the AISC Specification. Performing the gravity analysis before the seismic analysis reduced the number of iterative steps in the seismic analysis. Additionally, knowing the change in member sizes from gravity loading to combined gravity and seismic loading gave quantifiable data for assessing the significance of the change. For example, this data could be used to compare the cost of the building in a non-seismic area versus a seismic area.

4. Complete ASCE 7-16 Seismic Analysis Methods:
Using the preliminary member sizes, the three seismic analysis methods were performed. For each seismic analysis method, an iterative process was followed so as to account for the change in weight of the structure with each member sizing. The final design was chosen based on a single seismic analysis method. The reasons for this are detailed in Chapter 5.

3.3 Objective 3: Prepare Cost Estimate

Cost data was gathered from [15], which used RSMeans 2015. The cost estimate of the building was divided into distinct items: the cost of the substructure and the cost of the superstructure. The substructure included items such as standard foundations and the slab-on-grade. The superstructure included the cost of steel members and exterior walls, among other things. In addition to material costs, items such as installation costs were considered. A final cost estimate was prepared.

Upon completion of the body chapters, Chapters 4-6, a closing chapter composed of conclusions and recommendations, is provided. Limitations of the work are discussed, along with areas for future work.

4. Structural Analysis Using Finite Elements

This chapter presents the development of MATLAB computer programs based on the finite element method to perform the structural analysis for static and dynamic loading conditions. The purpose of this was to gain insight as to how commercial structural analysis software packages work. Gaining this insight was necessary for adhering to the health and safety guidelines outlined in the Capstone Design Statement. The final MATLAB programs were tested by comparing the output of the program to the output of a structural analysis software package, an analytical solution, or a textbook solution.

Section 4.1 presents the matrices developed for finite elements that are commonly used in structural analysis, namely beam and shell elements. Section 4.2 outlines the structure of the MATLAB computer programs. Section 4.3 presents a study performed on the convergence of the finite element solution. Section 4.4 presents test cases using the code written for this project. The bulk of work associated with Chapter 4 is the computer code located in Appendices A and B. The chapter follows a logical progression from the individual element matrices to the complete static and dynamic analysis of multi-element systems.

4.1 Matrices for Finite Elements

In this report, structural elements are considered those that are often used to model buildings, and these include beam and shell elements. This section goes over the formation of the element matrices found in the equation of motion:

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{F\} \tag{4.1}$$

where $[M]$ is the mass matrix, $[C]$ is the damping matrix, and $[K]$ is the stiffness matrix.

4.1.1 Beam Elements

Beam elements, including truss elements, are commonly used to model the structural frame of a building. For determining (beam) member forces, it is acceptable to model a structure only composed of beam elements as long as loads from items such as floors and walls are considered. The

beam element matrices were formed for beam elements based on Euler-Bernoulli (E-B) beam theory. Matrices were formed for beam elements in three-dimensional (3-D) space since the structure was analyzed in 3-D. Frames represented in 3-D space are also known as space frames. Beam elements formed for 3-D space can also be used to analyze frames in 2-D (plane frames) as long as the extraneous degrees of freedom are correctly restrained.

A 3-D beam element can be subject to axial, bending, and torsional loads. In 3-D, the element has six degrees of freedom at each node: three translations in the x, y, and z directions, and three rotations about the x, y, and z axes. For a beam element i, the 3-D stiffness matrix $\mathbf{k}^i$ (as derived in Appendix C) is given by

$$\mathbf{k}^i = \begin{bmatrix}
\frac{EA}{L} & 0 & 0 & \vdots & 0 & 0 & 0 & \vdots & \frac{-EA}{L} & 0 & 0 & \vdots & 0 & 0 & 0 \\
 & \frac{12EI_z}{L^3} & 0 & \vdots & 0 & 0 & \frac{6EI_z}{L^2} & \vdots & 0 & \frac{-12EI_z}{L^3} & 0 & \vdots & 0 & 0 & \frac{6EI_z}{L^2} \\
 & & \frac{12EI_y}{L^3} & \vdots & 0 & \frac{-6EI_y}{L^2} & 0 & \vdots & 0 & 0 & \frac{-12EI_y}{L^3} & \vdots & 0 & \frac{-6EI_y}{L^2} & 0 \\
\cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots \\
 & & & \vdots & \frac{GJ}{L} & 0 & 0 & \vdots & 0 & 0 & 0 & \vdots & \frac{-GJ}{L} & 0 & 0 \\
 & & & \vdots & & \frac{4EI_y}{L} & 0 & \vdots & 0 & 0 & \frac{6EI_y}{L^2} & \vdots & 0 & \frac{2EI_y}{L} & 0 \\
 & & & \vdots & & & \frac{4EI_z}{L} & \vdots & 0 & \frac{-6EI_z}{L^2} & 0 & \vdots & 0 & 0 & \frac{2EI_z}{L} \\
\cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots \\
 & & & \vdots & & & & \vdots & \frac{EA}{L} & 0 & 0 & \vdots & 0 & 0 & 0 \\
 & & & \vdots & & & & \vdots & & \frac{12EI_z}{L^3} & 0 & \vdots & 0 & 0 & \frac{-6EI_z}{L^2} \\
 & & & \vdots & & & & \vdots & & & \frac{12EI_y}{L^3} & \vdots & 0 & \frac{6EI_y}{L^2} & 0 \\
\cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots & \cdot & \cdots & \cdots & \cdots \\
 & & & \vdots & & & & \vdots & & & & \vdots & \frac{GJ}{L} & 0 & 0 \\
 & Sym. & & \vdots & & & & \vdots & & & & \vdots & & \frac{4EI_y}{L} & 0 \\
 & & & \vdots & & & & \vdots & & & & \vdots & & & \frac{4EI_z}{L}
\end{bmatrix} \tag{4.2}$$

where E is the modulus of elasticity, A is the cross-sectional area, L is the beam length, I_z is the moment of inertia about the z-axis, I_y is the moment of inertia about the y-axis, G is the modulus of rigidity, and J is the torsional constant. The formulation of the 3-D beam element stiffness matrix is contained in the ebbeamstiff3D.m function.

The technique used to formulate the mass matrix was considered as it is an essential aspect of dynamic analysis. The mass matrix can differ depending on the technique used to distribute mass at the nodes of an element. Consistent mass matrices and lumped mass matrices are two common ways to form the mass matrix. A consistent mass matrix is formed using the same shape functions as for the stiffness matrix. A lumped mass matrix is formed by assuming the masses are tied to individual degrees of freedom, eliminating cross-coupling, and leading to a diagonal mass matrix [33] [19]. For this project, a consistent mass matrix was formed. For a beam element i, the 3-D consistent mass matrix $\mathbf{m}^i$ is given by

$$\mathbf{m}^i = \rho AL \begin{bmatrix}
\frac{1}{3} & 0 & 0 & \vdots & 0 & 0 & 0 & \vdots & \frac{1}{6} & 0 & 0 & \vdots & 0 & 0 & 0 \\
 & \frac{13}{35} & 0 & \vdots & 0 & 0 & \frac{11L}{210} & \vdots & 0 & \frac{9}{70} & 0 & \vdots & 0 & 0 & \frac{-13L}{420} \\
 & & \frac{13}{35} & \vdots & 0 & \frac{-11L}{210} & 0 & \vdots & 0 & 0 & \frac{9}{70} & \vdots & 0 & \frac{13L}{420} & 0 \\
\cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots \\
 & & & \vdots & \frac{I_y+I_z}{3A} & 0 & 0 & \vdots & 0 & 0 & 0 & \vdots & \frac{I_y+I_z}{6A} & 0 & 0 \\
 & & & \vdots & & \frac{L^2}{105} & 0 & \vdots & 0 & 0 & \frac{-13L}{420} & \vdots & 0 & \frac{-L^2}{140} & 0 \\
 & & & \vdots & & & \frac{L^2}{105} & \vdots & 0 & \frac{13L}{420} & 0 & \vdots & 0 & 0 & \frac{-L^2}{140} \\
\cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots \\
 & & & \vdots & & & & \vdots & \frac{1}{3} & 0 & 0 & \vdots & 0 & 0 & 0 \\
 & & & \vdots & & & & \vdots & & \frac{13}{35} & 0 & \vdots & 0 & 0 & \frac{-11L}{210} \\
 & & & \vdots & & & & \vdots & & & \frac{13}{35} & \vdots & 0 & \frac{11L}{210} & 0 \\
\cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots & . & \cdots & \cdots & \cdots \\
 & & & \vdots & & & & \vdots & & & & \vdots & \frac{I_y+I_z}{3A} & 0 & 0 \\
 & Sym. & & \vdots & & & & \vdots & & & & \vdots & & \frac{L^2}{105} & 0 \\
 & & & \vdots & & & & \vdots & & & & \vdots & & & \frac{L^2}{105}
\end{bmatrix} \tag{4.3}$$

where ρ is the density. The formulation of the 3-D beam element mass matrix is contained in the ebbeamstiff3D.m function.

This project considered classical damping, also known as Rayleigh damping, where the damping matrix is proportional to the mass and stiffness matrices. Upon the formation of the mass and stiffness matrices, the damping matrix $[C]$ was found with

$$[C] = \alpha\,[M] + \beta\,[K] \tag{4.4}$$

where α and β are the proportional damping coefficients, and $[M]$ and $[K]$ are the mass and stiffness matrices [25] [2] [7] [1].

The mass and stiffness matrices in Eqs. (4.2) and (4.3) are presented for beams with rigid ends. If beams have end releases, new mass and stiffness matrices need to be developed. Because the Rayleigh damping matrix is developed from the mass and stiffness matrices, the formulation remains the same. As an example, a beam with a pinned connection at its right end will not be capable of developing a moment at this end connection, but it may still develop a rotation. As a result, additional mass and stiffness matrices are required to model non-rigid end conditions. The code in ebbeamstiff3D.m and ebbeammass3D.m includes the 3-D mass and 3-D stiffness matrices for beams with spherical hinges. Spherical hinges release moments about the x, y, and z axes of a beam element. A variety of other matrices can be formed depending on the desired type of end release. Both ebbeamstiff3D.m and ebbeammass3D.m require an element-type property input. The code allows for four end conditions: rigid, rigid-pinned, pinned-rigid, and pinned-pinned. Additionally, the element matrices in Eqs. (4.2) and (4.3) are in local coordinates, and must be transformed into global coordinates. This transformation is found at the ends of ebbeamstiff3D.m and ebbeammass3D.m, occurring after the complete formulation of the element matrices in local coordinates.

4.1.2 Shell Elements

While shell elements were not used in the final model of the building, code and examples were still developed for shells to gain an understanding of their behavior and simulation. Since shell elements are useful for modeling items such as floors and shear walls, modeling with shell elements provides an accurate representation of the stiffness, mass, and damping of a structure, whereas modeling a structure solely made up of beam elements does not. Shell elements have membrane resistance and bending (plate) resistance. The general shell element formed in this project did not have a *drilling stiffness*, although it is possible to add an artificial stiffness to avoid singular matrices. When performing an analysis with shells, as can be seen in the later test cases, it is possible to constrain the in-plane rotational degree of freedom for every shell element. This prevented singular matrices.

A shell element can be formed by combining a membrane element with a plate element [20]. In this project, a membrane element was first tested and then combined with a plate element to create a shell element. The membrane element has two translational degrees of freedom at each node, while the shell element has six degrees of freedom at each node like the beam element. Q4membranestiff.m contains the stiffness matrix formulation for a membrane element, and Q4shellstiff.m contains the stiffness matrix formulation for a shell element. Shell elements can also be used as membrane elements as long as the appropriate degrees of freedom are constrained. The shell elements in this project were based on the Reissner-Mindlin shell theory, which is thought of as an extension

of the Timoshenko beam theory to shells [20]. The shell elements were chosen to be four-node quadrilateral elements, although there are many more types of elements for consideration.

The stiffness matrix for a shell element is given by [27]

$$\left[k^i\right] = \iint [B]^T [D] [B] \, dx \, dy \tag{4.5}$$

where $[B]$ is the strain-displacement matrix, and $[D]$ is the stress-strain matrix. In the case of a square element, it is easy to evaluate Eq. (4.5) analytically because the shape functions are simple. However, for a generalized quadrilateral element, the shape functions can become complicated and Eq. (4.5) becomes difficult to integrate. Because of this, Eq. (4.5) is converted to a local coordinate system in parametric coordinates. By doing this, Eq. (4.5) can be integrated numerically using Gauss-Legendre quadrature. Using Gauss-Legendre quadrature, the stiffness matrix is computed with [27]

$$\left[k^i\right] = \sum_{i=1}^{nip} W det\,[J] [B]^T [D] [B] \tag{4.6}$$

where nip is the number of integration points, W is the integration point weight, and $det\,[J]$ is the determinant of the Jacobian matrix, also known as *the Jacobian*. Q4membranestiff.m and Q4shellstiff.m both contain the stiffness matrix formulation for a four-node quadrilateral element of arbitrary shape. For simplicity, the code found in Q4membranestiff.m and Q4shellstiff.m sets the default number of integration points to four and contains the coordinates and weights within the function.

4.2 Structure of the Finite Element Programs

The computer programs were structured to have a main file for calling a variety of functions. The functions were used to complete individual tasks, such as forming element matrices and assembling element degrees of freedom. The functions are summarized in Table 4.1 and are referred to as the *black-box* functions. The main files were used to sequence and execute all necessary components for completing an analysis. The main files presented in this report were used for the completion of the test cases in Section 4.4. The full code for the functions and main files is located in Appendix A.

Table 4.1: MATLAB Black-Box Functions

Function	**Description**
ebbeammass3D.m	Returns the mass matrix in global coordinates for a two-node E-B beam element in 3-D with rigid or pinned ends.
ebbeamstiff3D.m	Returns the stiffness matrix in global coordinates for a two-node E-B beam element in 3-D with rigid or pinned ends.
formK.m	Returns the system stiffness matrix assembled from the free degrees of freedom.
formM.m	Returns the system mass matrix assembled from the free degrees of freedom.
Q4membranestiff.m	Returns the stiffness matrix for a four-node quadrilateral membrane element.
Q4mesh.m	Generates a rectangular finite element mesh of four-node quadrilateral elements.
Q4shellmass.m	Returns the mass matrix for a four-node quadrilateral shell element.
Q4shellstiff.m	Returns the stiffness matrix for a four-node quadrilateral shell element.
Q4steering.m	Returns the steering vector for a four-node quadrilateral element and the x, y, and z coordinates of the four nodes. The steering vector contains the status of the degrees of freedom of each node of an element for the assembly of the system matrices.
steering3D.m	Returns the steering vector for a 3-D beam element.

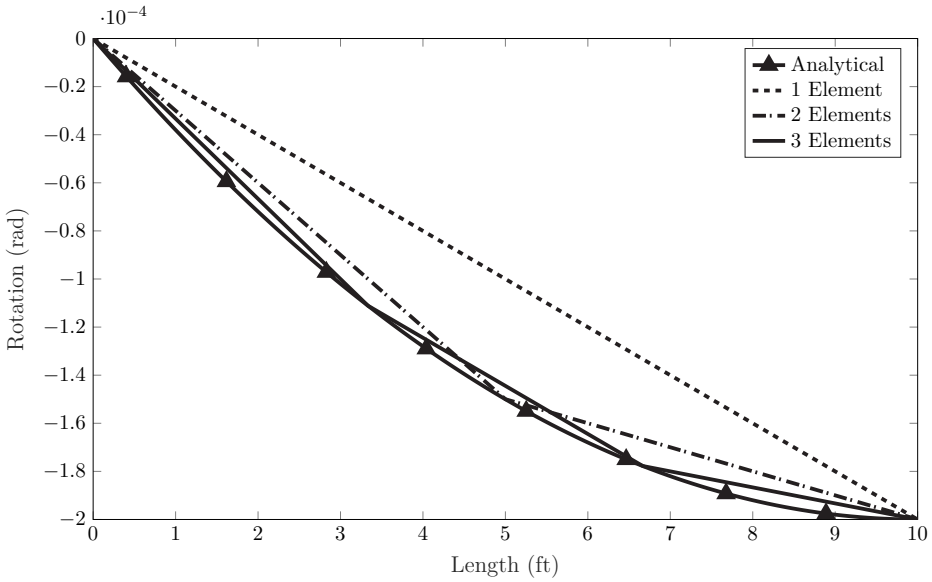

Figure 4.1: Analytical Versus Finite Element Solution for Rotation of a Cantilever Beam

4.3 Convergence of the Finite Element Solution

The finite element method provides an approximate solution to a continuous problem. Because of this, finite element models must be created carefully to ensure the finite element solution converges to the analytical solution. A study was completed to test two methods of the convergence of the finite element solution to the analytical solution. The two methods are (i) continuous discretization of the finite element model and (ii) the use of higher-order shape functions. These two methods are presented in Section 4.3. The coding for the two studies is located in Appendix A.

The first method presented for convergence of the finite element solution to the analytical solution is through continuous discretization of the finite element model, also known as mesh refinement. An example of this is provided in Figure 4.1 by comparing the analytical solution to the finite element solution for the rotation of a Timoshenko cantilever beam subject to a point load as the number of finite elements increase.

The analytical solution was found from the differential equations of a Timoshenko beam. The rotation plots for the finite element solution were calculated by first solving for the nodal rotations and then interpolating the rotations across the element using the element shape functions. The interpolation function used for each element was

$$\theta(x) = N_1\theta_1 + N_2\theta_2 \tag{4.7}$$

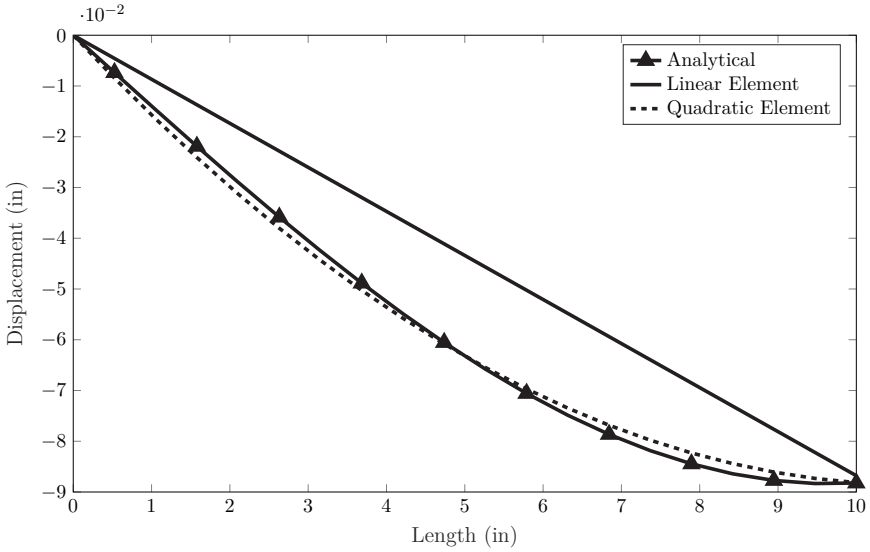

Figure 4.2: Analytical Versus Finite Element Solution for Displacement of a Rod

where θ_1 and θ_2 are the nodal rotations, and N_1 and N_2 are the shape functions given by

$$N_1 = 1 - \frac{x}{L} \tag{4.8}$$

$$N_2 = \frac{x}{L} \tag{4.9}$$

The shape functions for a simple two-node Timoshenko beam element are linear; hence, the finite element solution plots in Figure 4.1 are linear. Figure 4.1 shows that as the number of finite elements increase, the finite element solution converges to the analytical solution. In the case of a single point load, the finite element solution for rotation can converge quickly to the analytical solution. However, as the complexity of the loading increases, the finite element solution will not be able to converge as quickly due to the use of linear shape functions.

The preceding example illustrated the downfall of linear shape functions in the case a member is subject to a complex load. An alternative to increasing the number of finite elements with linear shape functions is to form elements with higher-order (quadratic, cubic, etc.) shape functions. Examples of this are provided in Figures 4.2 and 4.3 by comparing the analytical solution to the finite element solution for the displacement and stress distribution in a rod subject to a distributed load as the degree of the shape function increases.

A simple two-node rod element uses the same linear shape functions as described for the two-node Timoshenko beam element. However, it is possible to formulate a single rod element with more than two nodes through the use of higher-order shape functions, subsequently giving a better

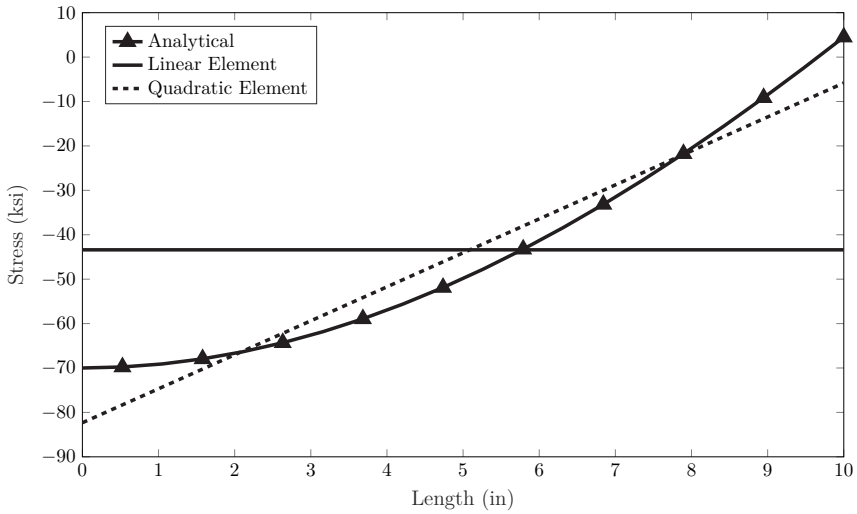

Figure 4.3: Analytical Versus Finite Element Solution for Stress Distribution in a Rod

approximation across the element. For a single rod element with i nodes, the displacement is interpolated across the element as

$$u(x) = N_1 u_1 + N_2 u_2 + N_3 u_3 + \cdots + N_i u_i \tag{4.10}$$

where N_i and u_i are the shape function and nodal displacement of the ith node. The shape functions of order n-1 for an element with n nodes can be generated by the Lagrangian polynomial interpolation formula [17],

$$N_i = \prod_{j=1(j\neq i)}^{n} \frac{x - x_j}{x_i - x_j} \tag{4.11}$$

where x_i and x_j are the node locations. The analytical solutions for the displacement and stress distribution were found from the differential equations of a rod.

It can be seen in Figure 4.2 that the quadratic element can closely match the analytical solution while the linear element cannot. This changes in Figure 4.3 for the comparison of the stress distribution. The finite element solution becomes significantly inaccurate because the originally quadratic element is now interpolating the quadratic analytical solution with a linear function. These examples show that finite element models must be prepared carefully to assure the convergence to the analytical solution.

The application of higher-order shape functions is not limited to beam elements. It is also possible to formulate eight-node quadrilateral elements and so on.

Table 4.2: Test Cases

Function	Description
TestCase1.m	Analysis of a space frame.
TestCase2.m	Analysis of cantilever beam subject to a time-varying point load.
TestCase3.m	Analysis of a beam composed of membrane elements subject to a point load.
TestCase4.m	Analysis of shell elements subject to a point load.

4.4 Test Cases

Four test cases were performed to test the MATLAB code. The test case files are the main files that call the black-box functions described in Table 4.1. The full test case files are located in Appendix A. The test cases are summarized in Table 4.2. Figures in Section 4.4 presenting the result of the finite element solution are considered the *post-processing* of the finite element solution.

Test Case 1:
The first test case was the analysis of a space frame. This program made use of the steering3D.m, ebbeamstiff3d.m, and formK.m functions. The main program file that ties these all together is TestCase1.m. This solution was compared with the solution in [27]. Figure 4.4 shows the scaled deflection plot of the space frame structure.

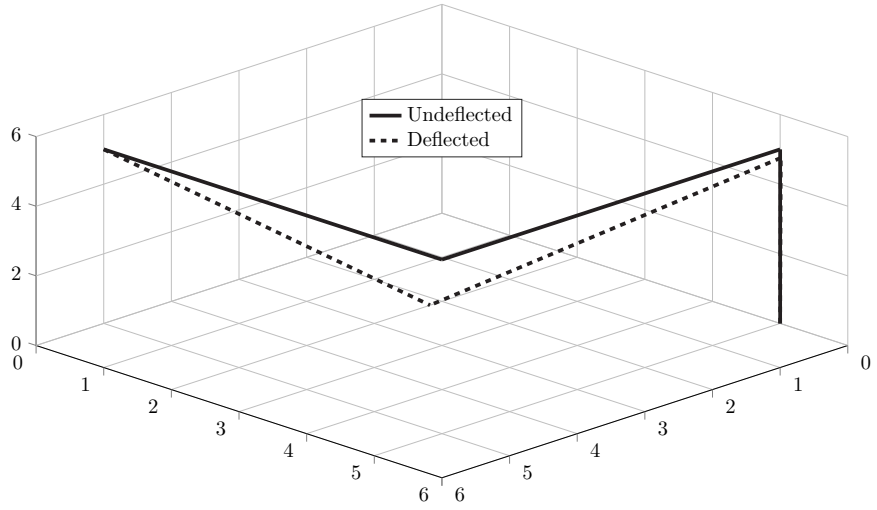

Figure 4.4: Deflection Plot (Scaled 200X) of Space Frame Structure

Figure 4.5: SDOF System Subject to Forced Vibration

Test Case 2:
The next test case was the analysis of a single-degree-of-freedom system subject to a time-varying point load using the Newmark integration method. The single-degree-of-freedom system is shown in Figure 4.5. The time-history of the applied force is shown in Figure 4.6. This program made use of the steering3D.m, ebbeamstiff3D.m, ebbeammass3D.m, formK.m, and formM.m functions. The main program file that ties all these together is TestCase2.m. The Newmark integration method presented in the main program file follows the same algorithm presented in Table 2.1.

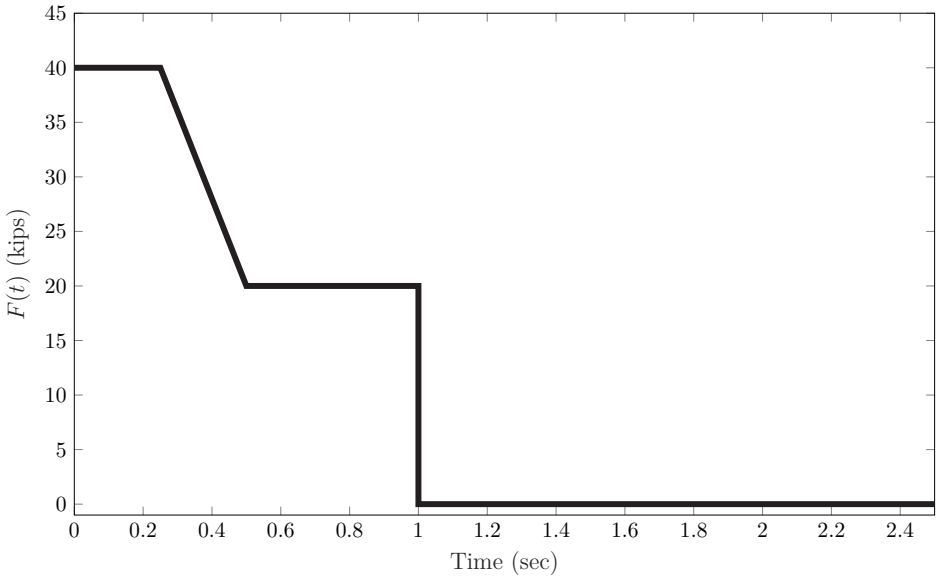

Figure 4.6: Time-History of Applied Force

Figure 4.7 shows that as the force becomes constant at 20 kips, the response of the damped beam eventually becomes a deformed steady-state position, and the undamped beam oscillates about the deformed shape. Once the applied force drops to zero, the response of the damped beam diminishes to the undeformed shape, and the undamped beam oscillates about the undeformed shape. This solution was compared to analytical solutions found in [13].

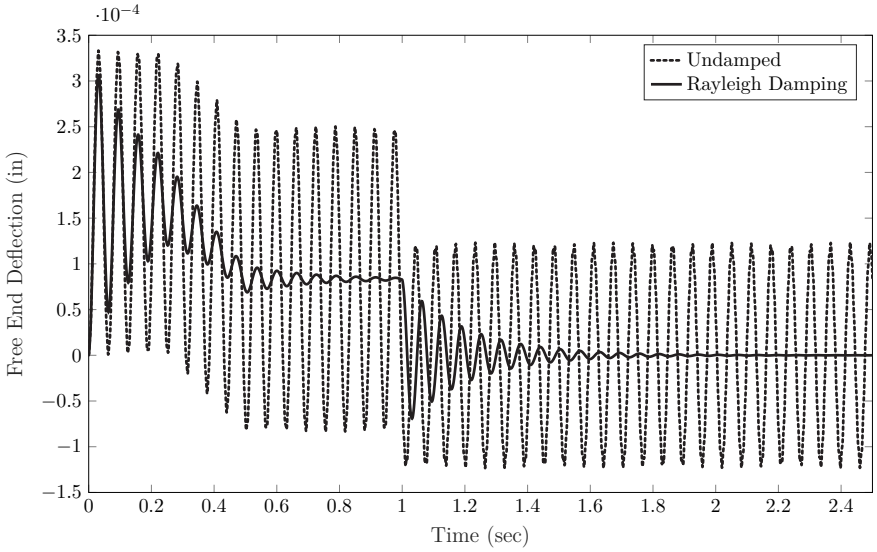

Figure 4.7: Damped Versus Undamped Response to Forced Vibration

Test Case 3:
The third test was subjecting a beam made up of membrane elements to a point load, as shown in Figure 4.8. This program made use of the Q4steering.m, Q4membranestiff.m, and formK.m functions. The main program file that ties all these together is TestCase3.m. The solution was compared to the output of a finite element model made in Ansys.

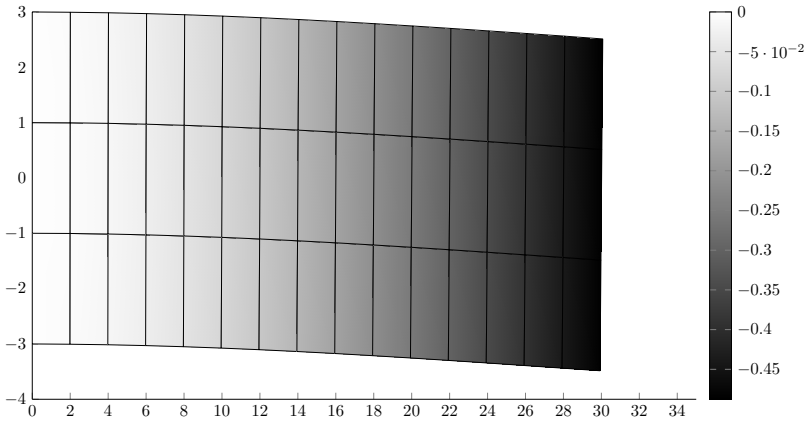

Figure 4.8: Vertical Displacement of Membrane Elements

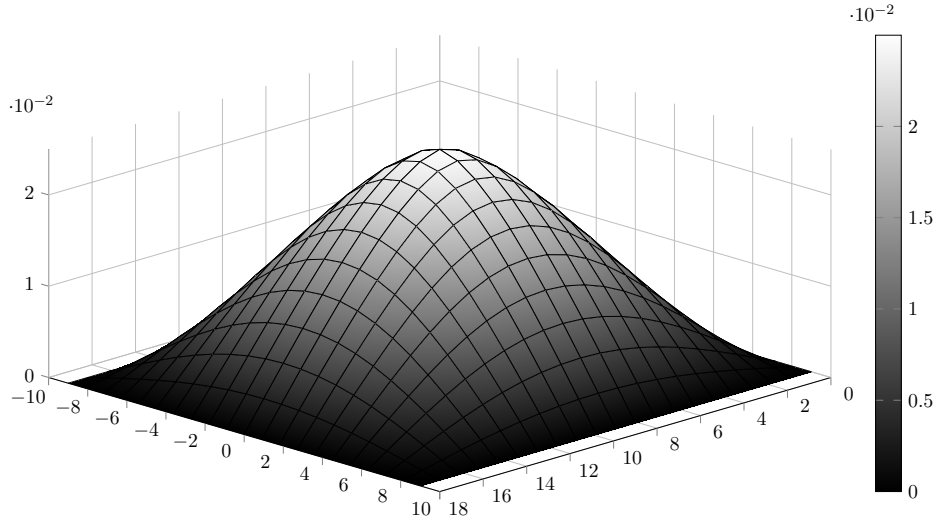

Figure 4.9: Simply Supported Shell Subject to Point Load

Test Case 4:
The fourth test investigated the response of a simply supported shell element subjected to a center point load. This program made use of the Q4mesh.m, Q4steering.m, Q4shellstiff.m, and formK.m functions. All of the previous test cases included the finite element model data within the main program file. This test case differs by calling an automatic mesh generator. The automatic mesh generator creates a rectangular finite element mesh made up of four-node quadrilateral elements. As seen in TestCase3.m, entering the nodal coordinates and element connectivity is very tedious even for a small number of elements. Because of this, automatic mesh generators are often employed. Figure 4.9 shows the deformed plot of the shell subject to a point load. The output of this program was compared to the analytical solution for a shell element found in [31].

In summary, the results from Test Cases 1 through 4 attest to the validity and capability of the code developed for use in this project.

5. Structural Analysis and Design

This chapter presents the structural analysis and design of the triangle-shaped building. The building geometry is shown in Section 5.1, followed by the creation of the structural model in Section 5.2. The structural analysis and design followed an iterative process. First, the building was sized for gravity loads. The gravity loads and preliminary member sizing is presented in Section 5.3. After the initial sizing, the seismic analysis began. The seismic analysis was an iterative process because all methods included the weight of the structure, which changed with changing member sizes. As the weight changed, the dynamic properties of the structure changed, and the analyses needed to be rerun until convergence of the member sizes. The seismic analysis is located in Section 5.4.

5.1 Building Design

Figure 5.1 shows a 3-D view of the final building design. The triangle-building structure has five stories above grade and no basement. All stories are 15 ft in height, aside from the first story, which is 20 ft. Figure 5.2 shows the final plan view design of the building, which is consistent for every story. All stories are 45-45-90-degree triangles in plan view, each with a height and base of 200 ft. The total square footage of each story is 20000 sq-ft. Each story contains 45 20 ft by 20 ft square bays and ten 45-45-90-degree triangle bays, each with a height and base of 20 ft. The girders are spaced at 20 ft, while the joists are spaced at $^{20}/_{3}$ ft. Figure 5.3 shows an elevation view of one of the frames. The columns are spaced at 20 ft since they connect to girders. Spacing columns at 20 ft equally divides the 200 ft base and height. At the corners of the 45-degree angles, the girders are cantilevered. The members along the hypotenuse side of the triangle also serve as girders.

The building made use of moment-resisting connections to resist lateral loads. For simplicity, all girder-to-column connections were considered rigid. All joist-to-girder connections were considered pinned and not able to transfer a torsional moment to the girders. A moment-resisting frame was chosen because it is one of the lateral-force resisting frame systems prescribed in ASCE 7-16.

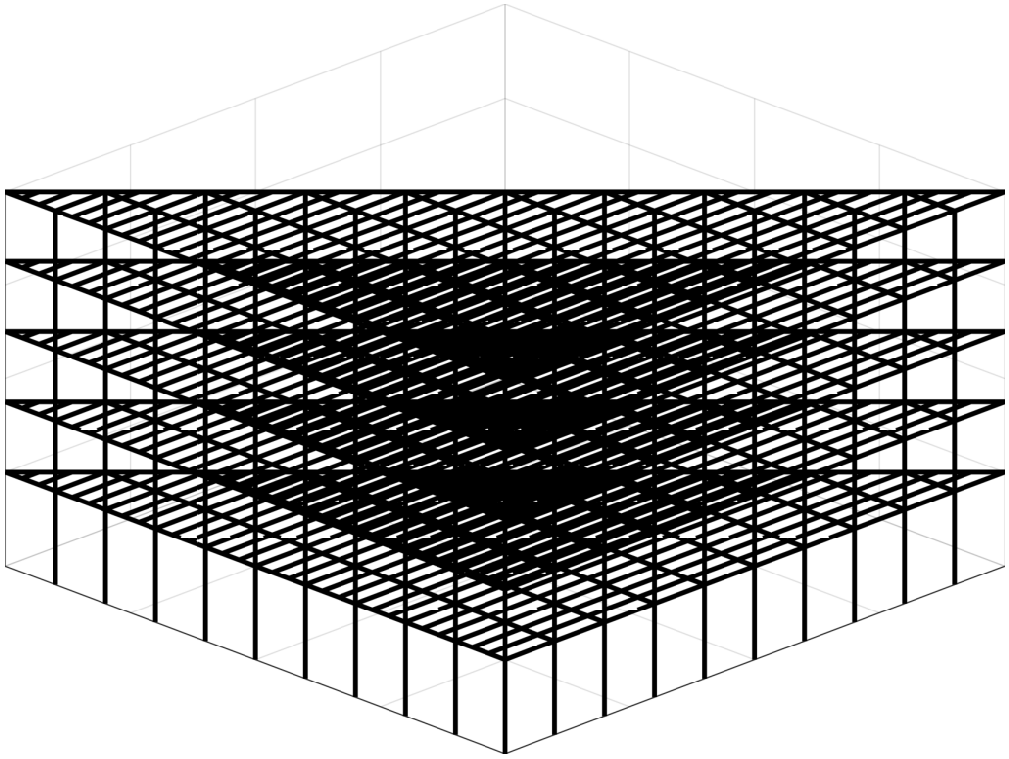

Figure 5.1: 3-D View of Building

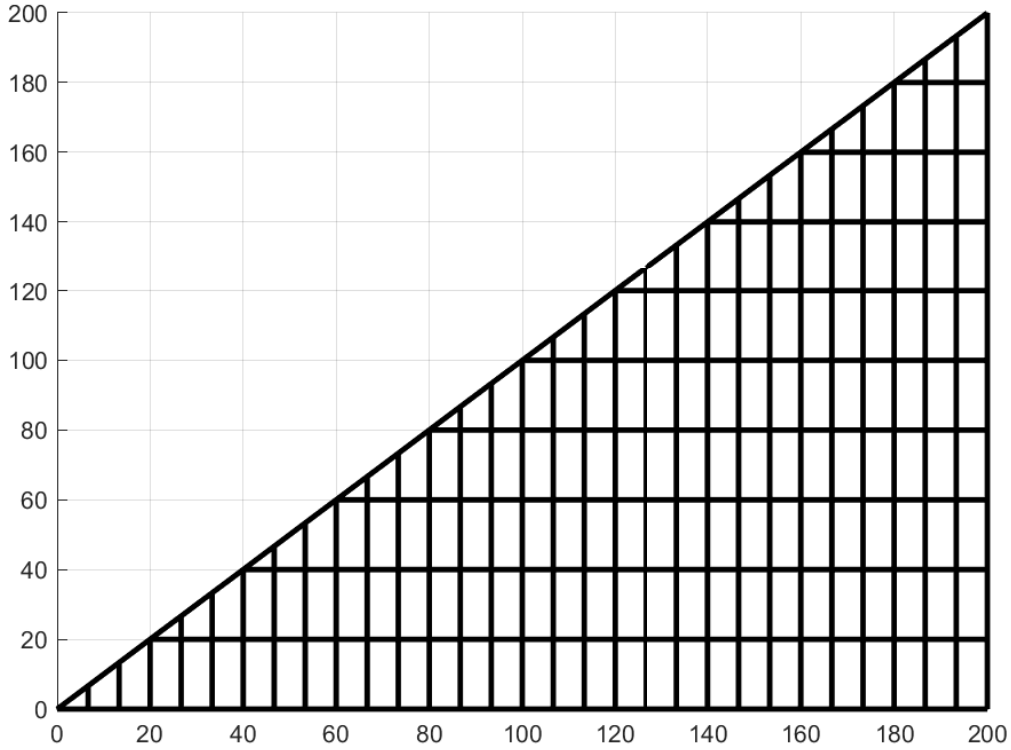

Figure 5.2: Plan View of Building

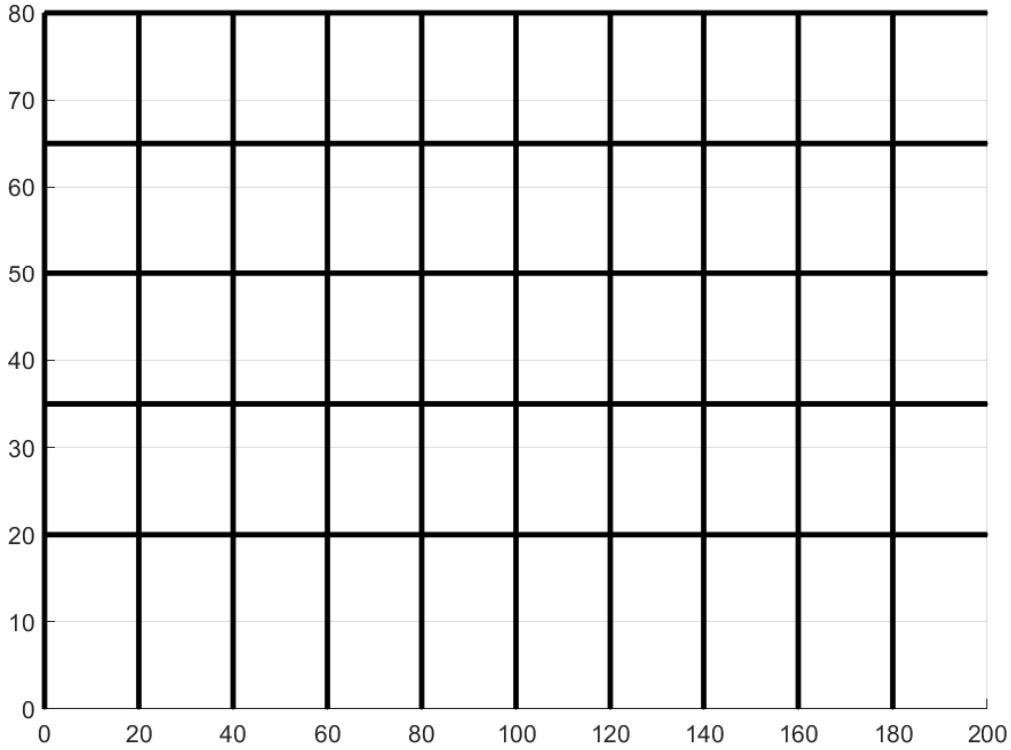

Figure 5.3: Elevation View of Building

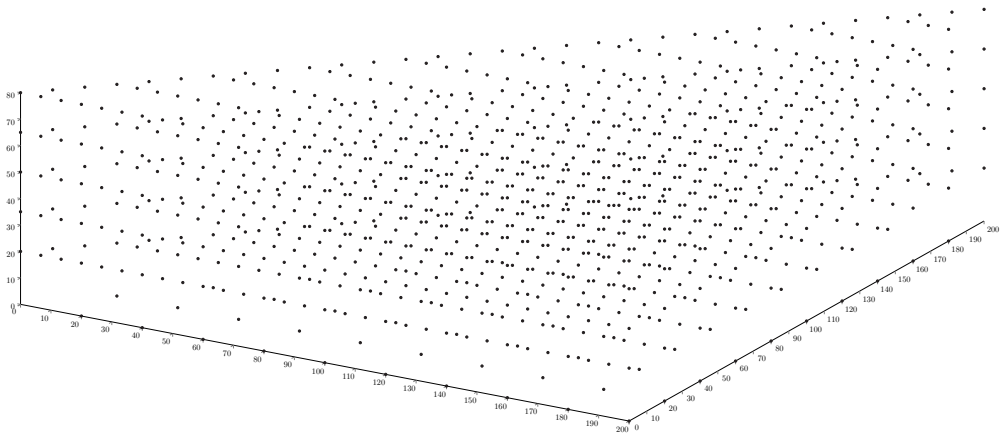

Figure 5.4: Column, Girder, and Joist Connections

5.2 Structural Model

Upon finalizing the building design, a structural model was created of the structure for the structural analysis. The creation of the structural model can be seen as defining a simple finite element mesh. Figure 5.4 shows the node locations of the girder-to-column connections and joist-to-girder connections. In this model, columns, girders, and joists are treated as single elements. Figure 5.5 shows the creation of additional elements by subdividing the columns, girders, and joists into six or more elements. The process is referred to as mesh refinement. Specifically, joists, girders, and columns located in stories two through five now comprise six elements. Due to the height of the first floor being 20 ft, two more elements were used to mesh it; now, the first story columns comprise eight elements. The final model used for analysis contained 7572 nodes for a possible 45432 degrees of freedom. Figure 5.6 shows the total nodes created for the subdivision of joist and girder members, and Figure 5.7 shows the total nodes created for the subdivision of column and girder members.

Appendix G contains the node numbering and node locations. Appendix H contains the element connectivity, element numbering, and element type. The element connectivity is the two end nodes of a member. The element type depends on the end connections of the beam. The choices are rigid-rigid, rigid-pinned, pinned-rigid, or pinned-pinned. For example, a joist to girder connection is pinned, while the intermesh of a joist is rigid.

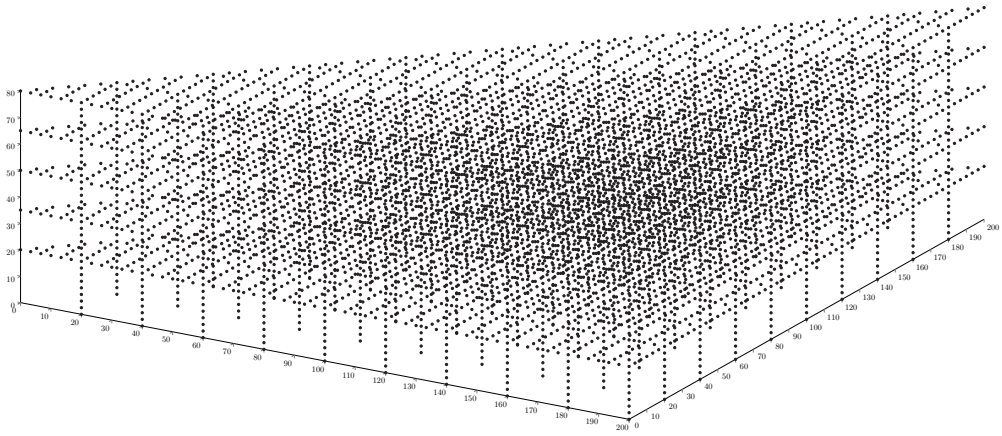

Figure 5.5: Subdivision of Columns, Girders, and Joists

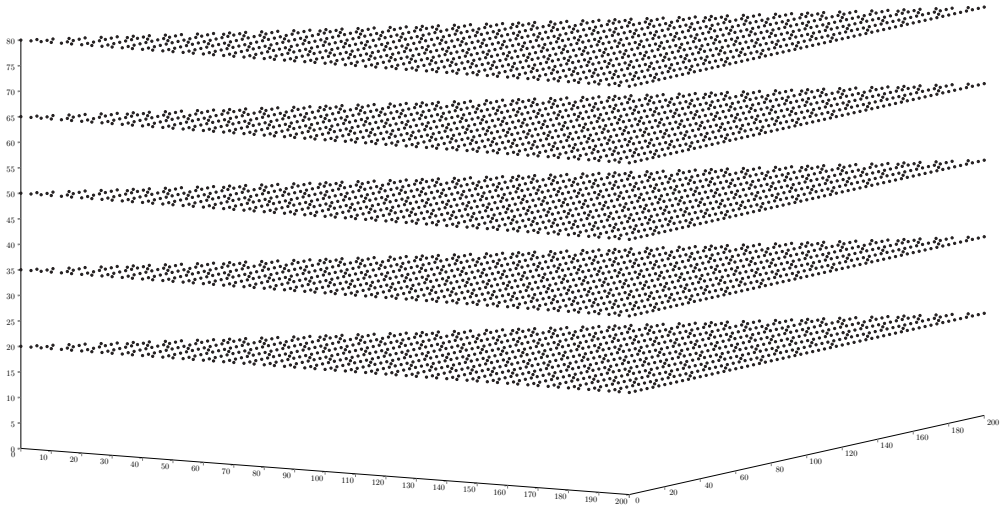

Figure 5.6: Subdivision of Joists and Girders

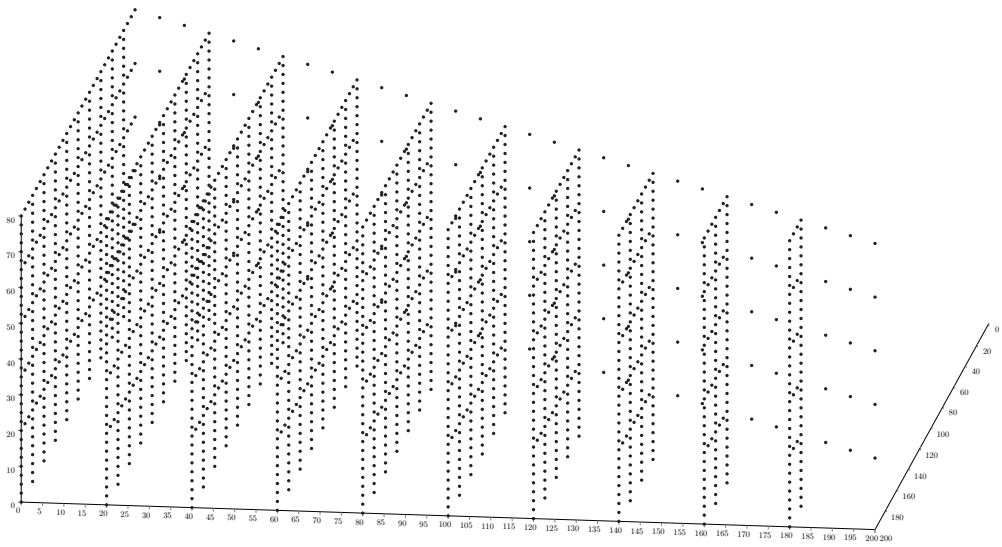

Figure 5.7: Subdivision of Columns and Girders

5.3 Gravity Loading for Preliminary Member Sizes

The governing load combination was calculated according to sections 2.3.1 and 2.3.6 in ASCE 7-16. Since seismic loading was considered, wind loading was neglected. ASCE 7-16 has specifications for both Load and Resistance Factor Design (LRFD) and Allowable Strength Design (ASD). For this project, LRFD load combinations were chosen.

The gravity load resisting system gathers and transfers the floor live loads, roof live loads, the superimposed dead load, and the self-weight of the structural members to the foundation elements. The superimposed dead loads in this building include the exterior wall loads, MEP loads, and flooring loads. The structural system was designed so that loads were uniformly distributed to the joists first, then transferred as girders as point loads, which then transferred the loads to the columns and the footings.

Chapters 3 and 4 of ASCE 7-16 were referenced when considering the gravity load system of the structure. The dead load was divided into the superimposed dead load and the self-weight of the structural members. Since this project mainly focused on the structural design of the building, the superimposed dead loads were assumed a specified combined value, as opposed to individually calculating them. From the building design, the first and second floors serve as office space, and the remaining floors serve as condominium apartments. This was important during the determination of the live loads because the office space and condominium apartments have different uniform live loads, according to ASCE 7-15. Roof live loads were also obtained from ASCE 7-15. The summary of the gravity loading types and corresponding weights are shown in Table 5.1 [15].

Table 5.1: Gravity Loads

Gravity Load Type	Weight (psf)
Office Uniform Live Load	50
Residential Uniform Live Load	40
Roof Live Load	20
Exterior Wall Load	48
Interior Wall Load	48
Ceiling/Electrical/Mechanical	4
Flooring (Concrete on Metal Deck) Load	27

The initial member sizes were sized following AISC 7 using ASTM A992 W-shape beams based on the loads in Table 5.1. Table 5.2 shows the initial member sizes. The initial member sizes were computed to provide a basis for the seismic analysis.The initial member sizes were sized following AISC 7 [28] using ASTM A992 W-shape beams based on the loads in Table 5.1. Load and Resistance Factor Design (LRFD) was used for the design of steel members. Joists and girders were assumed to be continuously braced due to the use of a concrete slab. For joists and girders, the strength limit state was examined first, and the serviceability limit state for deflection was examined second. Columns were designed for combined axial and bending forces. Table 5.2 shows the initial member sizes. The initial member sizes were computed to provide a basis for the seismic analysis.

Table 5.2: Preliminary Member Sizes

Story	Joists	Girders	Columns
1	W14x22	W18x50	W12x96
2	W14x22	W18x50	W12x96
3	W14x22	W18x50	W12x96
4	W14x22	W18x50	W12x96
Roof	W14x22	W18x50	W12x96

5.4 ASCE 7-16 Seismic Analysis Methods

The ASCE 7-16 seismic analysis methods are presented in the order they were completed. The method chosen for the final design was the modal response spectrum analysis. The final member forces used for member sizing are located in Appendix I.

5.4.1 Equivalent Lateral Force Analysis

The iterative equivalent lateral force analysis procedure is outlined in Figure 5.8. Based on the initial member sizing, the seismic base shear was calculated. The seismic base shear was then combined with the other ASCE 7-16 loads through the use of appropriate load factors and applied to the structure. After finding the member forces, the members were resized. Because the seismic weight increased after resizing the members, the seismic base shear needed to be calculated again and applied to the structure with the other ASCE 7-16 loads. This was an iterative process until convergence of the member sizes. Table 5.3 shows the parameters used for the initial equivalent lateral force analysis.

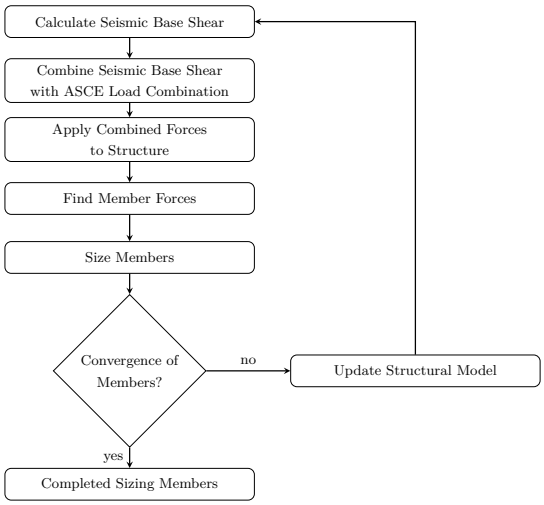

Figure 5.8: ELF Procedure

Table 5.3: ELF Factors

Factor	**Value**
0.2 Sec Spectral Accel, S_s	1.5g
1 Sec Spectral Accel, S_1	0.6g
Response Modification, R	8
Overstrength Factor, Ω	3
Deflection Amplification, C_d	5.5
Occupancy Importance, I	1.25
Site Class	D

5.4.2 Linear Time-History Analysis

The linear time-history analysis was performed based on the July 5th, 2019 Ridgecrest earthquake data. The July 5th, 2019 earthquake was chosen because it occurred recently and was located near San Francisco, CA. Additionally, July 5th, 2019, earthquake was strong enough to produce meaningful results, compared to low magnitude earthquakes that occur daily. The design forces were chosen based on the most significant ground acceleration experienced by the structure.

Figure 5.9 shows the iterative linear time-history analysis procedure that was followed. The structure was analyzed at 15050 acceleration points at .02 sec time steps. The raw data is located in Appendix F. This is the same data that is presented in Figures 2.1, 2.2, and 2.3. Using this data was not very realistic because it was from a recent small earthquake. The required member sizes decreased from the member sizes determined for the equivalent lateral force analysis.

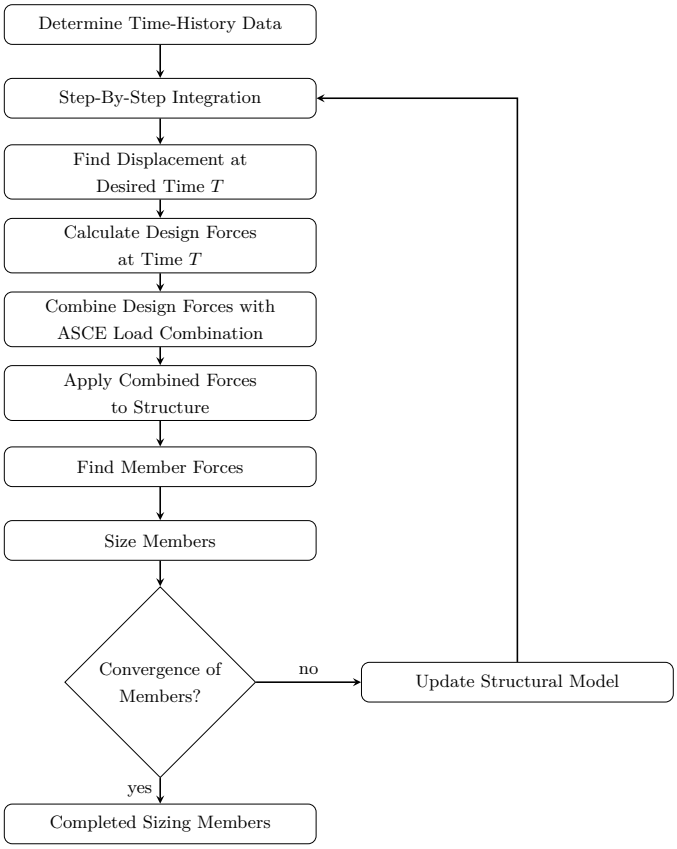

Figure 5.9: Time-History Procedure

5.4.3 Modal Response Spectrum Analysis

The modal response spectrum analysis was the last analysis performed, and its results were chosen for the final design. The equivalent lateral force analysis method is not allowed for the irregular shaped building designed in this project, and thus the final design could not be based on it. However, the equivalent lateral force analysis method was used to gain initial insight as to the response of a structure during an earthquake. For a linear time-history analysis, ASCE 7-16 specifies the structure at a minimum must be subject to three pairs of orthogonal ground motion data. In this project, due to the computational effort required for each time-history analysis, the structure was subjected to the single July 5th, 2019 Ridgecrest earthquake in each orthogonal direction. The goal of the project was still completed because an understanding of the linear time-history analysis method was gained. The preceding reasons ultimately led to modal response spectrum analysis being chosen as the final design method. Figure 5.10 shows the iterative process that was followed.

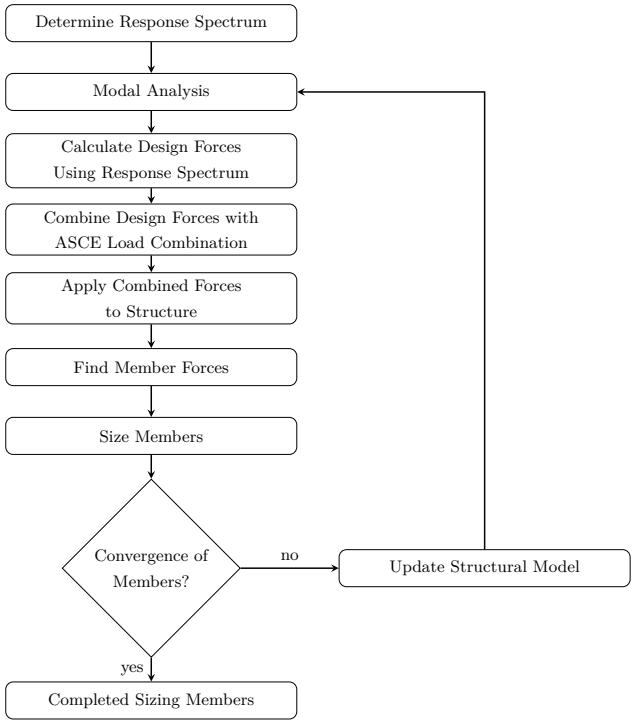

Figure 5.10: Response Spectrum Procedure

The analysis included the calculation of six modes, ultimately yielding 90.35% modal mass participation in the x-direction and 91.23% modal mass participation in the orthogonal y-direction, both above the ASCE 7-16 required combined modal mass participation of 90% in each orthogonal

Table 5.4: Final Modal Properties

Mode	**Period(sec)**	**X Mass**	**Y Mass**	**Z Rot. Mass**
1	0.749	0.1865	0.1362	0.5864
2	0.741	0.3277	0.5464	0.002
3	0.715	0.3549	0.2089	0.2851
4	0.663	0.0176	0.0134	0.0048
5	0.607	0.0147	0.006	0.022
6	0.536	0.0021	0.0014	0.0006

direction. Table 5.4 shows the final modal properties of the structure. While the structure is large, the exceptionally high stiffnesses of the final member sizes caused a short fundamental period of .749 sec. The final member sizes were calculated based on the modal properties in Table 5.4. The final member sizes calculated for the structure are shown in Table 5.5. The members were sized based on the considerations noted in Section 5.3.

Table 5.5: Final Member Sizes

Story	**Joists**	**Girders**	**Columns**
1	W14x22	W18x50	W14x311
2	W14x22	W18x50	W14x283
3	W14x22	W18x50	W14x257
4	W14x22	W18x50	W14x233
Roof	W14x22	W18x50	W14x211

6. Cost Estimate

This chapter provides a total cost estimate of the building, combined with a cost breakdown. The total building cost includes both the cost of materials and the cost of installations. RSMeans Square Foot Costs 2015 was chosen for the estimation of these costs.

To begin, a comparison must first be made between the building designed for gravity loads versus the building designed for combined gravity and seismic loads. The total amount of steel for the building designed for just gravity loads is approximately 427 tons. The total amount of steel for the building designed for combined gravity and seismic loads is approximately 759.5 tons. This difference highlights the significantly more building materials needed for a building designed for combined gravity and seismic loads versus a building designed for solely gravity loads.

The final building cost was scaled by a time-value index to ensure the cost reflected 2019 prices. The time value index was calculated as: $1.0275^{(present\ year-2015)}$. The material and installation cost estimate was calculated using the cost per length of columns and cost per floor area for joists and girders. The material and installation cost of the superstructure is approximately $4.2 million and 24.15% of the total building cost, excluding additional contractor and architect fees. Table 6.1 shows the cost estimate breakdown of the structure in terms of cost per square foot and includes the total cost estimate of each category [15]. Table 6.2 shows the additional fees estimate, including the scaled 2019 cost using the time-value index scaling factor [15]. Appendix E contains the cost estimate data used.

Table 6.1: Material and Installation Cost Estimate Breakdown

Category	**Estimate ($)**	**Percent of Total (%)**	**PSF ($)**
Substructure	$1,906,000	10.88%	$19.06
Superstructure	$4,232,000	24.15%	$42.32
Exterior Enclosure	$3,443,000	19.65%	$34.43
Roofing	$133,200	0.76%	$1.33
Interior	$3,180,400	18.15%	$31.80
Services	$4,629,400	26.42%	$46.29
Total Estimate	**$17,524,400**		

Table 6.2: Anticipated Additional Fees Cost Estimate Breakdown

Additional Fees	**Amount**
Total Est. Incl. Loc. Factor (1.25)	$22,280,000
Total Est. Incl. Time-Value	$24,169,111
Contractor Fees (25% of Subtotal)	$6,042,277
Architect Fees (8% of Subtotal)	$1,933,529
Total Building Cost Estimate	**$32,144,917**
Total Cost Per Sq. Ft.	**321** ($\$/ft^2$)

7. Conclusions and Recommendations

This chapter serves to provide conclusions and recommendations on the completed project. A summary of the conclusions for the significant areas of work is provided first, followed by recommendations for the significant areas of work. The significant areas of work correspond to the three completed objectives of the project: (i) develop finite element programs, (ii) perform structural analysis and design, and (iii) prepare cost estimate. The completion of each objective resulted in a significant completed area of work.

7.1 Conclusions

Objective 1 resulted in the completed finite element code for performing structural analysis of beam and shell elements for static and dynamic loading conditions. The code included also demonstrated post-processing capabilities. An area of the significance of using this code for structural analysis is that there are no restrictions. Student and demo versions of software often include model size restrictions, limiting the size of a structure that can be analyzed. Using self-written or open-source finite element code allows for any size model to be analyzed. The second area of significance includes the knowledge gained when writing finite element code for the application to commercial structural analysis software. The knowledge gained while writing finite element code will yield fewer errors and fewer erroneous results when using commercial structural analysis software. Lastly, the code developed for this project can serve as an aid to students who wish to write structural analysis programs based on the finite element method. While there are no other MQPs to benchmark this work, Chapter 4 provides test cases for verification of the code.

Objective 2 resulted in the completed structural design of a 3D building frame capable of withstanding large seismic loads. The dynamic analysis methods is an area of importance and significance for the structural analysis and design work. A review of MQP reports over the past six years indicates that the dynamic analysis methods included in Objective 2, the linear time-history analysis method and the modal response spectrum analysis method, have not been explored. This report can provide entry-level knowledge for other students who wish to explore these dynamic analysis methods. Additionally, compliance with the ASCE 7-16 dynamic analysis methods requires adhering to detailed code provisions in ASCE 7-16. Following engineering code provisions is an essential component of the Capstone Design Statement.

Objective 3 produced a cost estimate breakdown of the structure. Due to time constraints, the cost estimate breakdown was limited.

7.2 Recommendations

The completed finite element code could be expanded upon in many ways. The first area to build upon could be the examination of beam and shell connections for dynamic analysis. Determining a method for modeling combined beam and shell elements would provide a better representation of the structure. A second area that would prove useful is a graphical user interface or a simple geometry editor for placing beams, drawing two-dimensional elements, etc. Ultimately, for this project, only simple finite element code is needed for the examination of framed structures. For someone wanting to do a similar project, it would be beneficial to have an understanding of the finite element method and the ability to write finite element code before the start of the project.

During the time-history analysis, only a single time-history response was examined. As mentioned in Chapter 5, to comply with ASCE 7-16, the structure needs to be subjected to multiple time-history responses. A project could examine only the linear time-history analysis method by subjecting the structure to a multitude of responses to comply with ASCE 7-16. Similar to the development of the finite element code, someone wanting to do a project on dynamic analysis methods would benefit significantly by having prior knowledge of structural dynamics.

Lastly, the cost estimate could be broken down further. Additionally, further examination of the time-value index may provide more accurate results. While the time-value index considered the long-term trend, building materials are not ordered years in advance. In 2018, various producer price indices for steel products spiked at the start of the China-United States trade war. The effect of this on building costs could be examined.

Bibliography

[1] S. Adhikari and S. A. Phani. "Rayleigh's Classical Damping Revisited". In: *International Conference on Civil Engineering in the New Millennium: Opportunities and Challenges*. Jan. 2007.

[2] Sondipon Adhikari. "Damping Models for Structural Vibration". Doctoral dissertation. University of Cambridge, Sept. 2000.

[3] National Aeronautics and Space Administration. *NASA STRuctrual ANalysis (NASTRAN)*. URL: `https://software.nasa.gov/software/LAR-16804-GS`.

[4] National Aeronautics and Space Administration. *NASTRAN User's Manual*. NASA Headquarters. United States, 1986.

[5] National Aeronautics and Space Administration. *The NASTRAN Programmer's Manual*. NASA Headquarters. United States, 1972.

[6] National Aeronautics and Space Administration. *The NASTRAN Theoretical Manual*. NASA Headquarters. United States, 1981.

[7] Folake Akinpelu. "The Response of Viscously Damped Euler-Bernoulli Beam to Uniform Partially Distributed Moving Loads". In: *Applied Mathematics* 3.3 (Mar. 2012), pp. 199–204.

[8] Leonard D. Albano. "An Axiomatic Approach To Performance-Based Design". Doctoral thesis. Massachusetts Institute of Technology, Feb. 1992.

[9] Klaus-Jurgen Bathe. "What can go wrong in FEA?" In: *Mechanical Engineering* 120 (5 May 1998), pp. 63–65.

[10] Walter S. Brainerd. *Guide to Fortran 2008 Programming*. Springer, 2009.

[11] American Society of Civil Engineers. *Minimum Design Loads and Associated Criteria for Buildings and Other Structures*. American Society of Civil Engineers, 2017.

[12] Open Channel Foundation. *NTTC, OSC Celebrate Space Agency's Birthday, Launch "NASA CLASSICS" Software Apps to Commercial Markets*. URL: `https://web.archive.org/web/20160516224353/http:/www.openchannelsoftware.com/NASA_1.html`.

[13] Jerry H. Ginsberg. *Mechanical and Structural Vibrations: Theory and Applications*. John Wiley & Sons, 2001.

[14] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. 4th ed. The John Hopkins University Press, 2013.

[15] Khant Win Htet. *Wind Load Design of a Triangular Shaped Building Using Finite Element Analysis*. Major Qualifying Project. Worcester Polytechnic Institute, Mar. 2019.

[16] American Concrete Institute. *Manual of Concrete Practice 2017*. American Concrete Institute, 2017.

[17] H. Jeffreys and B.S. Jeffreys. *Methods of Mathematical Physics*. 3rd ed. Cambridge University Press, 1956.

[18] Ramaseshan Kannan. "Numerical Linear Algebra Problems in Structural Analysis". Doctoral dissertation. University of Manchester, 2014.

[19] Ki-ook Kim. "A review of mass matrices for eigenproblems". In: *Computers & Structures* 46 (6 Mar. 1993), pp. 1041–1048.

[20] Ioannis Koutromanos. *Fundamentals of Finite Element Analysis: Linear Finite Element Analysis*. John Wiley & Sons, 2018.

[21] Huei-Huang Lee. *Finite Element Simulations with ANSYS Workbench 16*. SDC Publications, 2015.

[22] Brian W. Mar and Richard N. Palmer. "Does Civil Engineering Need System Engineering?" In: *Journal of Professional Issues in Engineering* 115 (1 Jan. 1989), pp. 45–52.

[23] Richard Morante et al. "Evaluation of Modal Combination Methods for Seismic Response Spectrum Analysis". In: *Transactions of the 15th International Conference on Structural Mechanics in Reactor Technology*. Aug. 1999.

[24] Izuru Okawa, Yuji Ishiyama, and Makoto Watabe. *Fundamentals of Structural Dynamics*. Lecture Note. International Institute of Seismology and Earthquake Engineering (IISEE), 2007.

[25] Deborah F. Pilkey. "Computation of a Damping Matrix for Finite Element Model Updating". Doctoral dissertation. Virginia Polytechnic Institute and State University, Apr. 1998.

[26] Ivatury S. Raju, Jr. Norman F. Knight, and Kunigal N. Shivakumar. "Some Observations on the Current Status of Performing Finite Element Analyses". In: *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. Jan. 2015.

[27] I.M. Smith, D.V. Griffiths, and L. Margetts. *Programming the Finite Element Method*. 5th ed. John Wiley & Sons, 2014.

[28] American Institute of Steel Construction. *Steel Construction Manual*. 15th ed. American Institute of Steel Construction, 2017.

[29] Haluk Sucuoglu and Sinan Akkar. *Basic Earthquake Engineering: From Seismology to Analysis and Design*. Springer, 2014.

[30] Joseph W. Tedesco, William G. McDougal, and C. Allen Ross. *Structural Dynamics: Theory and Applications*. Addison Wesley Longman, 1999.

[31] S. Timoshenko and S. Woinowsky-Krieger. *Theory of Plates and Shells*. 2nd ed. McGraw-Hill, 1959.

[32] S. Timoshenko and D. H. Young. *Advanced Dynamics*. 1st ed. McGraw-Hill, 1948.

[33] P. Tong, T.H.H. Pian, and L.L. Bucciarblli. "Mode shapes and frequencies by finite element method using consistent and lumped masses". In: *Computers & Structures* 1 (4 Dec. 1971), pp. 623–638.

[34] Robert E. White. *Computational Mathematics: Models, Methods, and Analysis with MATLAB and MPI*. 2nd ed. CRC Press, 2016.

[35] Walter Wunderlich and Walter D. Pilkey. *Mechanics of Structures: Variational and Computational Methods*. 2nd ed. CRC Press, 2003.

A. MATLAB Code

Coding for Figures

Figure 2.4: Calculated Response Spectrum for Various Damping Ratios

```
clc;
clear;
variablewidths=[10,10,10,10,10,10,10,10];
opts=fixedWidthImportOptions('NumVariables',8,'VariableWidths',variablewidths);
accel=readmatrix('chinalakeacceleration.txt',opts);
accel=str2double(accel);
accel=transpose(reshape(accel,[],1));
accel=rmmissing(accel);
dperiod=.005;                                                    % natural period step
periodint=3;                                  % total natural period time interval to examine
periodstep=periodint/dperiod;                                    % number of natural period steps
gam=.5;                                                  % newmark constant gamma from Table 2.1
beta=.25;                                                % newmark constant beta from Table 2.1
dtim=.02;                                                        % time step from Table 2.1
tint=301;                                                % total time interval from Figure 2.1
nstep=tint/dtim;                                                 % number of time steps
K=1;                                                  % constant stiffness for each SDOF system
n=1;                                                             % system number of free DOF
zeta=.05;                                                                % damping ratio
spectraldisp=zeros(n,periodstep);          % initialize spectral displacement data for each period
spectralvelo=zeros(n,periodstep);             % initialize spectral velocity data for each period
spectralacce=zeros(n,periodstep);          % initialize spectral acceleration data for each period
for i=1:periodstep
    Tn=i*dperiod;                                                % natural period of SDOF system
    M=((Tn^2)*K)/(4*pi^2);                                   % varying mass for each SDOF system
    C=2*zeta*sqrt(K*M);                                  % varying damping for each SDOF system
    F=-M*accel;                            % varying time-dependent loading for each SDOF system
    disp=zeros(n,nstep);                       % initialize displacement data for each time step
    velo=zeros(n,nstep);                          % initialize velocity data for each time step
    acce=M\(F(:,1)-C*velo(:,1)-K*disp(:,1));       % calculate initial acceleration from Table 2.1
    a0=1/(beta*(dtim^2));                                        % constant a0 from Table 2.1
    a1=(gam/beta)*(1/dtim);                                      % constant a1 from Table 2.1
    a2=(1/gam)*a1;                                               % constant a2 from Table 2.1
    a3=(.5-beta)/beta;                                           % constant a3 from Table 2.1
    a4=(gam/beta)-1;                                             % constant a4 from Table 2.1
    a5=(.5-beta)*((gam*dtim)/beta)+(gam-1)*dtim;                 % constant a5 from Table 2.1
    a6=dtim*(1-gam);                                             % constant a6 from Table 2.1
    a7=gam*dtim;                                                 % constant a7 from Table 2.1
```

```
39      Keff=K+a0*M+a1*C;                                   % initial effective stiffness matrix from Table 2.1
40      for j=1:nstep-1                                     % Newmark method from Table 2.1
41          Feff=M*(a0*disp(:,j)+a2*velo(:,j)+a3*acce(:,j))...
42                  +C*(a1*disp(:,j)+a4*velo(:,j)...
43                  +a5*acce(:,j))+F(:,j+1);
44          disp(:,j+1)=Keff\Feff;
45          acce(:,j+1)=a0*(disp(:,j+1)-disp(:,j))-a2*velo(:,j)-a3*acce(:,j);
46          velo(:,j+1)=velo(:,j)+a6*acce(:,j)+a7*acce(:,j+1);
47      end
48      absaccel=acce+accel;                                % absolute acceleration of the DOF
49      spectraldisp(1,i)=max(abs(disp));                   % spectral displacement
50      spectralacce(1,i)=0.00102*max(abs(absaccel));       % spectral acceleration (convert to g)
51      spectralvelo(1,i)=max(abs(velo));                   % spectral velocity
52  end
53  y1=spectralacce(1,:);
54  x1=linspace(dperiod,periodint,periodstep);
55  figure(1)
56  plot(x1,y1,'-')                                         % plotting spectral acceleration shown in Figure 2.4
```

Figure 2.6: Primary Mode Shapes of Cantilever Beam

```
1   clc;
2   clear;
3   L=1/6;                                                  % length of each element
4   EI=0.08333;                                             % modulus of elasticity*moment of inertia
5   rhoA=1.0;                                               % density*area
6   nel=6;                                                  % number of elements
7   eldof=4;                                                % number of DOF per element
8   n=12;                                                   % system number of free DOF
9   nf=[0 0;1 2;3 4;5 6;7 8;9 10;11 12];                    % nodal DOF
10  connec=[1 2;2 3;3 4;4 5;5 6;6 7];                       % element connectivity
11  x=sym('x');                                             % symbolic variable for shape functions
12  coord=[0 0;1/6 0;2/6 0;3/6 0;4/6 0;5/6 0;1 0];          % node coordinates
13  N1=1-((3*x^2)/(L^2))+((2*x^3)/(L^3));                   % Hermitian shape function 1
14  N2=x-((2*x^2)/(L))+((x^3)/(L^2));                       % Hermitian shape function 2
15  N3=((3*x^2)/(L^2))-((2*x^3)/(L^3));                     % Hermitian shape function 3
16  N4=(-(x^2)/(L))+((x^3)/(L^2));                          % Hermitian shape function 4
17  N=[N1 N2 N3 N4];                                        % shape function matrix
18  keg(1,1)=6;                                             % stiffness matrix
19  keg(1,2)=3*L;
20  keg(1,3)=-6;
21  keg(2,2)=2*(L^2);
22  keg(2,3)=-3*L;
23  keg(2,4)=L^2;
24  keg(1,4)=keg(1,2);
25  keg(2,1)=keg(1,2);
26  keg(3,1)=keg(1,3);
27  keg(3,2)=keg(2,3);
28  keg(3,3)=keg(1,1);
29  keg(3,4)=keg(2,3);
30  keg(4,1)=keg(1,2);
31  keg(4,2)=keg(2,4);
32  keg(4,3)=keg(2,3);
33  keg(4,4)=keg(2,2);
34  keg=((2*EI)/(L^3))*keg;
35  meg(1,1)=156;                                           % mass matrix
```

```
meg(1,2)=22*L;
meg(1,3)=54;
meg(1,4)=-13*L;
meg(2,2)=4*(L^2);
meg(2,3)=13*L;
meg(3,4)=-22*L;
meg(2,4)=-3*(L^2);
meg(2,1)=meg(1,2);
meg(3,1)=meg(1,3);
meg(3,2)=meg(2,3);
meg(3,3)=meg(1,1);
meg(4,1)=meg(1,4);
meg(4,2)=meg(2,4);
meg(4,3)=meg(3,4);
meg(4,4)=meg(2,2);
meg=((rhoA*L)/420)*meg;
K=zeros(n);                                          % initialize system stiffness matrix
M=zeros(n);                                          % initialize system mass matrix
for i=1:nel                                          % form system matrices
    node1=connec(i,1);                               % node 1 of element i
    node2=connec(i,2);                               % node 2 of element i
    g=[nf(node1,1);                                  % steering vector of element i
       nf(node1,2);
       nf(node2,1);
       nf(node2,2)];
    for iel=1:eldof
        if g(iel)~=0
            for jel=1:eldof
                if g(jel)~=0
                    K(g(iel),g(jel))=K(g(iel),g(jel))+keg(iel,jel);
                    M(g(iel),g(jel))=M(g(iel),g(jel))+meg(iel,jel);
                end
            end
        end
    end
end
[eigvec,eigval]=eig(K,M);                            % solve eigenvalue problem from Eq. (2.9)
omega=sort(sqrt(diag(eigval)));                      % natural frequencies
phi=zeros(n);                                        % initialize mode shape matrix
for i=1:n                                            % point normalization of mode shapes
    phi(:,i)=eigvec(:,i)/eigvec(11,i);
end
for i=1:4                                            % plotting mode shapes shown in Figure 2.6
    u(1,1)=0;
    u(1,2)=0;
    u(2,1)=phi(1,i);
    u(2,2)=phi(2,i);
    u(3,1)=phi(3,i);
    u(3,2)=phi(4,i);
    u(4,1)=phi(5,i);
    u(4,2)=phi(6,i);
    u(5,1)=phi(7,i);
    u(5,2)=phi(8,i);
    u(6,1)=phi(9,i);
    u(6,2)=phi(10,i);
    u(7,1)=phi(11,i);
    u(7,2)=phi(12,i);
    for iel=1:nel
```

```
94          node1=connec(iel,1);
95          node2=connec(iel,2);
96          x1=coord(node1,1);
97          x2=coord(node2,1);
98          uel(1,1)=u(node1,1);                    % node 1 vertical displacement of element i
99          uel(2,1)=u(node1,2);                    % node 1 rotation of element i
100         uel(3,1)=u(node2,1);                    % node 2 vertical displacement of element i
101         uel(4,1)=u(node2,2);                    % node 2 rotation of element i
102         uy=N*uel;                               % modal vertical displacement field of element i
103         x=linspace(0,1/6,20);
104         uy=subs(uy,x);
105         xnew=linspace(x1,x2,20);
106         C={[0 0.4470 0.7410],[0.8500 0.3250 0.0980]...      % color of mode shape plots
107            ,[0.9290 0.6940 0.1250],[0.4940 0.1840 0.5560]};
108         plot(xnew,uy,'color',C{i},'Linewidth',3)
109         hold on
110     end
111 end
112 ax=gca;
113 ax.XAxisLocation='origin';
114 ax.YAxisLocation='origin';
```

Black-Box Functions

ebbeammass3D.m

```
1  function[meg]=ebbeammass3D(i,elconnec,elprop,ncoord)
2
3  n1=elconnec(i,1);                                   % node 1 of element i
4  n2=elconnec(i,2);                                   % node 2 of element i
5  x1=ncoord(n1,1);                                    % x-coord of node 1
6  y1=ncoord(n1,2);                                    % y-coord of node 1
7  z1=ncoord(n1,3);                                    % z-coord of node 1
8  x2=ncoord(n2,1);                                    % x-coord of node 2
9  y2=ncoord(n2,2);                                    % y-coord of node 2
10 z2=ncoord(n2,3);                                    % z-coord of node 2
11 L=sqrt(((x2-x1)^2)+((y2-y1)^2)+((z2-z1)^2));        % length of element i
12 A=elprop(i,2);                                      % area of element i
13 Iz=elprop(i,5);                                     % moment of inertia about the z-axis of element i
14 Iy=elprop(i,6);                                     % moment of inertia about the y-axis of element i
15 gamma=elprop(i,7);                                  % transformation angle (in degrees) of element i
16 gamma=gamma*pi/180;                                 % convert to radians
17 eltype=elprop(i,8);                                 % element type of element i
18 rho=elprop(i,9);                                    % density of element i
19 if eltype==1                                        % element type 1 is rigid-rigid
20     s1=1/3;                                         % constants for mass matrix
21     s2=1/6;
22     s3=13/35;
23     s4=(11*L)/(210);
24     s5=(9/70);
25     s6=(13*L)/(420);
26     s7=(Iy+Iz)/(3*A);
27     s8=(Iy+Iz)/(6*A);
28     s9=(L^2)/(105);
29     s10=(L^2)/(140);
```

```
30      s11=-s4;
31      s12=-s6;
32      s13=-s10;
33      mel=rho*A*L*[s1 0   0   0   0   0   s2 0   0   0   0   0;          % mass matrix in local coord
34                   0  s3  0   0   0   s4  0  s5  0   0   0 s12;
35                   0  0   s3  0  s11 0   0  0   s5  0  s6  0;
36                   0  0   0   s7 0   0   0  0   0   s8 0   0;
37                   0  0   s11 0  s9  0   0  0   s12 0  s13 0;
38                   0  s4  0   0   0   s9  0  s6  0   0   0 s13;
39                   s2 0   0   0   0   0   s1 0   0   0   0   0;
40                   0  s5  0   0   0   s6  0  s3  0   0   0 s11;
41                   0  0   s5  0  s12 0   0  0   s3  0  s4  0;
42                   0  0   0   s8 0   0   0  0   0   s7 0   0;
43                   0  0   s6  0  s13 0   0  0   s4  0  s9  0;
44                   0  s12 0   0   0   s13 0  s11 0   0   0 s9];
45  elseif eltype==2                              % element type 2 is xyz rotation released pinned-rigid
46      s1=1/3;                                                          % constants for mass matrix
47      s2=1/6;
48      s3=1/12;
49      s4=-1/24;
50      s5=13/48;
51      s6=L/120;
52      s7=-L/120;
53      s8=(7*L)/240;
54      s9=(-7*L)/240;
55      s10=(Iy+Iz)/(4*A);
56      s11=(L^2)/240;
57      mel=rho*A*L*[s1 0  0  0 0 0 s2 0  0  0   0    0;
58                   0  s3 0  0 0 0 0  s4 0  0   0   s6;
59                   0  0  s3 0 0 0 0  0  s4 0   s7   0;
60                   0  0  0  0 0 0 0  0  0  0   0    0;
61                   0  0  0  0 0 0 0  0  0  0   0    0;
62                   0  0  0  0 0 0 0  0  0  0   0    0;
63                   s2 0  0  0 0 0 s1 0  0  0   0    0;
64                   0  s4 0  0 0 0 0  s5 0  0   0   s9;
65                   0  0  s4 0 0 0 0  0  s5 0   s8   0;
66                   0  0  0  0 0 0 0  0  0  s10 0    0;
67                   0  0  s7 0 0 0 0  0  s8 0   s11  0;
68                   0  s6 0  0 0 0 0  s9 0  0   0   s11];
69  elseif eltype==3                          % element type 3 is x,y,z rotation released rigid-pinned
70      s1=1/3;                                                          % constants for mass matrix
71      s2=1/6;
72      s3=1/12;
73      s4=-1/24;
74      s5=13/48;
75      s6=L/120;
76      s7=-L/120;
77      s8=(7*L)/240;
78      s9=(-7*L)/240;
79      s10=(Iy+Iz)/(4*A);
80      s11=(L^2)/240;
81      mel=rho*A*L*[s1 0  0  0   0   0   s2 0  0  0 0  0;
82                   0  s5 0  0   0   s8  0  s4 0  0 0  0;
83                   0  0  s5 0   s9  0   0  0  s4 0 0  0;
84                   0  0  0  s10 0   0   0  0  0  0 0  0;
85                   0  0  s9 0   s11 0   0  0  s6 0 0  0;
86                   0  s8 0  0   0   s11 0  s7 0  0 0  0;
87                   s2 0  0  0   0   0   s1 0  0  0 0  0;
```

```
88                      0   s4  0   0     0    s7   0   s3  0   0  0   0;
89                      0   0   s4  0     s6   0    0   0   s3  0  0   0;
90                      0   0   0   0     0    0    0   0   0   0  0   0;
91                      0   0   0   0     0    0    0   0   0   0  0   0;
92                      0   0   0   0     0    0    0   0   0   0  0  0];
93  elseif eltype==4                                % element type 4 is x,y,z rotation released pinned-pinned
94      s1=1/3;                                                          % constants for mass matrix
95      s2=1/6;
96      s3=1/15;
97      s4=1/60;
98      mel=rho*A*L*[s1  0   0   0  0  0  s2  0   0   0  0   0;
99                   0   s3  0   0  0  0  0   s4  0   0  0   0;
100                  0   0   s3  0  0  0  0   0   s4  0  0   0;
101                  0   0   0   0  0  0  0   0   0   0  0   0;
102                  0   0   0   0  0  0  0   0   0   0  0   0;
103                  0   0   0   0  0  0  0   0   0   0  0   0;
104                  s2  0   0   0  0  0  s1  0   0   0  0   0;
105                  0   s4  0   0  0  0  0   s3  0   0  0   0;
106                  0   0   s4  0  0  0  0   0   s3  0  0   0;
107                  0   0   0   0  0  0  0   0   0   0  0   0;
108                  0   0   0   0  0  0  0   0   0   0  0   0;
109                  0   0   0   0  0  0  0   0   0   0  0  0];
110
111 end
112 sg=sin(gamma);                                          % constants for member rotation matrix
113 cg=cos(gamma);
114 r=zeros(3);                                             % initialize member rotation matrix
115 if x1==x2 && z1==z2                                     % if element is parallel to global y-axis
116     r(1,2)=1;
117     r(2,1)=-cg;
118     r(2,3)=sg;
119     r(3,1)=sg;
120     r(3,3)=cg;
121 else                                                    % if element is not parallel to global y-axis
122     Cx=(x2-x1)/L;                                                    % direction cosines
123     Cy=(y2-y1)/L;
124     Cz=(z2-z1)/L;
125     den=sqrt((Cx^2)+(Cz^2));
126     r(1,1)=Cx;
127     r(1,2)=Cy;
128     r(1,3)=Cz;
129     r(2,1)=(-Cx*Cy*cg-Cz*sg)/den;
130     r(2,2)=den*cg;
131     r(2,3)=(-Cy*Cz*cg+Cx*sg)/den;
132     r(3,1)=(Cx*Cy*sg-Cz*cg)/den;
133     r(3,2)=-den*sg;
134     r(3,3)=(Cy*Cz*sg+Cx*cg)/den;
135 end
136 T=zeros(size(mel));                                     % initialize transformation matrix
137 T(1:3,1:3)=r;
138 T(4:6,4:6)=r;
139 T(7:9,7:9)=r;
140 T(10:12,10:12)=r;
141 meg=T'*mel*T;                                           % transform mass matrix to global coordinates
142
143 end                                                                  % end function
```

ebbeamstiff3D.m

```
function [keg]=ebbeamstiff3D(i,elconnec,elprop,ncoord)

n1=elconnec(i,1);                                                    % node 1 of element i
n2=elconnec(i,2);                                                    % node 2 of element i
x1=ncoord(n1,1);                                                     % x-coord of node 1
y1=ncoord(n1,2);                                                     % y-coord of node 1
z1=ncoord(n1,3);                                                     % z-coord of node 1
x2=ncoord(n2,1);                                                     % x-coord of node 2
y2=ncoord(n2,2);                                                     % y-coord of node 2
z2=ncoord(n2,3);                                                     % z-coord of node 2
L=sqrt(((x2-x1)^2)+((y2-y1)^2)+((z2-z1)^2));                         % length of element i
E=elprop(i,1);                                          % modulus of elasticity of element i
A=elprop(i,2);                                                       % area of element i
G=elprop(i,3);                                            % modulus of rigidity of element i
J=elprop(i,4);                                             % torsional constant of element i
Iz=elprop(i,5);                                % moment of inertia about the z-axis of element i
Iy=elprop(i,6);                                % moment of inertia about the y-axis of element i
gamma=elprop(i,7);                              % transformation angle (in degrees) of element i
gamma=gamma*pi/180;                                                  % convert to radians
eltype=elprop(i,8);                                                  % element type of element i
if eltype==1                                                % element type 1 is rigid-rigid
    s1=(E*A)/(L);                                           % constants for stiffness matrix
    s2=(12*E*Iz)/(L^3);
    s3=(6*E*Iz)/(L^2);
    s4=(12*E*Iy)/(L^3);
    s5=(6*E*Iy)/(L^2);
    s6=(G*J)/(L);
    s7=(4*E*Iy)/(L);
    s8=(2*E*Iy)/(L);
    s9=(4*E*Iz)/(L);
    s10=(2*E*Iz)/(L);
    s11=-s1;
    s12=-s2;
    s13=-s4;
    s14=-s5;
    s15=-s6;
    s16=-s3;
    kel=[s1  0   0   0   0   0   s11 0   0   0   0   0;          % stiffness matrix in local coord
         0   s2  0   0   0   s3  0   s12 0   0   0   s3;
         0   0   s4  0   s14 0   0   0   s13 0   s14 0;
         0   0   0   s6  0   0   0   0   0   s15 0   0;
         0   0   s14 0   s7  0   0   0   s5  0   s8  0;
         0   s3  0   0   0   s9  0   s16 0   0   0 s10;
         s11 0   0   0   0   0   s1  0   0   0   0   0;
         0   s12 0   0   0 -s3  0   s2  0   0   0 s16;
         0   0   s13 0   s5  0   0   0   s4  0   s5  0;
         0   0   0   s15 0   0   0   0   0   s6  0   0;
         0   0   s14 0   s8  0   0   0   s5  0   s7  0;
         0   s3  0   0   0   s10 0   s16  0   0   0 s9];
elseif eltype==2                          % element type 2 is x,y,z rotation released pinned-rigid
    s1=(E*A)/(L);                                           % constants for stiffness matrix
    s2=(3*E*Iz)/(L^3);
    s3=(3*E*Iz)/(L^2);
    s4=(3*E*Iz)/(L);
    s5=(3*E*Iy)/(L^3);
    s6=(3*E*Iy)/(L^2);
```

```
57      s7=(3*E*Iy)/(L);
58      s8=-s1;
59      s9=-s2;
60      s10=-s3;
61      s11=-s5;
62      s12=-s6;
63      kel=[s1 0  0   0 0 0 s8 0   0   0 0   0;                    % stiffness matrix in local coord
64           0  s2 0   0 0 0 0  s9  0   0 0  s3;
65           0  0  s5  0 0 0 0  0   s11 0 s12 0;
66           0  0  0   0 0 0 0  0   0   0 0   0;
67           0  0  0   0 0 0 0  0   0   0 0   0;
68           0  0  0   0 0 0 0  0   0   0 0   0;
69           s8 0  0   0 0 0 s1 0   0   0 0   0;
70           0  s9 0   0 0 0 0  s2  0   0 0 s10;
71           0  0  s11 0 0 0 0  0   s5  0 s6  0;
72           0  0  0   0 0 0 0  0   0   0 0   0;
73           0  0  s12 0 0 0 0  0   s6  0 s7  0;
74           0  s3 0   0 0 0 0  s10 0   0 0 s4];
75  elseif eltype==3                          % element type 3 is x,y,z rotation released rigid-pinned
76      s1=(E*A)/(L);                                                % constants for stiffness matrix
77      s2=(3*E*Iz)/(L^3);
78      s3=(3*E*Iz)/(L^2);
79      s4=(3*E*Iz)/(L);
80      s5=(3*E*Iy)/(L^3);
81      s6=(3*E*Iy)/(L^2);
82      s7=(3*E*Iy)/(L);
83      s8=-s1;
84      s9=-s2;
85      s10=-s3;
86      s11=-s5;
87      s12=-s6;
88      kel=[s1 0  0   0 0   0   s8 0   0   0 0  0;                  % stiffness matrix in local coord
89           0  s2 0   0 0   s3  0  s9  0   0 0  0;
90           0  0  s5  0 s12 0   0  0   s11 0 0  0;
91           0  0  0   0 0   0   0  0   0   0 0  0;
92           0  0  s12 0 s7  0   0  0   s6  0 0  0;
93           0  s3 0   0 0   s4  0  s10 0   0 0  0;
94           s8 0  0   0 0   0   s1 0   0   0 0  0;
95           0  s9 0   0 0   s10 0  s2  0   0 0  0;
96           0  0  s11 0 s6  0   0  0   s5  0 0  0;
97           0  0  0   0 0   0   0  0   0   0 0  0;
98           0  0  0   0 0   0   0  0   0   0 0  0;
99           0  0  0   0 0   0   0  0   0   0 0 0];
100 elseif eltype==4                             % element type 4 is x,y,z released pinned-pinned
101     s1=(E*A)/(L);                                            % constants for stiffness matrix
102     s2=-s1;
103     kel=[s1 0 0 0 0 0 s2 0 0 0 0  0;                         % stiffness matrix in local coord
104          0  0 0 0 0 0 0  0 0 0 0  0;
105          0  0 0 0 0 0 0  0 0 0 0  0;
106          0  0 0 0 0 0 0  0 0 0 0  0;
107          0  0 0 0 0 0 0  0 0 0 0  0;
108          0  0 0 0 0 0 0  0 0 0 0  0;
109          s2 0 0 0 0 0 s1 0 0 0 0  0;
110          0  0 0 0 0 0 0  0 0 0 0  0;
111          0  0 0 0 0 0 0  0 0 0 0  0;
112          0  0 0 0 0 0 0  0 0 0 0  0;
113          0  0 0 0 0 0 0  0 0 0 0  0;
114          0  0 0 0 0 0 0  0 0 0 0 0];
```

```
115 end
116 Cx=(x2-x1)/L;                                                    % direction cosines
117 Cy=(y2-y1)/L;
118 Cz=(z2-z1)/L;
119 sg=sin(gamma);                                                  % constants for member rotation matrix
120 cg=cos(gamma);
121 den=sqrt((Cx^2)+(Cz^2));
122 r=zeros(3);                                                     % initialize member rotation matrix
123 if x1==x2 && z1==z2                                            % if element is parallel to global y-axis
124     r(1,2)=1;
125     r(2,1)=-Cy*cg;
126     r(2,3)=sg;
127     r(3,1)=Cy*sg;
128     r(3,3)=cg;
129 else                                                           % if element is not parallel to global y-axis
130     r(1,1)=Cx;
131     r(1,2)=Cy;
132     r(1,3)=Cz;
133     r(2,1)=(-Cx*Cy*cg-Cz*sg)/den;
134     r(2,2)=den*cg;
135     r(2,3)=(-Cy*Cz*cg+Cx*sg)/den;
136     r(3,1)=(Cx*Cy*sg-Cz*cg)/den;
137     r(3,2)=-den*sg;
138     r(3,3)=(Cy*Cz*sg+Cx*cg)/den;
139 end
140 T=zeros(size(kel));                                            % initialize transformation matrix
141 T(1:3,1:3)=r;
142 T(4:6,4:6)=r;
143 T(7:9,7:9)=r;
144 T(10:12,10:12)=r;
145 keg=transpose(T)*kel*T;                                        % transform stiffness matrix to global coordinates
146
147 end
```

formK.m

```
1  function [K]=formK(K,keg,g,eldof)
2
3  for i=1:eldof
4      if g(i)~=0                                                 % if DOF is not constrained
5          for j=1:eldof
6              if g(j)~=0                                         % if DOF is not constrained
7                  K(g(i),g(j))=K(g(i),g(j))+keg(i,j);            % add component tied to free DOF
8              end
9          end
10     end
11 end
12
13 end
```

formM.m

```
1  function [M]=formM(M,meg,g,eldof)
2
```

```
for i=1:eldof
    if g(i)~=0                                                  % if DOF is not constrained
        for j=1:eldof
            if g(j)~=0                                          % if DOF is not constrained
                M(g(i),g(j))=M(g(i),g(j))+meg(i,j);             % add component tied to free DOF
            end
        end
    end
end

end
```

Q4membranestiff.m

```
function[keg]=Q4membranestiff(i,coord,prop,eldof)
E=prop(i,1);                                                    % modulus of elasticity of element i
v=prop(i,2);                                                    % poisson ratio of element i
th=prop(i,3);                                                   % thickness of element i
nip=4;                                                          % number of integration points
Dm=(E*th)/(1-v^2);
s1=(1-v)/2;
deem=Dm*[1 v 0;v 1 0;0 0 s1];                                   % membrane stress-strain matrix
gauss=[-1/sqrt(3) -1/sqrt(3) 1;          % coordinates and weights for Gauss-Legendre integration
        1/sqrt(3) -1/sqrt(3) 1;
        1/sqrt(3)  1/sqrt(3) 1;
       -1/sqrt(3) 1/sqrt(3) 1];
keg=zeros(eldof);                                               % initialize stiffness matrix
for j=1:nip                                                % stiffness matrix formulation from Eq. (5.6)
    eta=gauss(j,1);                                             % coordinates for each integration point
    xi=gauss(j,2);
    w=gauss(j,3);                                               % integration point weight
    der=0.25*[-(1-eta) (1-eta) (1+eta) -(1+eta);                % derivatives of shape functions
              -(1-xi) -(1+xi)  (1+xi)  (1-xi)];
    jac=der*coord;                                              % jacobian matrix
    deriv=jac\der;                                              % der in global coordinates
    beem=zeros(3,8);                                % initialize membrane strain-displacement matrix
    beem(1,1)=deriv(1,1);
    beem(1,3)=deriv(1,2);
    beem(1,5)=deriv(1,3);
    beem(1,7)=deriv(1,4);
    beem(2,2)=deriv(2,1);
    beem(2,4)=deriv(2,2);
    beem(2,6)=deriv(2,3);
    beem(2,8)=deriv(2,4);
    beem(3,1)=deriv(2,1);
    beem(3,2)=deriv(1,1);
    beem(3,3)=deriv(2,2);
    beem(3,4)=deriv(1,2);
    beem(3,5)=deriv(2,3);
    beem(3,6)=deriv(1,3);
    beem(3,7)=deriv(2,4);
    beem(3,8)=deriv(1,4);
    keg=keg+det(jac)*th*w*beem'*deem*beem;                      % stiffness matrix
end

end
```

Q4mesh.m

```
function [ncoord,connec,nnd,nel]=Q4mesh(nelx,nely,dhx,dhy,xorigin,yorigin)

nnd=0;
nel=0;
for i = 1:nelx
   for j=1:nely
        nel = nel + 1;
        n1=j+(i-1)*(nely+1);
        ncoord(n1,1)=(i-1)*dhx-xorigin;
        ncoord(n1,2)=(j-1)*dhy-yorigin;
        n2=j+i*(nely+1);
        ncoord(n2,1)=i*dhx-xorigin;
        ncoord(n2,2)=(j-1)*dhy-yorigin;
        n4=n1+1;
        ncoord(n4,1)=(i-1)*dhx-xorigin;
        ncoord(n4,2)=j*dhy-yorigin;
        n3=n2+1;
        ncoord(n3,1)=i*dhx-xorigin;
        ncoord(n3,2)=j*dhy-yorigin;
        connec(nel,1)=n1;
        connec(nel,2)=n2;
        connec(nel,3)=n3;
        connec(nel,4)=n4;
        nnd=n3;
   end
end
for i=1:nnd
    ncoord(i,3)=0;
end

end
```

Q4shellmass.m

```
function [meg]=Q4shellmass(i,coord,prop,eldof)
rho=prop(i,4);
nip=4;                                                  % number of integration points
xcoord=coord(:,1);
ycoord=coord(:,2);
xycoord=[xcoord ycoord];
gauss=[-1/sqrt(3)  -1/sqrt(3) 1;
        1/sqrt(3)  -1/sqrt(3) 1;
        1/sqrt(3)   1/sqrt(3) 1;
       -1/sqrt(3)   1/sqrt(3) 1];
meg=zeros(eldof);
for j=1:nip
    eta=gauss(j,1);                             % coordinates for each integration point
    xi=gauss(j,2);
    w=gauss(j,3);                                                % integration point weight
    fun=0.25*[(1-xi-eta+xi*eta);              % shape functions for each integration point
              (1+xi-eta-xi*eta);
              (1+xi+eta+xi*eta);
              (1-xi+eta-xi*eta)];
    der=0.25*[-(1-eta)  (1-eta) (1+eta) -(1+eta);         % derivatives of shape functions
```

```
21                  -(1-xi)   -(1+xi)   (1+xi)    (1-xi)];
22        jac=der*xycoord;                                                  % jacobian matrix
23        N1=fun(1,1);
24        N2=fun(2,1);
25        N3=fun(3,1);
26        N4=fun(4,1);
27        nee=zeros(5,24);
28        nee(1,1)=N1;
29        nee(1,7)=N2;
30        nee(1,13)=N3;
31        nee(1,19)=N4;
32        nee(2,2)=N1;
33        nee(2,8)=N2;
34        nee(2,14)=N3;
35        nee(2,20)=N4;
36        nee(3,3)=N1;
37        nee(3,9)=N2;
38        nee(3,15)=N3;
39        nee(3,21)=N4;
40        nee(4,4)=N1;
41        nee(4,10)=N2;
42        nee(4,16)=N3;
43        nee(4,22)=N4;
44        nee(5,5)=N1;
45        nee(5,11)=N2;
46        nee(5,17)=N3;
47        nee(5,23)=N4;
48        meg=meg+det(jac)*w*transpose(nee)*rho*nee;
49    end
50
51    end
```

Q4shellstiff.m

```
1   function[keg]=Q4shellstiff(i,coord,prop,eldof)
2   E=prop(i,1);                                          % modulus of elasticity of element i
3   v=prop(i,2);                                          % poisson ratio of element i
4   th=prop(i,3);                                         % thickness of element i
5   x1=coord(1,1);
6   x2=coord(2,1);
7   x3=coord(3,1);
8   x4=coord(4,1);
9   y1=coord(1,2);
10  y2=coord(2,2);
11  y3=coord(3,2);
12  y4=coord(4,2);
13  z1=coord(1,3);
14  z2=coord(2,3);
15  z3=coord(3,3);
16  z4=coord(4,3);
17  L13=sqrt(((x3-x1)^2)+((y3-y1)^2)+((z3-z1)^2));    % begin transformation for global to local coord
18  L24=sqrt(((x4-x2)^2)+((y4-y2)^2)+((z4-z2)^2));
19  v1=(1/L13)*[x3-x1;y3-y1;z3-z1];
20  v2=(1/L24)*[x4-x2;y4-y2;z4-z2];
21  n=(cross(v1,v2))/(norm(cross(v1,v2)));
22  t1=(v1-v2)/(norm(v1-v2));
```

```
23  t2=(v1+v2)/(norm(v1+v2));
24  t(1,1)=t1(1,1);
25  t(1,2)=t1(2,1);
26  t(1,3)=t1(3,1);
27  t(2,1)=t2(1,1);
28  t(2,2)=t2(2,1);
29  t(2,3)=t2(3,1);
30  t(3,1)=n(1,1);
31  t(3,2)=n(2,1);
32  t(3,3)=n(3,1);
33  n1coord=(coord(1,:))';
34  n2coord=(coord(2,:))';
35  n3coord=(coord(3,:))';
36  n4coord=(coord(4,:))';
37  n1local=t*n1coord;
38  n2local=t*n2coord;
39  n3local=t*n3coord;
40  n4local=t*n4coord;
41  lcoord=[n1local n2local n3local n4local];
42  lcoord=lcoord';
43  nip=4;                                                    % number of integration points
44  Dm=(E*th)/(1-v^2);
45  s1=(1-v)/2;
46  deem=Dm*[1 v 0;v 1 0;0 0 s1];                             % membrane stress-strain matrix
47  Db=(E*(th^3))/(12*(1-v^2));
48  deeb=Db*[1 v 0;v 1 0;0 0 s1];                             % bending stress-strain matrix
49  G=E/(2*(1+v));                                                    % shear modulus
50  k=5/6;                                                      % shear correction factor
51  dees=k*G*th*[1 0;0 1];                                    % shear stress-strain matrix
52  gauss=[-1/sqrt(3) -1/sqrt(3) 1;          % coordinates and weights for Gauss-Legendre integration
53          1/sqrt(3) -1/sqrt(3) 1;
54          1/sqrt(3)  1/sqrt(3) 1;
55         -1/sqrt(3) 1/sqrt(3) 1];
56  kel=zeros(eldof);                                         % initialize stiffness matrix
57  kem=zeros(eldof);                                       % initialize membrane stiffness matrix
58  keb=zeros(eldof);                                        % initialize bending stiffness matrix
59  kes=zeros(eldof);                                          % initialize shear stiffness matrix
60  for j=1:nip                                       % stiffness matrix formulation from Eq. (5.6)
61      eta=gauss(j,1);                                    % coordinates for each integration point
62      xi=gauss(j,2);
63      w=gauss(j,3);                                             % integration point weight
64      fun=0.25*[(1-xi-eta+xi*eta);                                      % shape functions
65                (1+xi-eta-xi*eta);
66                (1+xi+eta+xi*eta);
67                (1-xi+eta-xi*eta)];
68      der=0.25*[-(1-eta) (1-eta) (1+eta) -(1+eta);              % derivatives of shape functions
69                -(1-xi) -(1+xi)  (1+xi)   (1-xi)];
70      jac=der*lcoord(:,1:2);                                            % jacobian matrix
71      deriv=jac\der;                                          % der in global coordinates
72      dn1dx=deriv(1,1);
73      dn2dx=deriv(1,2);
74      dn3dx=deriv(1,3);
75      dn4dx=deriv(1,4);
76      dn1dy=deriv(2,1);
77      dn2dy=deriv(2,2);
78      dn3dy=deriv(2,3);
79      dn4dy=deriv(2,4);
80      N1=fun(1,1);
```

```
81      N2=fun(2,1);
82      N3=fun(3,1);
83      N4=fun(4,1);
84      beem=zeros(3,24);                    % initialize membrane strain-displacement matrix
85      beem(1,1)=dn1dx;                     % could use loop for strain-displacement matrix
86      beem(1,7)=dn2dx;                     % full matrix is shown for easier verification
87      beem(1,13)=dn3dx;
88      beem(1,19)=dn4dx;
89      beem(2,2)=dn1dy;
90      beem(2,8)=dn2dy;
91      beem(2,14)=dn3dy;
92      beem(2,20)=dn4dy;
93      beem(3,1)=dn1dy;
94      beem(3,7)=dn2dy;
95      beem(3,13)=dn3dy;
96      beem(3,19)=dn4dy;
97      beem(3,2)=dn1dx;
98      beem(3,8)=dn2dx;
99      beem(3,14)=dn3dx;
100     beem(3,20)=dn4dx;
101     beeb=zeros(3,24);                    % initialize bending strain-displacement matrix
102     beeb(1,5)=-dn1dx;
103     beeb(1,11)=-dn2dx;
104     beeb(1,17)=-dn3dx;
105     beeb(1,23)=-dn4dx;
106     beeb(2,4)=dn1dy;
107     beeb(2,10)=dn2dy;
108     beeb(2,16)=dn3dy;
109     beeb(2,22)=dn4dy;
110     beeb(3,5)=-dn1dx;
111     beeb(3,11)=-dn2dx;
112     beeb(3,17)=-dn3dx;
113     beeb(3,23)=-dn4dx;
114     beeb(3,4)=dn1dy;
115     beeb(3,10)=dn2dy;
116     beeb(3,16)=dn3dy;
117     beeb(3,22)=dn4dy;
118     bees=zeros(2,24);                    % initialize shear strain-displacement matrix
119     bees(1,3)=dn1dx;
120     bees(1,5)=N1;
121     bees(1,9)=dn2dx;
122     bees(1,11)=N2;
123     bees(1,15)=dn3dx;
124     bees(1,17)=N3;
125     bees(1,21)=dn4dx;
126     bees(1,23)=N4;
127     bees(2,3)=dn1dy;
128     bees(2,4)=-N1;
129     bees(2,9)=dn2dy;
130     bees(2,10)=-N2;
131     bees(2,15)=dn3dy;
132     bees(2,16)=-N3;
133     bees(2,21)=dn4dy;
134     bees(2,22)=-N4;
135     kem=kem+det(jac)*th*w*beem'*deem*beem;          % membrane stiffness matrix
136     keb=keb+det(jac)*th*w*beeb'*deeb*beeb;          % bending stiffness matrix
137     kes=kes+det(jac)*th*w*bees'*dees*bees;          % shear stiffness matrix
138     kel=kel+kem+keb+kes;                 % combination for complete shell stiffness matrix
```

```
139 end
140 r=zeros(6);
141 r(1,1)=t1(1,1);
142 r(1,2)=t1(2,1);
143 r(1,3)=t1(3,1);
144 r(2,1)=t2(1,1);
145 r(2,2)=t2(2,1);
146 r(2,3)=t2(3,1);
147 r(3,1)=n(1,1);
148 r(3,2)=n(2,1);
149 r(3,3)=n(3,1);
150 r(4,4)=t1(1,1);
151 r(4,5)=t1(2,1);
152 r(4,6)=t1(3,1);
153 r(5,4)=t2(1,1);
154 r(5,5)=t2(2,1);
155 r(5,6)=t2(3,1);
156 r(6,4)=n(1,1);
157 r(6,5)=n(2,1);
158 r(6,6)=n(3,1);
159 T=zeros(24);
160 T(1:6,1:6)=r;
161 T(7:12,7:12)=r;
162 T(13:18,13:18)=r;
163 T(19:24,19:24)=r;
164 keg=T'*kel*T;
165
166 end                                                    % end function
```

Q4steering.m

```
1  function[coord,g] = Q4steering(i,nne,ndim,connec,ncoord,nf,ndof)
2
3  coord=zeros(nne,ndim);
4  for k=1:nne
5      for j=1:ndim
6          coord(k,j)=ncoord(connec(i,k),j);
7      end
8  end
9  p=0;
10 for iel=1:nne
11     for j=1:ndof
12         p=p+1;
13         g(p)=nf(connec(i,iel),j);
14     end
15 end
16
17 end
```

steering3D.m

```
1  function[g]=steering3D(i,elconnec,ndof)
2
3  node1=elconnec(i,1);                                % first node of element i
```

```
node2=elconnec(i,2);                                    % second node of element i
g=[ndof(node1,1);
    ndof(node1,2);
    ndof(node1,3);
    ndof(node1,4);
    ndof(node1,5);
    ndof(node1,6);
    ndof(node2,1);
    ndof(node2,2);
    ndof(node2,3);
    ndof(node2,4);
    ndof(node2,5);
    ndof(node2,6)];

end
```

Mesh Refinement Study

```
clc;
clear;
x=sym('x');
L=10/3;
nf=[0 0 0;1 2 3;4 5 6;7 8 9];
F=[0 0 0;0 0 0;0 0 0;0 10 0];
nel=3;
connec=[1 2;2 3;3 4];
nnd=4;
nodof=3;
eldof=6;
E=5000;
I=500;
A=10;
G=20;
shear=5/6;
n=9;
kglobal=zeros(n);
N1=1-x/L;
N2=x/L;
dn1dx=diff(N1,x);
dn2dx=diff(N2,x);
B = [dn1dx  0      0   dn2dx  0      0;
       0     0    dn1dx  0     0    dn2dx;
       0   dn1dx -N1     0   dn2dx  -N2];
D=zeros(3);
D(1,1)=E*A;
D(2,2)=E*I;
D(3,3)=shear*G*A;
K=int(B'*D*B,x,0,L);
for i=1:nel
    g(1,1)=nf(connec(i,1),1);
    g(2,1)=nf(connec(i,1),2);
    g(3,1)=nf(connec(i,1),3);
    g(4,1)=nf(connec(i,2),1);
    g(5,1)=nf(connec(i,2),2);
    g(6,1)=nf(connec(i,2),3);
    for j=1:eldof
```

```
39          if g(j,1)~=0
40              for k=1:eldof
41                  if g(k,1)~=0
42                      kglobal(g(j,1),g(k,1))=kglobal(g(j,1),g(k,1))+K(j,k);
43                  end
44              end
45          end
46      end
47  end
48  for i=1:nnd
49      for j=1:nodof
50          if nf(i,j)~=0
51              fglobal(nf(i,j),1)=F(i,j);
52          end
53      end
54  end
55  delta=kglobal\fglobal;
56  for i=1:nnd
57      for j=1:nodof
58          if nf(i,j)~=0
59              globaldelta(i,j)=delta(nf(i,j));
60          else
61              globaldelta(i,j)=0;
62          end
63      end
64  end
```

Test Case 1

```
1   clc;
2   clear;
3   nel=3;                                                          % number of elements
4   nnd=4;                                                          % number of nodes
5   ndof=6;                                                         % number of DOF per node
6   eldof=12;                                                       % number of DOF per element
7   elprop=[1 4e6 1 .3e6 .3e6 1e6  0 1;                             % element properties
8           1 4e6 1 .3e6 .3e6 1e6  0 1;
9           1 4e6 1 .3e6 .3e6 1e6 90 1];
10  ncoord=[0 5 5;5 5 5;5 5 0;5 0 0];                               % node coordinates
11  elconnec=[1 2;3 2;4 3];                                         % element connectivity
12  nf=[0 0 0 0 0 0;1 1 1 1 1 1;                                    % nodal DOF
13      1 1 1 1 1 1;0 0 0 0 0 0];
14  n=0;                                                % initialize system number of free DOF
15  for i=1:nnd                                         % begin counting system number of free DOF
16      for j=1:ndof
17          if nf(i,j)==1
18              n=n+1;
19              nf(i,j)=n;
20          else
21          end
22      end
23  end
24  force=zeros(nnd,ndof);                  % initialize force matrix (NOT system force vector)
25  force(2,2)=-100;                                            % assign vertical load to node 2
26  F=zeros(n,1);                                               % initialize system force vector
27  for i=1:nnd
```

```
28      for j=1:ndof
29          if nf(i,j)~=0
30              F(nf(i,j))=force(i,j);
31          else
32          end
33      end
34  end
35  K=zeros(n);                                                  % initialize system stiffness matrix
36  for i=1:nel                                                        % form system stiffness matrix
37      g=steering3D(i,elconnec,nf);
38      keg=ebbeamstiff3D(i,elconnec,elprop,ncoord);
39      K=formK(K,keg,g,eldof);
40  end
41  D=K\F;                                                       % calculate system displacement vector
42  disp=zeros(nnd,ndof);          % initialize displacement vector (NOT system displacement vector)
43  for i=1:nnd
44      for j=1:ndof
45          if nf(i,j)~=0
46              disp(i,j)=D(nf(i,j));
47          else
48              disp(i,j)=0;
49          end
50      end
51  end
52  xyzdisp=disp(:,1:3);                          % xyzdisp contains only translational displacements
53  deformcoord=ncoord+xyzdisp;                                          % deformed node coordinates
54  scaldeform=ncoord+200*xyzdisp;                                % scaled deformed node coordinates
55  for i=1:nel                                                                    % plotting Figure 5.5
56      n1=elconnec(i,1);                                                          % node 1 of element i
57      n2=elconnec(i,2);                                                          % node 2 of element i
58      line([ncoord(n1,3) ncoord(n2,3)],[ncoord(n1,1) ncoord(n2,1)]...            % plotting undeformed
59          ,[ncoord(n1,2) ncoord(n2,2)],'color','c','Marker','.'...
60          ,'MarkerSize',.5,'LineWidth',2);
61      hold on                                                                    % both on same plot
62      line([scaldeform(n1,3) scaldeform(n2,3)],[scaldeform(n1,1)...       % plotting scaled deformed
63          scaldeform(n2,1)],[scaldeform(n1,2) scaldeform(n2,2)]...
64          ,'Marker','.','MarkerSize',.5,'LineWidth',2);
65      grid on
66  end                                                          % this plot shows a linear deformation
67                                                  % actual plot would use hermitian shape functions
68  view([3,3,3.8])                                                                % plot view
```

Test Case 2

```
1   clc;
2   clear;
3   nel=1;                                                                  % number of elements
4   nnd=2;                                                                    % number of nodes
5   ndof=6;                                                               % number of DOF per node
6   eldof=12;                                                           % number of DOF per element
7   gam=.5;                                                  % newmark constant gamma from Table 2.1
8   beta=.25;                                                % newmark constant beta from Table 2.1
9   dtim=.001;                                                          % time step from Table 2.1
10  tint= 2.5;                                               % total time interval from Figure 5.7
11  nstep=tint/dtim;                                                      % number of time steps
12  fm=.001;                                    % mass rayleigh damping coefficient from Eq. (5.4)
```

```
fk=.001;                                        % stiffness rayleigh damping coefficient form Eq. (5.4)
elprop=[10 1 1 1 4000 1 1 1 100];                                          % element properties
ncoord=[0 0 0;1 0 0];                                                        % node coordinates
elconnec=[1 2];                                                          % element connectivity
nf=[0 0 0 0 0 0;0 1 0 0 0 0];                                                          % nodal DOF
n=0;                                                          % initialize system number of free DOF
for i=1:nnd                                                 % begin counting system number of free DOF
    for j=1:ndof
        if nf(i,j)==1
            n=n+1;
            nf(i,j)=n;
        else
        end
    end
end
K=zeros(n);                                                        % initialize system stiffness matrix
M=zeros(n);                                                           % initialize system mass matrix
for i=1:nel                                       % form system mass and system stiffness matrices
    g=steering3D(i,elconnec,nf);
    keg=ebbeamstiff3D(i,elconnec,elprop,ncoord);
    meg=ebbeammass3D(i,elconnec,elprop,ncoord);
    K=formK(K,keg,g,eldof);
    M=formM(M,meg,g,eldof);
end
C=fk*K+fm*M;                                                    %  form system rayleigh damping matrix
F=zeros(n,nstep);                                             % initialize force data for each time step
for i=1:nstep                                                          % form force data from Figure 5.7
    if i*dtim<=.25
        F(1,i)=40000;
    elseif i*dtim<=.5
        F(1,i)=-80000*(i*dtim)+60000;
    elseif i*dtim <= 1
        F(1,i)=20000;
    else
        F(1,i)=0;
    end
end
disp=zeros(n,nstep);                                % initialize displacement data for each time step
velo=zeros(n,nstep);                                   % initialize velocity data for each time step
acce=M\(F(:,1)-C*velo(:,1)-K*disp(:,1));             % calculate initial acceleration from Table 2.1
a0=1/(beta*(dtim^2));                                                    % constant a0 from Table 2.1
a1=(gam/beta)*(1/dtim);                                                  % constant a1 from Table 2.1
a2=(1/gam)*a1;                                                           % constant a2 from Table 2.1
a3=(.5-beta)/beta;                                                       % constant a3 from Table 2.1
a4=(gam/beta)-1;                                                         % constant a4 from Table 2.1
a5=(.5-beta)*((gam*dtim)/beta)+(gam-1)*dtim;                             % constant a5 from Table 2.1
a6=dtim*(1-gam);                                                         % constant a6 from Table 2.1
a7=gam*dtim;                                                             % constant a7 from Table 2.1
Keff=K+a0*M+a1*C;                                % initial effective stiffness matrix from Table 2.1
for i=1:nstep-1                                                        % Newmark method from Table 2.1
    Feff=M*(a0*disp(:,i)+a2*velo(:,i)+a3*acce(:,i))...
            +C*(a1*disp(:,i)+a4*velo(:,i)...
            +a5*acce(:,i))+F(:,i+1);
    disp(:,i+1)=Keff\Feff;
    acce(:,i+1)=a0*(disp(:,i+1)-disp(:,i))-a2*velo(:,i)-a3*acce(:,i);
    velo(:,i+1)=velo(:,i)+a6*acce(:,i)+a7*acce(:,i+1);
end
y1=disp(1,:);
```

```
x1=linspace(0,2.5,2500);
plot(x1,y1)                                % plotting Rayleigh Damping portion of Figure 5.8
xlabel('Time (sec)')
ylabel('Free End Deflection')
```

Test Case 3

```
clc;
clear;
nel=45;
nnd=64;
nne=4;
ndof=2;
eldof=nne*ndof;
prop=zeros(nel,3);
for i=1:nel
    prop(i,1)=200000;
    prop(i,2)=.3;
    prop(i,3)=5;
end
ncoord=[0    -3;
        0    -1;
        0     1;
        0     3;
        2    -3;
        2    -1;
        2     1;
        2     3;
        4    -3;
        4    -1;
        4     1;
        4     3;
        6    -3;
        6    -1;
        6     1;
        6     3;
        8    -3;
        8    -1;
        8     1;
        8     3;
       10    -3;
       10    -1;
       10     1;
       10     3;
       12    -3;
       12    -1;
       12     1;
       12     3;
       14    -3;
       14    -1;
       14     1;
       14     3;
       16    -3;
       16    -1;
       16     1;
       16     3;
```

```
50          18      -3;
51          18      -1;
52          18       1;
53          18       3;
54          20      -3;
55          20      -1;
56          20       1;
57          20       3;
58          22      -3;
59          22      -1;
60          22       1;
61          22       3;
62          24      -3;
63          24      -1;
64          24       1;
65          24       3;
66          26      -3;
67          26      -1;
68          26       1;
69          26       3;
70          28      -3;
71          28      -1;
72          28       1;
73          28       3;
74          30      -3;
75          30      -1;
76          30       1;
77          30       3];
78    connec=[1       5       6       2;
79          2       6       7       3;
80          3       7       8       4;
81          5       9      10       6;
82          6      10      11       7;
83          7      11      12       8;
84          9      13      14      10;
85         10      14      15      11;
86         11      15      16      12;
87         13      17      18      14;
88         14      18      19      15;
89         15      19      20      16;
90         17      21      22      18;
91         18      22      23      19;
92         19      23      24      20;
93         21      25      26      22;
94         22      26      27      23;
95         23      27      28      24;
96         25      29      30      26;
97         26      30      31      27;
98         27      31      32      28;
99         29      33      34      30;
100        30      34      35      31;
101        31      35      36      32;
102        33      37      38      34;
103        34      38      39      35;
104        35      39      40      36;
105        37      41      42      38;
106        38      42      43      39;
107        39      43      44      40;
```

```
108         41      45      46      42;
109         42      46      47      43;
110         43      47      48      44;
111         45      49      50      46;
112         46      50      51      47;
113         47      51      52      48;
114         49      53      54      50;
115         50      54      55      51;
116         51      55      56      52;
117         53      57      58      54;
118         54      58      59      55;
119         55      59      60      56;
120         57      61      62      58;
121         58      62      63      59;
122         59      63      64      60];
123  nf=ones(nnd,ndof);
124  nf(1,1)=0;
125  nf(1,2)=0;
126  nf(2,1)=0;
127  nf(2,2)=0;
128  nf(3,1)=0;
129  nf(3,2)=0;
130  nf(4,1)=0;
131  nf(4,2)=0;
132  n=0;
133  for i=1:nnd
134      for j=1:ndof
135          if nf(i,j)==1
136              n=n+1;
137              nf(i,j)=n;
138          else
139          end
140      end
141  end
142  force=zeros(nnd,ndof);
143  force(64,2)=-5000;
144  F=zeros(n,1);
145  K=zeros(n);
146  for i=1:nnd
147      for j=1:ndof
148          if nf(i,j)~=0
149              F(nf(i,j))=force(i,j);
150          else
151          end
152      end
153  end
154  for i=1:nel
155      [coord,g]=Q4steering(i,nne,ndof,connec,ncoord,nf,ndof);
156      keg=Q4membranestiff(i,coord,prop,eldof);
157      K=formK(K,keg,g,eldof);
158  end
159  D=K\F;                                                    % unknown displacements
160  disp=zeros(nnd,ndof);
161  for i=1:nnd
162      for j=1:ndof
163          if nf(i,j)~=0
164              disp(i,j)=D(nf(i,j));
165          else
```

```
166                 disp(i,j)=0;
167             end
168         end
169     end
170     deformcoord=disp+ncoord;
171     uy=disp(:,2);
172     patch('Faces',connec,'Vertices',deformcoord,'FaceVertexCData',uy,...
173           'Facecolor','interp','Marker','.');
174     colorbar;
```

Test Case 4

```
1   clc;
2   clear;
3   nne=4;
4   ndof=6;
5   ndim=3;
6   eldof=nne*ndof;
7   xlength=18;
8   ylength=18;
9   nelx=20;
10  nely=20;
11  dhx=xlength/nelx;
12  dhy=ylength/nely;
13  xorigin=0;
14  yorigin=ylength/2;
15  [ncoord,connec,nnd,nel]=Q4mesh(nelx,nely,dhx,dhy,xorigin,yorigin);
16  nf=ones(nnd,ndof);
17  for i=1:nel
18      prop(i,1)=30e6;
19      prop(i,2)=.3;
20      prop(i,3)=.25;
21      prop(i,4)=300;
22  end
23  for i=1:nnd
24      nf(i,1)=0;
25      nf(i,2)=0;
26      nf(i,6)=0;
27      if ncoord(i,2)==-9 || ncoord(i,2)==9
28          nf(i,3)=0;
29          nf(i,5)=0;
30      elseif ncoord(i,1)==0 || ncoord(i,1)==18
31          nf(i,4)=0;
32          nf(i,3)=0;
33      end
34  end
35  n=0;
36  for i=1:nnd
37      for j=1:ndof
38          if nf(i,j)==1
39              n=n+1;
40              nf(i,j)=n;
41          else
42          end
43      end
44  end
```

```
force=zeros(nnd,ndof);
for i=1:nnd
        if ncoord(i,1)==9 && ncoord(i,2)==0
            force(i,3)=500;
        end
end
F=zeros(n);
for i=1:nnd
    for j=1:ndof
        if nf(i,j)~=0
            F(nf(i,j))=force(i,j);
        else
        end
    end
end
K=zeros(n);
M=zeros(n);
for i=1:nel
    [coord,g]=Q4steering(i,nne,ndim,connec,ncoord,nf,ndof);
    keg=Q4shellstiff(i,coord,prop,eldof);
    K=formK(K,keg,g,eldof);
end
D=K\F;
disp=zeros(nnd,ndof);
for i=1:nnd
    for j=1:ndof
        if nf(i,j)~=0
            disp(i,j)=D(nf(i,j));
        else
            disp(i,j)=0;
        end
    end
end
deformcoord=disp(:,1:3)+ncoord;
uz = disp(:,3);
patch('Faces', connec,'Vertices',deformcoord,'FaceVertexCData',uz,...
      'Facecolor','interp','Marker','.');
colorbar;
grid on
view([3,3,3])
```

Mesh Refinement Study

```
clc;
clear;
x=sym('x');
L=10/3;
nf=[0 0 0;1 2 3;4 5 6;7 8 9];
F=[0 0 0;0 0 0;0 0 0;0 10 0];
nel=3;
connec=[1 2;2 3;3 4];
nnd=4;
nodof=3;
eldof=6;
E=5000;
I=500;
```

```
A=10;
G=20;
shear=5/6;
n=9;
kglobal=zeros(n);
N1=1-x/L;
N2=x/L;
dn1dx=diff(N1,x);
dn2dx=diff(N2,x);
B = [dn1dx   0      0   dn2dx   0      0;
       0     0    dn1dx   0     0    dn2dx;
       0   dn1dx -N1     0   dn2dx  -N2];
D=zeros(3);
D(1,1)=E*A;
D(2,2)=E*I;
D(3,3)=shear*G*A;
K=int(B'*D*B,x,0,L);
for i=1:nel
    g(1,1)=nf(connec(i,1),1);
    g(2,1)=nf(connec(i,1),2);
    g(3,1)=nf(connec(i,1),3);
    g(4,1)=nf(connec(i,2),1);
    g(5,1)=nf(connec(i,2),2);
    g(6,1)=nf(connec(i,2),3);
    for j=1:eldof
        if g(j,1)~=0
            for k=1:eldof
                if g(k,1)~=0
                    kglobal(g(j,1),g(k,1))=kglobal(g(j,1),g(k,1))+K(j,k);
                end
            end
        end
    end
end
for i=1:nnd
    for j=1:nodof
        if nf(i,j)~=0
            fglobal(nf(i,j),1)=F(i,j);
        end
    end
end
delta=kglobal\fglobal;
for i=1:nnd
    for j=1:nodof
        if nf(i,j)~=0
            globaldelta(i,j)=delta(nf(i,j));
        else
            globaldelta(i,j)=0;
        end
    end
end
```

Higher-Order Shape Function Study

```
clc;
clear;
```

```
eldof=3;
nf=[0 1 2];
x=sym('x');
x5=sym('x5');
x1=0;
x2=5;
x3=10;
A=(x^2)/450+2;
E=5000;
F=[0;0;10];
b=-3*x;
N1=((x-x2)*(x-x3))/((x1-x2)*(x1-x3));
N2=((x-x1)*(x-x3))/((x2-x1)*(x2-x3));
N3=((x-x1)*(x-x2))/((x3-x1)*(x3-x2));
N=[N1 N2 N3];
B=diff(N,x);
K=int(B'*E*A*B,x,x1,x3);
f1=(int(N1*b,x,x1,x3))+subs(N1,x3)*F(3,1);
f2=(int(N2*b,x,x1,x3))+subs(N2,x3)*F(3,1);
f3=(int(N3*b,x,x1,x3))+subs(N3,x3)*F(3,1);
F=[f1;f2;f3];
for i=1:eldof
    if nf(i)~=0
        fglobal(nf(i),1)=F(i);
    end
end
for i=1:eldof
    if nf(i)~=0
        for j=1:eldof
            if nf(j)~=0
                kglobal(nf(i),nf(j))=K(i,j);
            end
        end
    end
end
delta=kglobal\fglobal;
for i=1:eldof
    if nf(i)~=0
        globaldelta(i,1)=delta(nf(i));
    else
        globaldelta(i,1)=0;
    end
end
y=(-3/200)*(149*pi-9*x-298*acot(x/30));
%sigmaexact=E*((9*(3*(x5.^2)-280))/(200*((x5.^2)+900)));
x=linspace(0,10,20);
y=subs(y,x);
plot(x,y,'LineWidth',2)
hold on
y=N*globaldelta;
%sigma3=E*B*globaldelta;
x=linspace(0,10,20);
y=subs(y,x);
plot(x,y,'--k')
clear;
x=sym('x');
eldof=2;
nf=[0 1];
```

```
n=2;
x1=0;
x2=10;
A=(x^2)/450+2;
E=5000;
F=[0;10];
b=-3*x;
N1=(x-x2)/(x1-x2);
N2=(x-x1)/(x2-x1);
N=[N1 N2];
B=diff(N,x);
K=int(B'*E*A*B,x,x1,x2);
f1=(int(N1*b,x,x1,x2))+subs(N1,x2)*F(2,1);
f2=(int(N2*b,x,x1,x2))+subs(N2,x2)*F(2,1);
F=[f1;f2];
for i=1:eldof
    if nf(i)~=0
        fglobal(nf(i),1)=F(i);
    end
end
for i=1:eldof
    if nf(i)~=0
        for j=1:eldof
            if nf(j)~=0
                kglobal(nf(i),nf(j))=K(i,j);
            end
        end
    end
end
delta=kglobal\fglobal;
for i=1:eldof
    if nf(i) ~= 0
        globaldelta(i,1)=delta(nf(i));
    else
        globaldelta(i,1) = 0;
    end
end
y=N*globaldelta;
%sigma2=E*B*globaldelta;
x=linspace(0,10,20);
y=subs(y,x);
plot(x,y,'--r')
hold off
h=legend({'Analytical','Quadratic Element','Linear Element'},'Location','northeast');
xlabel('Length(in)','FontSize',13)
ylabel('Displacement(in)','FontSize',13)
```

B. Fortran Code

Because the Fortran code is similar to the MATLAB code, only a few examples are provided.

Fortran Module

main.m

```
1  MODULE main
2  !
3  INTERFACE
4  !
5  SUBROUTINE ebbeammass3D(eldof,i,prop,coord,bme)
6      IMPLICIT NONE
7      INTEGER,INTENT(IN)::eldof,i
8      REAL,DIMENSION(:,:),INTENT(IN)::prop,coord
9      REAL,DIMENSION(:,:),INTENT(OUT)::bme
10 END SUBROUTINE ebbeammass3D
11 !
12 SUBROUTINE ebbeamstiff3D(i,coord,prop,eldof,ke,hinge)
13     IMPLICIT NONE
14     INTEGER,INTENT(IN)::eldof,i
15     INTEGER,DIMENSION(:,:),INTENT(IN)::hinge
16     REAL,DIMENSION(:,:),INTENT(IN)::prop,coord
17     REAL,DIMENSION(:,:),INTENT(OUT)::ke
18 END SUBROUTINE ebbeamstiff3D
19 !
20 SUBROUTINE Q4steering(i,nne,nodof,nf,connec,g,ncoord,coord,ndim)
21     IMPLICIT NONE
22     INTEGER,INTENT(IN)::i,nne,nodof,ndim
23     INTEGER,DIMENSION(:,:),INTENT(IN)::nf,connec
24     REAL,DIMENSION(:,:),INTENT(IN)::ncoord
25     INTEGER,DIMENSION(:,:),INTENT(OUT)::g
26     REAL,DIMENSION(:,:),INTENT(OUT)::coord
27 END SUBROUTINE Q4steering
28 !
29 SUBROUTINE gausspoints(ngp,gauss)
30     IMPLICIT NONE
31     INTEGER,INTENT(IN)::ngp
32     REAL,DIMENSION(:,:),INTENT(OUT)::gauss
33 END SUBROUTINE gausspoints
34 !
35 SUBROUTINE Q4shellstiff(i,coord,nodof,nne,ngp,prop,gauss,keg)
36     IMPLICIT NONE
```

```
37        INTEGER,INTENT(IN)::i,nodof,nne,ngp
38        REAL,DIMENSION(:,:),INTENT(IN)::coord,prop,gauss
39        REAL,DIMENSION(:,:),INTENT(OUT)::keg
40    END SUBROUTINE Q4shellstiff
41    !
42    SUBROUTINE Q8shellstiff(i,coord,nodof,nne,ngp,prop,gauss,keg)
43        IMPLICIT NONE
44        INTEGER,INTENT(IN)::i,nodof,nne,ngp
45        REAL,DIMENSION(:,:),INTENT(IN)::coord,prop,gauss
46        REAL,DIMENSION(:,:),INTENT(OUT)::keg
47    END SUBROUTINE Q8shellstiff
48    !
49    SUBROUTINE globalstiffness(kglobal,mglobal,bme,ke,g,eldof)
50        IMPLICIT NONE
51        REAL,DIMENSION(:,:),INTENT(IN OUT)::kglobal,mglobal
52        INTEGER,INTENT(IN)::eldof
53        INTEGER,DIMENSION(:,:),INTENT(IN)::g
54        REAL,DIMENSION(:,:),INTENT(IN)::ke,bme
55    END SUBROUTINE globalstiffness
56    !
57    SUBROUTINE Q4shellmass(i,coord,nodof,nne,ngp,prop,gauss,meg)
58        IMPLICIT NONE
59        INTEGER,INTENT(IN)::i,nodof,nne,ngp
60        REAL,DIMENSION(:,:),INTENT(IN)::coord,prop,gauss
61        REAL,DIMENSION(:,:),INTENT(OUT)::meg
62    END SUBROUTINE Q4shellmass
63    !
64    SUBROUTINE steering3D(i,nf,bconnec,g)
65        IMPLICIT NONE
66        INTEGER,INTENT(IN)::i
67        INTEGER,DIMENSION(:,:),INTENT(IN)::nf,bconnec
68        INTEGER,DIMENSION(:),INTENT(OUT)::g
69    END SUBROUTINE steering3D
70    !
71    SUBROUTINE formM(mglobal,bme,g,eldof)
72        IMPLICIT NONE
73        INTEGER,INTENT(IN)::eldof
74        INTEGER,DIMENSION(:,:),INTENT(IN)::g
75        REAL,DIMENSION(:,:),INTENT(IN)::bme
76        REAL,DIMENSION(:,:),INTENT(IN OUT)::mglobal
77    END SUBROUTINE formM
78    !
79    SUBROUTINE formK(kglobal,ke,g,eldof)
80        IMPLICIT NONE
81        INTEGER,INTENT(IN)::eldof
82        INTEGER,DIMENSION(:,:),INTENT(IN)::g
83        REAL,DIMENSION(:,:),INTENT(IN)::ke
84        REAL,DIMENSION(:,:),INTENT(IN OUT)::kglobal
85    END SUBROUTINE formK
86    !
87    END INTERFACE
88    !
89    END MODULE main
```

Fortran Subroutines

formK.f

```
1  SUBROUTINE formK(kglobal,ke,g,eldof)
2  !-----------------------------------------------------------------------
3  ! This subroutine forms the global stiffness matrix for different elements.
4  ! The global stiffness array (kglobal) is a n by n matrix where n is the
5  ! total free degrees of freedom in the model.
6  !-----------------------------------------------------------------------
7  IMPLICIT NONE
8  INTEGER,INTENT(IN)::eldof
9  INTEGER,DIMENSION(:,:),INTENT(IN)::g
10 REAL,DIMENSION(:,:),INTENT(IN)::ke
11 REAL,DIMENSION(:,:),INTENT(IN OUT)::kglobal
12 INTEGER::i,j
13
14 DO i=1,eldof
15      IF(g(i,1)/=0)THEN
16           DO j=1,eldof
17                IF(g(j,1)/=0)THEN
18                     kglobal(g(i,1),g(j,1))=kglobal(g(i,1),g(j,1))+ke(i,j)
19                END IF
20           END DO
21      END IF
22 END DO
23
24 END SUBROUTINE formK
```

gausselimination.f

```
1  SUBROUTINE gausselimination(kglobal,delta,force,error)
2
3  IMPLICIT NONE
4  REAL,DIMENSION(:),INTENT(IN)::force
5  REAL,DIMENSION(:,:),INTENT(IN)::kglobal
6  REAL,DIMENSION(:),INTENT(OUT)::delta
7  REAL,DIMENSION(:,:),ALLOCATABLE::m
8  REAL,DIMENSION(:),ALLOCATABLE::f
9  LOGICAL,INTENT(OUT)::error
10 INTEGER::i,j,iel,neq,k
11 REAL::L
12 !-----------------------------------------------------------------------
13 ! kglobal is the global stiffness matrix
14 !-----------------------------------------------------------------------
15 error=size(kglobal,dim=1)/=size(force) .or.                              &
16      size(kglobal,dim=2)/=size(force)
17 IF(error)THEN
18      delta=0.0
19      RETURN
20 END IF
21 neq=size(kglobal,dim=1)
22 ALLOCATE(m(neq,neq),f(neq))
23 m(1:neq,1:neq)=kglobal
24 f(1:neq)=force
25 !-----------------------------------------------------------------------
```

```
26  ! Forming the upper triangular.
27  !-------------------------------------------------------------------------------
28  DO i=1,neq-1
29       k=k+1
30       DO j=i,neq-1
31            L=m(j+1,i)/m(i,i)
32            DO iel=1,neq
33                 m(j+1,iel)=m(j+1,iel)-L*m(i,iel)
34            END DO
35            f(j+1)=f(j+1)-L*f(i)
36       END DO
37  END DO
38  !-------------------------------------------------------------------------------
39  ! Back subsitution phase.
40  !-------------------------------------------------------------------------------
41  DO i=neq,1,-1
42       delta(i)=(f(i)-sum(m(i,i+1:neq)*delta(i+1:neq)))/(m(i,i))
43  END DO
44
45  END SUBROUTINE gausselimination
```

ebbeamstiff3D.f

```
1   SUBROUTINE ebbeamstiff3D(i,coord,prop,eldof,ke,hinge)
2   !-------------------------------------------------------------------------------
3   ! This subroutine forms the stiffness matrix for a beam-column element in three-
4   ! dimensional (3-D) space. It includes the stiffness matrices for the presence
5   ! of a spherical hinge at either or both ends.
6   !-------------------------------------------------------------------------------
7   IMPLICIT NONE
8   INTEGER,INTENT(IN)::eldof,i
9   INTEGER,DIMENSION(:,:),INTENT(IN)::hinge
10  REAL,DIMENSION(:,:),INTENT(IN)::prop,coord
11  REAL,DIMENSION(:,:),INTENT(OUT)::ke
12  REAL,DIMENSION(eldof,eldof)::T
13  REAL,DIMENSION(3,3)::r
14  REAL::x1,y1,z1,x2,y2,z2,L,A,E,J,Iz,Iy,G,cx,cy,cz,s1,s2,s3,s4,s5,s6,s7,s8,s9,    &
15       s10,s11,s12,s13,s14,s15,s16,pi,psirad,psi,cg,sg,den
16  INTEGER::z
17  x1=coord(1,1)
18  y1=coord(1,2)
19  z1=coord(1,3)
20  x2=coord(2,1)
21  y2=coord(2,2)
22  z2=coord(2,3)
23  A=prop(i,1)
24  G=prop(i,2)
25  E=prop(i,3)
26  Iy=prop(i,4)
27  Iz=prop(i,5)
28  J=prop(i,6)
29  psi=prop(i,7)
30  pi=acos(-1.0)
31  psirad=((psi*pi)/(180.0))
32  L=sqrt((x2-x1)**2+(y2-y1)**2+(z2-z1)**2)
33  !-------------------------------------------------------------------------------
34  ! Forming the element stiffness matrix with rigid ends.
```

```
35 !-------------------------------------------------------------------------------
36 IF(hinge(1,1)==0 .and. hinge(1,2)==0)THEN
37       s1=(E*A)/(L);
38       s2=(12*E*Iz)/(L**3);
39       s3=(6*E*Iz)/(L**2);
40       s4=(12*E*Iy)/(L**3);
41       s5=(6*E*Iy)/(L**2);
42       s6=(G*J)/(L);
43       s7=(4*E*Iy)/(L);
44       s8=(2*E*Iy)/(L);
45       s9=(4*E*Iz)/(L);
46       s10=(2*E*Iz)/(L);
47       s11=-s1
48       s12=-s2
49       s13=-s4
50       s14=-s5
51       s15=-s6
52       s16=-s3
53       ke=0.0
54       ke(1,1)=s1
55       ke(1,7)=s11
56       ke(2,2)=s2
57       ke(2,6)=s3
58       ke(2,8)=s12
59       ke(2,12)=s3
60       ke(3,3)=s4
61       ke(3,5)=s14
62       ke(3,9)=s13
63       ke(3,11)=s14
64       ke(4,4)=s6
65       ke(4,10)=s15
66       ke(5,3)=s14
67       ke(5,5)=s7
68       ke(5,9)=s5
69       ke(5,11)=s8
70       ke(6,2)=s3
71       ke(6,6)=s9
72       ke(6,8)=s16
73       ke(6,12)=s10
74       ke(7,1)=s11
75       ke(7,7)=s1
76       ke(8,2)=s12
77       ke(8,6)=s16
78       ke(8,8)=s2
79       ke(8,12)=s16
80       ke(9,3)=s13
81       ke(9,5)=s5
82       ke(9,9)=s4
83       ke(9,11)=s5
84       ke(10,4)=s15
85       ke(10,10)=s6
86       ke(11,3)=s14
87       ke(11,5)=s8
88       ke(11,9)=s5
89       ke(11,11)=s7
90       ke(12,2)=s3
91       ke(12,6)=s10
92       ke(12,8)=s16
```

```
93        ke(12,12)=s9
94  !-----------------------------------------------------------------------
95  ! Forming the element stiffness matrix with a hinge at its left end.
96  !-----------------------------------------------------------------------
97  ELSEIF(hinge(1,1)==1 .and. hinge(1,2)==0)THEN
98        s1=(E*A)/(L);
99        s2=(3*E*Iz)/(L**3);
100       s3=(3*E*Iz)/(L**2);
101       s4=(3*E*Iz)/(L);
102       s5=(3*E*Iy)/(L**3);
103       s6=(3*E*Iy)/(L**2);
104       s7=(3*E*Iy)/(L);
105       s8=-s1;
106       s9=-s2;
107       s10=-s3;
108       s11=-s5;
109       s12=-s6;
110       ke=0.0
111       ke(1,1)=s1
112       ke(1,7)=s8
113       ke(2,2)=s2
114       ke(2,8)=s9
115       ke(2,12)=s3
116       ke(3,3)=s5
117       ke(3,9)=s11
118       ke(3,11)=s12
119       ke(7,1)=s8
120       ke(7,7)=s1
121       ke(8,2)=s9
122       ke(8,8)=s2
123       ke(8,12)=s10
124       ke(9,3)=s11
125       ke(9,9)=s5
126       ke(9,11)=s6
127       ke(11,3)=s12
128       ke(11,9)=s6
129       ke(11,11)=s7
130       ke(12,2)=s3
131       ke(12,8)=s10
132       ke(12,12)=s4
133 !-----------------------------------------------------------------------
134 ! Forming the element stiffness matrix with a hinge at its right end.
135 !-----------------------------------------------------------------------
136 ELSEIF(hinge(1,1)==0 .and. hinge(1,2)==1)THEN
137       s1=(E*A)/(L);
138       s2=(3*E*Iz)/(L**3);
139       s3=(3*E*Iz)/(L**2);
140       s4=(3*E*Iz)/(L);
141       s5=(3*E*Iy)/(L**3);
142       s6=(3*E*Iy)/(L**2);
143       s7=(3*E*Iy)/(L);
144       s8=-s1;
145       s9=-s2;
146       s10=-s3;
147       s11=-s5;
148       s12=-s6;
149       ke=0.0
150       ke(1,1)=s1
```

```
151         ke(1,7)=s8
152         ke(2,2)=s2
153         ke(2,6)=s3
154         ke(2,8)=s9
155         ke(3,3)=s5
156         ke(3,5)=s12
157         ke(3,9)=s11
158         ke(5,3)=s12
159         ke(5,5)=s7
160         ke(5,9)=s6
161         ke(6,2)=s3
162         ke(6,6)=s4
163         ke(6,8)=s10
164         ke(7,1)=s8
165         ke(7,7)=s1
166         ke(8,2)=s9
167         ke(8,6)=s10
168         ke(8,8)=s2
169         ke(9,3)=s11
170         ke(9,5)=s6
171         ke(9,9)=s5
172   !-------------------------------------------------------------------------------
173   ! Forming the element stiffness matrix with both ends hinged.
174   !-------------------------------------------------------------------------------
175   ELSEIF(hinge(1,1)==1 .and. hinge(1,2)==1)THEN
176         s1=(E*A)/(L)
177         s2=-s1
178         ke=0.0
179         ke(1,1)=s1
180         ke(1,7)=s2
181         ke(7,1)=s2
182         ke(7,7)=s1
183   END IF
184   !-------------------------------------------------------------------------------
185   ! Forming transformation matrix to transform  the element local stiffness
186   ! matrix to the element global stiffness matrix.
187   !-------------------------------------------------------------------------------
188   sg=sin(psirad)
189   cg=cos(psirad)
190   IF(x2/=x1 .and. z2/=z1)THEN
191         T=0.0
192         T(1,2)=1.0
193               T(2,1)=-cg
194               T(2,3)=sg
195               T(3,1)=sg
196               T(3,3)=cg
197               T(4,5)=T(1,2)
198               T(5,4)=T(2,1)
199               T(5,6)=T(2,3)
200               T(6,4)=T(3,1)
201               T(6,6)=T(3,3)
202               T(7,8)=T(1,2)
203               T(8,7)=T(2,1)
204               T(8,9)=T(2,3)
205               T(9,7)=T(3,1)
206               T(9,9)=T(3,3)
207               T(10,11)=T(1,2)
208               T(11,10)=T(2,1)
```

```
209             T(11,12)=T(2,3)
210             T(12,10)=T(3,1)
211             T(12,12)=T(3,3)
212     ELSE
213             T=0.0
214             cx=(x2-x1)/L
215             cy=(y2-y1)/L
216             cz=(z2-z1)/L
217             den=sqrt((cx**2)+(cz**2))
218             T(1,1)=cx
219             T(1,2)=cy
220             T(1,3)=cz
221             T(2,1)=(-Cx*Cy*cg-Cz*sg)/den;
222             T(2,2)=den*cg;
223             T(2,3)=(-Cy*Cz*cg+Cx*sg)/den;
224             T(3,1)=(Cx*Cy*sg-Cz*cg)/den;
225             T(3,2)=-den*sg;
226             T(3,3)=(Cy*Cz*sg+Cx*cg)/den;
227             T(4,4)=T(1,1)
228             T(4,5)=T(1,2)
229             T(4,6)=T(1,3)
230             T(5,4)=T(2,1)
231             T(5,5)=T(2,2)
232             T(5,6)=T(2,3)
233             T(6,4)=T(3,1)
234             T(6,5)=T(3,2)
235             T(6,6)=T(3,3)
236             T(7,7)=T(1,1)
237             T(7,8)=T(1,2)
238             T(7,9)=T(1,3)
239             T(8,7)=T(2,1)
240             T(8,8)=T(2,2)
241             T(8,9)=T(2,3)
242             T(9,7)=T(3,1)
243             T(9,8)=T(3,2)
244             T(9,9)=T(3,3)
245             T(10,10)=T(1,1)
246             T(10,11)=T(1,2)
247             T(10,12)=T(1,3)
248             T(11,10)=T(2,1)
249             T(11,11)=T(2,2)
250             T(11,12)=T(2,3)
251             T(12,10)=T(3,1)
252             T(12,11)=T(3,2)
253             T(12,12)=T(3,3)
254     END IF
255     ke=matmul(transpose(T),ke)
256     ke=matmul(ke,T)
257     END SUBROUTINE ebbeamstiff3D
```

C. 3-D Stiffness Matrix Derivation

The stiffness matrix for a Euler-Bernoulli beam element can be derived using the transfer matrix method. The transfer matrix relates the field variables at the end nodes to one another. Once the transfer matrix is formed, a relation is derived to form the stiffness matrix. The method presented formed the transfer matrix by direct integration of the governing differential equations for a E-B beam. The governing differential equations are broken into the differential equations for bending in the X-Z plane, bending in the X-Y plane, axial extension, and torsional rotation. Following integration of the differential equations, the transfer matrix is formed, and a relation is derived to form the 3-D stiffness matrix. The derivation for a 3-D beam presented herein extends the derivation for a 2-D beam presented in [35]. The governing differential equations for a Euler-Bernoulli beam were found in [35].

Bending in the X-Z Plane:

The governing differential equations for bending of a E-B beam in the X-Z plane are

$$\begin{aligned} \frac{dw}{dx} &= -\theta_y \\ \frac{d^2w}{dx^2} &= -\frac{M_y}{EI_y} \\ \frac{d^3w}{dx^3} &= -\frac{V_z}{EI_y} \\ \frac{d^4w}{dx^4} &= \frac{p_z}{EI_y} \end{aligned} \tag{C.1}$$

Integrating the fourth order differential equation four times gives

$$\begin{aligned} V_z &= -EI_y \frac{d^3w}{dx^3} = -C_1 - \int p_z(x)\, dx \\ M_y &= -EI \frac{d^2w}{dx^2} = -C_2 - C_1 x - \iint p_z(x)\, dx \\ \theta_y &= -\frac{dw}{dx} = -\frac{C_3}{EI_y} - \frac{C_2 x}{EI_y} - \frac{C_1 x^2}{2EI_y} - \iiint \frac{p_z(x)}{EI_y}\, dx \\ w &= \frac{C_4}{EI_y} + \frac{C_3 x}{EI_y} + \frac{C_2 x^2}{2EI_y} + \frac{C_1 x^3}{6EI_y} + \iiiint \frac{p_z(x)}{EI_y}\, dx \end{aligned} \tag{C.2}$$

To obtain values for constants in terms of the nodal variables, Eq. (C.2) is solved with $x = 0$, and it is assumed that there is no loading at $x = 0$, that is, $p_z(0) = 0$. Implementing these constraints gives the constants of integration in terms of the nodal variables at node 1:

$$\begin{aligned} C_1 &= -V_{z1} \\ C_2 &= -M_{y1} \\ C_3 &= -EI_y \theta_{y1} \\ C_4 &= EI_y w_1 \end{aligned} \tag{C.3}$$

Substituting the constants of integration Eq. (C.3) into Eq. (C.2) and setting $x = L$ gives

$$\begin{aligned} w_2 &= w_1 - \theta_{y1} L - V_{z1} \frac{L^3}{6EI_y} - M_{y1} \frac{L^2}{2EI_y} + \iiiint \frac{p_z(x)}{EI_y}\, dx \\ \theta_{y2} &= \theta_{y1} + V_{z1} \frac{L^2}{2EI_y} + M_{y1} \frac{L}{EI_y} - \iiint \frac{p_z(x)}{EI_y}\, dx \\ V_{z2} &= V_{z1} - \int \frac{p_z(x)}{EI_y}\, dx \\ M_{y2} &= V_{z1} L + M_{y1} - \iint \frac{p_z(x)}{EI_y}\, dx \end{aligned} \tag{C.4}$$

Bending in the X-Y Plane:

The governing differential equations for bending of a E-B beam in the X-Y plane are

$$\begin{aligned} \frac{dv}{dx} &= \theta_z \\ \frac{d^2v}{dx^2} &= \frac{M_z}{EI_z} \\ \frac{d^3v}{dx^3} &= -\frac{V_y}{EI_z} \\ \frac{d^4v}{dx^4} &= \frac{p_y}{EI_z} \end{aligned} \tag{C.5}$$

Integrating the fourth order differential equation four times gives

$$\begin{aligned} V_y &= -EI_z\frac{d^3v}{dx^3} = -C_1 - \int p_y(x)\ dx \\ M_z &= EI_z\frac{d^2v}{dx^2} = C_2 + C_1x + \iint p_y(x)\ dx \\ \theta_z &= \frac{dv}{dx} = \frac{C_3}{EI_z} + \frac{C_2x}{EI_z} + \frac{C_1x^2}{2EI_z} + \iiint \frac{p_y(x)}{EI_z}\ dx \\ v &= \frac{C_4}{EI_z} + \frac{C_3x}{EI_z} + \frac{C_2x^2}{2EI_z} + \frac{C_1x^3}{6} + \iiiint \frac{p_z(x)}{EI_y}\ dx \end{aligned} \tag{C.6}$$

To obtain values for constants in terms of the nodal variables, Eq. (C.6) is solved with $x = 0$, and it is assumed that there is no loading at $x = 0$, that is, $p_y(0) = 0$. Implementing these constraints gives the constants of integration in terms of the nodal variables at node 1:

$$\begin{aligned} C_1 &= -V_{y1} \\ C_2 &= M_{z1} \\ C_3 &= EI_z\theta_{z1} \\ C_4 &= EI_zv_1 \end{aligned} \tag{C.7}$$

Substituting the constants of integration Eq. (C.7) into Eq. (C.6) and setting $x = L$ gives

$$\begin{aligned} v_2 &= v_1 + \theta_{z1}L - V_{y1}\frac{L^3}{6EI_z} + M_{z1}\frac{L^2}{2EI_z} + \iiint\!\!\iint p_y(x)\ dx \\ \theta_{z2} &= \theta_{z1} - V_{y1}\frac{L^2}{2EI_z} + M_{z1}\frac{L}{EI_z} + \iiint \frac{p_y(x)}{EI_z}\ dx \\ V_{y2} &= V_{y1} - \int p_y(x)\ dx \\ M_{z2} &= -V_{y1}L + M_{z1} + \iint p_y(x)\ dx \end{aligned} \tag{C.8}$$

Axial Extension:

The governing differential equations for axial extension of a beam are

$$\begin{aligned} \frac{du}{dx} &= \frac{N}{EA} \\ \frac{dN}{dx} &= -p_x \end{aligned} \tag{C.9}$$

Integration of Eq. (C.9) gives

$$\begin{aligned} N &= C_1 - \int p_x\ dx \\ u &= C_2 + \frac{C_1 x}{EA} - \iint \frac{p_x}{EA}\ dx \end{aligned} \tag{C.10}$$

To obtain values for constants in terms of the nodal variables, Eq. (C.10) is solved with $x = 0$, and it is assumed that there is no loading at $x = 0$, that is, $p_x(0) = 0$. Implementing these constraints gives the constants of integration in terms of the nodal variables at node 1:

$$\begin{aligned} C_1 &= N_1 \\ C_2 &= u_1 \end{aligned} \tag{C.11}$$

Substituting the constants of integration Eq. (C.11) into Eq. (C.10) gives

$$\begin{aligned} N_2 &= N_1 - \int p_x\ dx \\ u_2 &= u_1 + N_1\frac{L}{EA} - \iint \frac{p_x}{EA}\ dx \end{aligned} \tag{C.12}$$

Torsional Rotation:

The governing differential equations for torsional rotation of a beam are

$$\begin{aligned} \frac{d\phi}{dx} &= \frac{M_t}{GJ} \\ \frac{dM_t}{dx} &= -m_t \end{aligned} \tag{C.13}$$

Integration of Eq. (C.13) gives

$$\begin{aligned} M_t &= C_1 - \int m_t dx \\ \phi &= C_2 + \frac{C_1 x}{GJ} - \iint \frac{m_t}{GJ} dx \end{aligned} \tag{C.14}$$

To obtain values for constants in terms of the nodal variables, Eq. (C.14) is solved with $x = 0$, and it is assumed that there is no loading at $x = 0$, that is, $p_t(0) = 0$. Implementing these constraints gives the constants of integration in terms of the nodal variables at node 1:

$$\begin{aligned} C_1 &= M_{t1} \\ C_2 &= \phi_1 \end{aligned} \tag{C.15}$$

Substituting the constants of integration Eq. (C.15) into Eq. (C.14) gives

$$\begin{aligned} M_{t2} &= M_{t1} - \int m_t \; dx \\ \phi_2 &= \phi_1 + M_{t1}\frac{L}{GJ} - \iint \frac{m_t}{GJ} \; dx \end{aligned} \tag{C.16}$$

Transfer Matrix Relation:

Combining (4), (8), (12), and (16) as:

$$\begin{aligned} \mathbf{d}_2 &= \mathbf{T}_{dd}\mathbf{d}_1 + \mathbf{T}_{df}\mathbf{f}_1 \\ \mathbf{f}_2 &= \mathbf{T}_{ff}\mathbf{f}_1 \end{aligned} \tag{C.17}$$

where $\mathbf{d}_1$ and $\mathbf{d}_2$ are the displacement vectors of nodes 1 and 2, and $\mathbf{f}_1$ and $\mathbf{f}_2$ are the force vectors of nodes 1 and 2. Expressing Eq. (C.17) in matrix form leads to Eq. (C.18),

$$\begin{bmatrix} \mathbf{d}_2 \\ \cdots \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{dd} & \vdots & \mathbf{T}_{df} \\ \cdots & \cdot & \cdots \\ \mathbf{0} & \vdots & \mathbf{T}_{ff} \end{bmatrix} \begin{bmatrix} \mathbf{d}_1 \\ \cdots \\ \mathbf{f}_1 \end{bmatrix} \tag{C.18}$$

Expanding Eq. (C.18) gives,

$$\underbrace{\begin{bmatrix} u_2 \\ v_2 \\ w_2 \\ \phi_2 \\ \theta_{y2} \\ \theta_{z2} \\ \cdots \\ N_2 \\ V_{y2} \\ V_{z2} \\ M_{t2} \\ M_{y2} \\ M_{z2} \end{bmatrix}}_{\mathbf{q}_2} = \underbrace{\left[\begin{array}{cccccc:cccccc} 1 & 0 & 0 & 0 & 0 & 0 & \frac{L}{EA} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & L & 0 & \frac{-L^3}{6EI_z} & 0 & 0 & 0 & \frac{L^2}{2EI_z} \\ 0 & 0 & 1 & 0 & -L & 0 & 0 & 0 & \frac{-L^3}{6EI_y} & 0 & \frac{-L^2}{2EI_y} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \frac{L}{GJ} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{L^2}{2EI_y} & 0 & \frac{L}{EI_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{-L^2}{2EI_z} & 0 & 0 & 0 & \frac{L}{EI_z} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & L & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -L & 0 & 0 & 0 & 1 \end{array}\right]}_{\mathbf{T}^i} \underbrace{\begin{bmatrix} u_1 \\ v_1 \\ w_1 \\ \phi_1 \\ \theta_{y1} \\ \theta_{z1} \\ \cdots \\ N_1 \\ V_{y1} \\ V_{z1} \\ M_{t1} \\ M_{y1} \\ M_{z1} \end{bmatrix}}_{\mathbf{q}_1} \tag{C.19}$$

The matrix $\mathbf{T}^i$ is referred to as the *transfer matrix* since it "transfers" the variables from node 1 to node 2. The vectors $\mathbf{q}_1$ and $\mathbf{q}_2$ of displacements and forces are referred to as the *state vectors* because the variables fully describe the response/state of the beam [35].

The stiffness matrix for a beam element relates all of the displacements to all of the forces. The stiffness matrix relation is given by,

$$\mathbf{p}^i = \mathbf{k}^i \mathbf{d}^i \tag{C.20}$$

where

$$\begin{aligned} \mathbf{d}^i &= \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & w_1 & \phi_1 & \theta_{y1} & \theta_{z1} & u_2 & v_2 & w_2 & \phi_2 & \theta_{y2} & \theta_{z2} \end{bmatrix}^T \\ \mathbf{p}^i &= \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} N_1 & V_{y1} & V_{z1} & M_{t1} & M_{y1} & M_{z1} & N_2 & V_{y2} & V_{z2} & M_{t2} & M_{y2} & M_{z2} \end{bmatrix}^T \end{aligned} \tag{C.21}$$

The stiffness matrix is written in a different sign convention than the transfer matrix. The transfer matrix formed above is based on a positive bending moment sign convention. For

a stiffness relation, a positive sign convention which aligns with the positive global axes is needed. Rewriting Eq. (C.17) as

$$\begin{aligned} \mathbf{d}_2 &= \mathbf{T}_{dd}\mathbf{d}_1 + \mathbf{T}_{dp}\mathbf{p}_1 \\ \mathbf{p}_2 &= \mathbf{T}_{pp}\mathbf{p}_1 \end{aligned} \tag{C.22}$$

where $\mathbf{T}_{pp} = -\mathbf{T}_{ss}$ and $\mathbf{T}_{dp} = -\mathbf{T}_{dp}$. Solving Eq. (C.22) in terms of $\mathbf{p}_1$ and $\mathbf{p}_2$ gives,

$$\begin{aligned} \mathbf{p_1} &= \mathbf{T}_{dp}^{-1}\mathbf{d}_2 - \mathbf{T}_{dp}^{-1}\mathbf{T}_{dd}\mathbf{d}_1 \\ \mathbf{p_2} &= \mathbf{T}_{pp}\mathbf{p}_1 = \mathbf{T}_{pp}\mathbf{T}_{dp}^{-1}\mathbf{d}_2 - \mathbf{T}_{pp}\mathbf{T}_{dp}^{-1}\mathbf{T}_{dd}\mathbf{d}_1 \end{aligned} \tag{C.23}$$

Rewriting Eq. (C.23) in matrix form, results in the stiffness matrix relation given in Eq. (C.20) (ignoring loads applied along the beam),

$$\underbrace{\begin{bmatrix} \mathbf{p}_1 \\ \cdots \\ \mathbf{p}_2 \end{bmatrix}}_{\mathbf{d}^i} = \underbrace{\begin{bmatrix} -\mathbf{T}_{dp}^{-1}\mathbf{T}_{dd} & \vdots & \mathbf{T}_{dp}^{-1} \\ \cdots & . & \cdots \\ -\mathbf{T}_{pp}\mathbf{T}_{dp}^{-1}\mathbf{T}_{dd} & \vdots & \mathbf{T}_{pp}\mathbf{T}_{dp}^{-1} \end{bmatrix}}_{\mathbf{k}^i} \underbrace{\begin{bmatrix} \mathbf{d}_1 \\ \cdots \\ \mathbf{d}_2 \end{bmatrix}}_{\mathbf{d}^i} \tag{C.24}$$

Expanding the stiffness $\mathbf{k}^i$ in Eq. (C.24) gives the 3-D stiffness matrix for a E-B beam,

$$\mathbf{k}^i = \left[\begin{array}{ccc:ccc:ccc:ccc} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{-EA}{L} & 0 & 0 & 0 & 0 & 0 \\ & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & \frac{-12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\ & & \frac{12EI_y}{L^3} & 0 & \frac{-6EI_y}{L^2} & 0 & 0 & 0 & \frac{-12EI_y}{L^3} & 0 & \frac{-6EI_y}{L^2} & 0 \\ \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots \\ & & & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{-GJ}{L} & 0 & 0 \\ & & & & \frac{4EI_y}{L} & 0 & 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\ & & & & & \frac{4EI_z}{L} & 0 & \frac{-6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \\ \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots \\ & & & & & & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{-6EI_z}{L^2} \\ & & & & & & & & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\ \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots & \hdots \\ & & & & & & & & & \frac{GJ}{L} & 0 & 0 \\ & Sym. & & & & & & & & & \frac{4EI_y}{L} & 0 \\ & & & & & & & & & & & \frac{4EI_z}{L} \end{array}\right] \tag{C.25}$$

D. Proposal

Design and Finite Element Analysis of a Triangular Shaped Building Structure

A Major Qualifying Project proposal submitted to the faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science

Submitted by:

Jon Palczynski

Khant Win Htet

Submitted to:

Professor Leonard Albano

October 11, 2018

Abstract

The primary purpose of this project is to design the structure of a five-story triangular shaped mixed-use office and residential building in San Francisco, CA. The geometry of the building will be triangular shaped and during the design process, certain scopes of work will be intensely focused. Since the building will be approximately 80 feet tall and located in earthquake zones of San Francisco, the effects of lateral loads will be considered in the design process. Application of Finite Element Analysis software such as ANSYS and PATRAN is one of the goals of the project. Moreover, design evaluation and cost analysis will be undergone as the final scope of the project. Alterations of different structural elements (bracing systems, shear wall systems and column positions/spacings) will be tested out to find a cost-efficient and structurally sound building. The project also aims as a case study for buildings with irregular configurations.

Table of Contents

List of Figures

List of Tables

1. Introduction

Modern geometric architecture is becoming more popular; however, it presents new structural design challenges. New structural framing methods, materials and computer tools must be investigated and used to address the complex geometric structures and associated loading scenarios. Triangular shaped buildings are a type of modern geometric structure but due to structural design challenges they are not very common. Nonetheless, a few examples exist; the Potsdamer Platz 11 in Berlin, Germany and the Flatiron Building in New York, New York. Both buildings are considered iconic landmarks of the area and draw tourist attention due to their unique shapes. However, these buildings are not in areas subject to dangerous seismic and wind loads.

To set a standard for the design of a modern triangular shaped building subject to seismic and wind loads, this project will explore the design and evaluation of a theoretical five-story triangular shaped mixed-use residential and office building located in San Francisco, CA, primarily focused on the steel framing system. The project will investigate different structural framing methods and computer software to aid in the design of a complex building structure subject to seismic and wind loads. Additionally, a cost estimate will be performed for each framing method to better evaluate and assess the viability of each option. The major tasks of this project will be two structural frame designs, CAD Models, Finite Element Models, Finite Element Analysis and cost estimates. The final outcome of the project will be the completed evaluation and design of the triangular shaped structure along with guidelines to follow when designing a uniquely shaped structure subject to seismic and wind loads.

2. Background

This section provides the information needed to understand the major components of the project. The major components of the project are the building location, structural analysis and the utilization of computer software tools.

2.1 Building Site Location

The building site for the triangular shaped mixed-use residential and office building is located at 1811 Jerrold Ave San Francisco, CA 94124. It is a 60000 Sq. Ft. triangular shaped lot currently for sale for $8 million. This site was chosen due to its already level surface and large size. The general site location was chosen due to it being in an area subject to high seismic and wind loads.

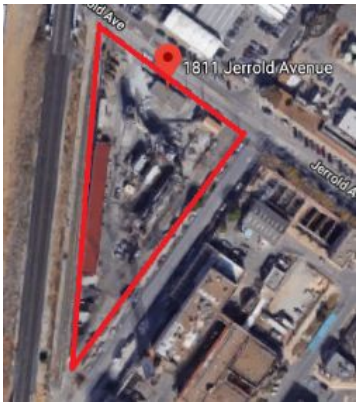

Figure 1: 1811 Jerrold Ave San Francisco, CA 94124 Building Site

2.2 Structural Analysis

Design loads will be calculated from the *American Society of Civil Engineering Specification 7-16* (ASCE 7-16). This building site is in a critical location because the building design must address the significant lateral loads. Lateral loads are live loads that act like a horizontal force acting on the side of a building, consisting of seismic loads and wind loads. It is important to follow ASCE 7-16 as it contains new specification regarding wind and seismic loading criteria. Many locations have transferred to or have deadlines set to transfer from ASCE 7-10 to ASCE 7-16. ASCE 7-16 contains safer, updated methods but also stricter methods for Site Class D structures and tall structures. The seismic design loads for tall Site Class D structures will have an additional increase of up to 50%. ASCE 7-16 also removed the provision that allowed for an 18% reduction in seismic design loads for tall structures. Updated ground motion maps and soil coefficients have also contributed to the increase in seismic design loads (Spaulding, 2018). For our structure seismic loading is the controlling lateral load.

Seismic loads are caused by earthquakes. The building site location is classified as a Seismic Design Category D zone by ASCE 7-16. A Seismic Design Category is given to a structure based on its usage and level of possible ground motion. Sites classified as Seismic Design Category D risk loss of soil strength. Structures classified as Seismic Design Category D could experience very strong shaking capable of producing a Modified Mercalli Intensity Scale (MMI) of VIII. The Modified Mercalli Intensity Scale has values assigned to a location after an earthquake has occurred; they are a measure of the viewable physical effects that the location experienced. An MMI of VIII is classified as damage slight in specially designed structures and considerable damage and partial collapse in ordinary buildings (U.S. Geological Survey, 1989).

Wind Loads are caused by air pressure acting on a building's surface. Unlike a rectangular building, a triangular building will not experience symmetrical wind distributions. The maximum and minimum pressures occur on a triangular building when the wind blows parallel to one of the faces. Due to the triangular shape the building will experience much higher twisting moments about its base when compared to a rectangular building (Abdusemed & Ahuja, 2015).

Dead loads are the frame's weight and any object that is permanently fixed to the building. Live loads are loads which can move and are not permanently fixed to the building. Live load design values are generally based on building occupancy classification and can be found in ASCE 7-16.

The structural design of a building can be divided into two parts; the superstructure and foundation. The superstructure design consists of the column layout and framing system. The column layout must account for usability and functionality of the spatial layout. The spatial layout is the floor plan and the arrangement of furniture, cubicles, counters, equipment and other items located within the floor plan (Nha & Leblanc, 2002). If columns are placed only to create the best structure with no consideration for the spatial layout, it may yield a completely unusable building. The superstructure design also consists of sizing the beams, columns and framing connections. The foundation design consists of sizing footings, the foundation wall dimensions and other components of the foundation. Foundation design follows the *American Concrete Institute Manual of Concrete Practice* (ACI Concrete Manual).

Beam and column sizes depend largely on the applied load combinations but also the frame layout. Modern structures typically contain large open spaces which increase the unbraced and effective lengths of beams and columns respectively; adversely affecting their strength. Connection variations include welded design, bolted design and coped (bolt) vs. not coped (weld) connections. Connections can be subject to pure shear, pure axial or a combination of both. Design of beams, columns and connections all follow *American Institute of Steel Construction: Steel Construction Manual* (AISC Steel Manual).

2.3 Finite Element Analysis

Finite Element Analysis will be used to analyze overall structure response and framing connection responses to seismic and wind loads present in San Francisco, CA. Finite Element Analysis works by dividing a complex structure into simple shaped elements connected by nodes. It can perform static analysis, modal analysis, transient dynamic analysis, buckling analysis and more. By simplifying a structure into simple shaped elements, a computer is able to solve the structure through large sets of simultaneous equations. Properties are calculated at the nodes and then interpolated over the element. Commonly used elements are 1D Beam Elements,

2D Plate Elements and 3D Solid Elements; a mix of elements can be used in a Finite Element Model. After the Finite Element Model is solved it can display items such as displacement, stress, strain, mode shapes and temperature. A common way to represent results is with a contour plot (Weck & Yong, 2004).

Modal Response Spectrum Analysis must be used to analyze the building structure due to the seismic loads. Modal Response Spectrum Analysis is a linear dynamic analysis method which measures the contribution from each natural mode of vibration to determine the maximum seismic response of an elastic structure. Because of the unique shape of the building and its location, a linear static analysis cannot be used. Once the analysis is complete, shear forces are computed to be distributed along the height of the building. (Emrah, 2016). The building will be designed from a seismic loading point of view and then will be checked for wind loading. Similar to the seismic design, because the structure is a unique shape, a normal wind load procedure cannot be used. A Computational Fluid Dynamics model must be made to analyze the wind loads acting on the triangular structure. Computational Fluid Dynamics generates fluid flow through a model of the structure and uses a numerical analysis method, such as the Finite Element Method, to find a solution (Lohner, 2009).

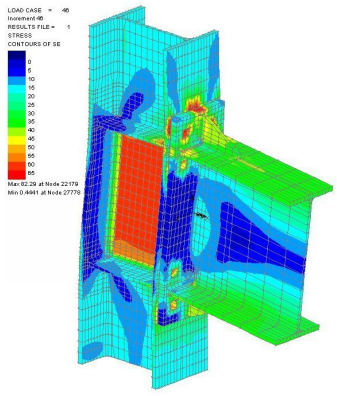

Figure 2: Finite Element Model with Stress Contour Plot (Lindsey, 2017)

Finite Element Analysis can be divided into three main parts; pre-processing, solving and post-processing. Pre-processing is the development of a Finite Element Model. This is where a structure is meshed into elements, material properties are assigned, loads and boundary conditions are applied, etc. After a Finite Element Model is created it needs to be solved. A text

file is created with all components of the pre-processing phase; node locations, material properties, loads, etc. This text file is then sent to the solver. Solving is when the set of equations associated with the structure are put into matrix form and the variables are then solved for. The matrices can be extremely large and require a lot of computing power to be solved within a reasonable amount of time. After the solving is complete the user can view and analyze the results, known as post-processing. Post-processing is where results of stress, strain, displacement, etc. can be viewed as contour plots, graphs, etc. For example, solvers can calculate various types of stresses throughout a structure including; Mises, Tresca, Principal, Axial and Shear (Roensch, 2008).

There are a lot of commercially available Finite Element Analysis software packages; these consist of pre-processors, solvers, post-processors or "all in one" packages. A few examples include ABAQUS, HyperMesh, MSC NASTRAN, MSC PATRAN, ANSYS and STAAD. For this project the software will be limited to an educational version that does not have substantial model size limitations. When choosing a pre and post-processor it is important to consider the Finite Element Model creation capabilities, ability to import CAD geometry, solver support, user interface, results presentation, automation and scripting capabilities and overall value (Wertel, 2018).

From our initial investigation we have found that the ANSYS Workbench software package can create the overall structure and provide member forces through Modal Response Spectrum Analysis. ANSYS Workbench contains Design Modeler, a built-in CAD tool to create a framing system that can then be meshed with the built-in pre-processor. ANSYS Workbench also includes a built-in solver and post-processor. For connection design we have found it is generally easier to use CAD software for creating the model geometry rather than creating the geometry within a pre-processor. CATIA is a CAD tool that can create framing connections and then have the geometry directly inputted to MSC PATRAN for pre-processing. ANSYS, MSC NASTRAN, MSC PATRAN and CATIA V5 all have free educational versions available to students which make them the most viable options.

3. Methodology

This section outlines the major tasks of the project, how they will be completed and who will complete them. Each task includes the associated resources to assist in completion. The flowchart below lists the major steps.

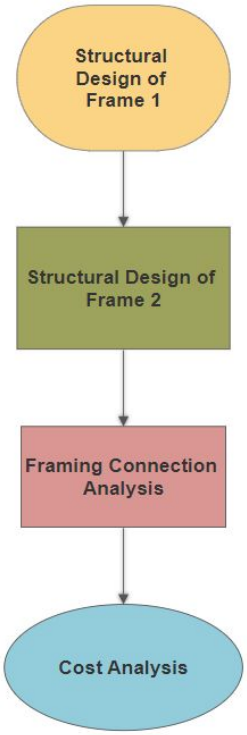

Figure 3: Methodology Flowchart

3.1 Structural Design

The structural design includes the superstructure and foundation design. Two frame designs (each of a different type) will be created and both will follow the same overall design procedure.

Table 1: Building Structure Design Tasks

Task	Resources	Team Member
Choose Framing System	ASCE 7-16	Jon P. & Khant H.
Set Column Grid	ASCE 7-16	Khant H.
Determine Design Loads	ASCE 7-16	Khant H.
Modal Response Spectrum Analysis	ASCE 7-16	Jon P.
Compute Member Sizes	AISC Steel Manual	Khant H.
Design Framing Connections	AISC Steel Manual	Jon P.
Design Foundation Elements	ACI Concrete Manual	Khant H.
Check Wind Loads	ASCE 7-16	Khant H.

3.1.1 Choose Framing System

Seismic resisting frame systems presented in ASCE 7-16 will be investigated and two will be chosen to develop into a full structure. Framing system types include steel eccentrically braced frames, steel special moment frames, steel special truss moment frames, etc. The following steps apply to both frame choices.

3.1.2 Set Column Grid

A column layout will be set to maximize the spatial layout but also provide for a safe structure. The column layout will be designed in AutoCAD.

3.1.3 Determine Design Loads

Design loads will be determined from ASCE 7-16 based on building occupancy type and other classifications. Design loads include gravity and lateral loads. Because the building is located in an earthquake zone, a separate seismic analysis will need to be performed for lateral loading. The tributary areas of each beam will need to be calculated to determine the distributed load to be applied along the beam from gravity load combinations.

3.1.3.1 Load Combinations

Governing load combination is calculated according to section 2.3.1 and 2.3.6 from ASCE 7-16. ASCE also states that wind and seismic loads need not to be considered to act simultaneously.

3.1.3.2 Gravity Load

Gravity loads that is applied for the building include dead load, live load, snow load and rain load. Uniformly distributed and concentrated live load is selected according to Table 4.3-1 from ASCE 7-16. Whether the live load reduction is permitted or not has to be further calculated as refers in Section 4.7.

To calculate the ground snow load, figure 7.2-1 from ASCE 7-16 is used as a reference. Ground snow load is then converted into flat roof snow loads from section 7.3-1 using different factor values such as exposure factor,thermal factor,importance factor.

Chapter 8 of ASCE 7-16 discuss design of rain loads. It says rain loads is based on the static head and hydraulic head values which are associated with the primary and secondary drainage system.

3.1.4 Modal Response Spectrum Analysis

Modal Response Spectrum Analysis will follow the general guidelines set forth by ASCE 7-16. First, a 3D Finite Element Model will be created in accordance with ASCE 7-16 using ANSYS. After the 3D Finite Element Model is created, Modal Response Spectrum Analysis will be performed. An adequate number of mode shapes will be found to have a total of 90% modal mass participation in each orthogonal direction. The base shear forces to be distributed along the height of the building will be combined with the design loads to compute the member forces utilizing the original 3D Finite Element Model.

3.1.5 Compute Member Sizes

After the member forces in the columns, beams and bracing members are found, the steel members will be sized. Excel sheets will be made to automate the procedure. All steel members will be designed in accordance with the AISC Steel Manual.

3.1.6 Design Framing Connections

Once the members are sized, the framing connections will be designed. Excel sheets will be made to automate the procedure. The connections will be designed in accordance with the AISC Steel Manual.

3.1.7 Design Foundation Elements

The pile footings and connections to the superstructure will be designed once the superstructure is complete. The concrete elements will be designed in accordance with the ACI Concrete Manual.

3.1.8 Check Wind Loads

After the building is designed for the seismic loads, it will be checked for the wind loads. A Computational Fluid Dynamics model will need to be created. This will be completed in ANSYS. Wind will be applied at various angles to see the effects on the triangular structure. The wind loads will be determined and the building will be checked to see if it is still compliant.

3.2 Framing Connection Analysis

Finite Element Analysis will be used to see how different types of framing connections respond to seismic loads. The connections will also be checked for fatigue and other conditions.

Table 2: Framing Connection Analysis Tasks

Task	Resource	Team Member
3D CAD Model Creation	CATIA V5 Workbook	Jon P.
Finite Element Analysis of Connections	Patran User's Guide	Jon P.

3.2.1 3D Model Creation

The framing connection geometry for different types of connections including the angles and bolts will be modeled in CATIA V5.

3.2.2 Finite Element Analysis of Connections

The CATIA V5 models will be directly imported to MSC PATRAN for meshing. MSC NASTRAN will be used as a solver. The results will be viewed in the MSC PATRAN post-processor. A comparison of the effects of seismic loading on different types of framing connections will be laid out.

3.3 Cost Estimate

Table 3: Cost Estimate Tasks

Task	Resource	Team Member
Cost Estimate	RS Means Construction Data	Jon P.

3.3.1 Cost Estimate

A cost estimate will be performed for each framing system using RS Means Construction Data. Additionally, full prices (non-student editions) of all software used will be found to give a perspective on how much it would cost for a design firm to utilize them all.

4. Capstone Design Statement

To comply with the Accreditation Board for Engineering and Technology (ABET), the Department of Civil and Environmental Engineering at WPI requires all Major Qualifying Projects (MQP) to include a Capstone Design Experience. The Capstone Design Experience requires students to design a system, component or process to meet desired needs by applying knowledge and skills acquired in earlier coursework and incorporating engineering standards and realistic design constraints. To fulfill this requirement, this MQP is the design and evaluation of a five-story triangular shaped mixed-use residential and office building located in San Francisco, CA. Alternate building designs will be investigated and proposed.

During the design process of the building, various effects of lateral loads due to wind and earthquakes will be investigated. Since the proposed property is situated in a location geologically vulnerable to earthquakes, seismic-resisting structures will be intensely studied and utilized. The design will follow standard engineering building codes; AISC Steel Manual, ACI Concrete Manual and ASCE 7-16. To evaluate the constructability and efficiency of the building, Finite Element Analysis will be undergone with different design scenarios. The main study of the project will explore how slight alteration of different design components, (beam/column sizes, floor-to-ceiling height, column positioning) will influence and improve the overall design of the structure. The final focus of the project will be reporting a cost analysis of the structural skeleton of the building. The MQP will incorporate economic, constructability, ethical, health and safety, and social design constraints.

4.1 Economic

A cost estimate of each structure will be completed and broken down into individual components like beams, columns and connections. Different beam and columns sizes, column arrangements and flooring systems will be considered to determine the most cost-effective design. The recommendation for the final structure will consider cost.

4.2 Constructability

The aspect of constructability will be highly prioritized during development of design scenarios. Efficient constructability includes using beam/column sizes and column spacings.

4.3 Ethical

During the design process, the code of ethics for engineers provided by National Society of Professional Engineers (NSPE) will be followed. The building is in an area subject to seismic loads and will be required to adhere to seismic design provisions in ASCE 7-16. While this may require additional design work and increased material costs, compromising the overall quality of

the structure for cost-efficiency will be strictly avoided. Moreover, creating a design with structural integrity will be the top priority of this capstone project.

4.4 Health and Safety

Health and safety during the construction as well as post construction is one of the fundamental code of ethics by NSPE. All structural designs will be followed by standard building codes such as ASCE 7-16. Because the building will support office and residential spaces and is in a location subject to seismic loads the appropriate risk category will be assigned. Assigning the appropriate risk category and following the design procedures associated with it will ensure the safety of occupants.

4.5 Social

Although this project is mainly aimed to design to structural skeleton of the building, the proposed site is planned to be office/residential building. The building will comprise of small and medium sized office spaces, small shop spaces and residential apartments for different types of social background. Column grids must account for the spatial layout to ensure adequate and functional offices and apartments can be implemented into the building.

5. Professional Licensure Statement

To design a product, either electronic device or living space, one of the utmost important factors is safety and health of the public. The engineers are required to train rigorously to a certain level where they are considered as competent to design or review a product. As civil/structural engineers, designing or constructing a building comes with a significant amount of risk and require years of academic knowledge and work experience.

To create a standard limit of becoming a credential engineer, the National Society of Professional Engineers (NSPE) specifies, in order to archive a PE (professional Engineer), one must: (1) finish a 4-year engineering program from accredited university/college, (2) pass the FE (Fundamental of Engineering) exam. (3) complete 4 years of work experience under a PE and (4) pass the PE exam. Once an engineer becomes a certified PE, it comes with a lot of authority as well as responsibility. Since PE engineer can design, review and approve a project, following the ethical guidelines would be the main responsibility.

This capstone design project is almost strictly related structural professional practice. It includes structural members sizing, structural analysis and detail drawings. The outcome of the project will be a product which need a professional engineers' approval if the project ever come to reality.

6. Deliverables

The following deliverables will be included in addition to the final MQP report.

Table 4: Deliverables

ANSYS Building Model
CATIA Connection Models
CAD Drawings
Excel Calculation Sheets
Hand Calculations

7. Conclusion

As a capstone design project, the final aim is to not only utilize the theoretical knowledge that is gained from 4-years of civil engineering modules but also connect the different civil engineering aspects and consider the most efficient outcome with structural integrity. At the end of the project, it is also expected that the team will gain knowledge of different design processes, structural analysis applications, simulation software and design optimization processes. Moreover, the project is expected to extend the capacity of the team as structural engineers and also improve the problem-solving skills. Properly utilizing personal experience and academic knowledge as well as the guidance of the advisor will be one of the primary intent of the project.

8. Project Schedule

The schedule below outlines the process the project will follow. The two frame designs will be completed one after another to better structure the project outcome. After the two frame designs are complete, Finite Element Analysis will be used to investigate different connection combinations for each structure to see which performs the best under seismic loading. When preparing the final report a final recommendation will be made as to which is the best framing design and what are the best connection designs for an irregular shaped building subject to seismic loads.

Table 5: Project Schedule

Task	Month							
	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Site Evaluation								
Frame Design 1								
Column Design 1								
Beam Design 1								
Connection Design 1								
Foundation Design 1								
Frame Design 2								
Beam Design 2								
Column Design 2								
Connection Design 2								
Foundation Design 2								
FEA of Connections								
Cost Analysis								
Proposal								
Final Report								
Project Presentation								

References

Abdusemed, Muftha Ahmed, and Ashok Kumar Ahuja. “Wind Pressure Distribution on Triangular Shape Tall Buildings.” *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 4, no. 8, Aug. 2015.

ACI Manual of Concrete Practice 2017. American Concrete Institute, 2017.

ASCE/SEI 7-16 Minimum Design Loads and Associated Criteria for Buildings and Other Structures. American Society of Civil Engineers, 2017.

Eduran, Emrah. “Response Spectrum Analysis: Theory, Benefits and Limitations.” njtf.no/wp-content/uploads/2016/10/Technical-review-of-Modal-Analysis.pdf.

Lindsey, Jon. *Stresses in Exterior Connection with Full-Depth Column Web Stiffeners*. 2017, www.lusas.com/case/civil/connection_research.html.

Lohner, Rainald. *Applied CFD Techniques: An Introduction Based on Finite Element Methods*. 2nd ed., Wiley, 2009.

Mahendran, Mahen. “International Conference on Computer Aided Engineering.” 11 Mar. 2008.

Nguyen, Nha, and Gaston Leblanc. “Contact Personnel, Physical Environment and the Perceived Corporate Image of Intangible Services by New Clients.” *International Journal of Service Industry Management*, vol. 13, no. 3, 2002, pp. 242–262.

Roensch, Steve. “Finite Element Analysis: Post-Processing.” *Finite Element*, 2008, www.finiteelement.com/feawhite4.html.

Spaulding, Erin. “IBC 2018/ASCE 7-16: DCI ENGINEERS PREPARES FOR BIG CHANGES.” *DCI Engineers*, 18 May 2018, www.dci-engineers.com/news/ibc-2018asce-7-16-dci-engineers-prepares-big-changes.

Steel Construction Manual Fifteenth Edition. American Institute of Steel Construction, 2017.

The Severity of an Earthquake. U.S. Geological Survey, pubs.usgs.gov/gip/earthq4/severitygip.html.

Weck, Olivier de, and Il Yong Kim. “Finite Element Method.” Engineering Design and Rapid Prototyping. web.mit.edu/16.810/www/16.810_L4_CAE.pdf.

Wertel, Scott. *FEA Buyer's Guide for Pre- and Postprocessing Software*. 27 Sept. 2013, www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/6391/FEA-Buyers-Guide-for-Pre-and-Postprocessing-Software.aspx.

E. Cost Estimate Data

Cost data gathered from [15].

Building Category	Description	Unit	Unit Cost	Cost per S.F
Substructure				
Standard Foundations	[002/004]12" thick, 4000 psi concrete 9.5 feet wide footing	each	40.95	7.32
Slab on Grade	[003/1020] 5" thick, reinforced, non-industrial	S.F floor	5.5	5.5
Foundation Excavation	[001/008] Excavate, Back fill, 8 feet deep 20' x20'	each	960	6.25
B10 Superstructure				
Floor Constructions	[208/6400] 500kips load, 20ft unsupported	V.L.F	194.1	88.10
	[254/0720] 20'x20' bay size 5"slab thickness-126psf-W shape. Composite Deck	S.F floor	22.25	22.25
	[720/3450] Column Fireproof Gypsum Board	V.L.F	26.11	1.77
Roof Constructions	[112/1500] 15'x20' bay size, 40psf steel joists-beams-deck on columns	S.F floor	5.68	5.68
B20 Exterior Enclosure				
Exterior Wall	[103/5950]6" thickness, 20x10 panel size, precast concrete with rigid insulation	S.F wall	46	31.41
Exterior Windows	[106/6450]Aluminum, Double Hung, Insulated 4'5" x 5'3"	each	620	1.16
Exterior Doors	[110/6300]Glazed Door with Aluminum and Glass 3' x 7'	each	2975	1.86
B30 Roofing				
Roof Coverings	[120/1000]Single Ply Membrane	S.F floor	3.56	0.712
Roof Deck Rigid Insulations	[320/1650]Roof Deck Rigid Insulations 2.5" thick	S.F floor	1.93	0.386
Roof Edges	[420/1000] Aluminium Roof Edges 0.050" thick	L.F floor	25.3	0.173
Gutters	[610/0050] Gutter Box, Aluminium 0.027" thick	L.F floor	8.97	0.061
Interiors				
Partitions	[126/7850]Dry Wall Paritions/ Metal Stud Framing 5/8" FR wall	S.F wall	7.38	5.039
Interior Doors	[102/2500]Single Leaf wood 3' x 7' x 1 3/8"	each	626	1.96
Stairs	[110/0700] 24 risers, with landing	flight	16,625	0.665
Wall Finishes	[230/0080] Painting Interior /primer & 2 coats	S.F wall	1.2	3.82
Floor Finishes	[410/0660] Tile& coverings 3/4" thick	S.F Floor	7.78	7.78
Ceiling Finishes	[105/4500] Plaster Ceiling 3.4# metal	S.F Floor	12.54	12.54
Services				
D10 Conveying				
Elevators and Lift	[140/1300] Passenger 2000lbs, 5 floors	each	171,100	6.844
D20 Plumbing				
Plumbing Fixtures	[932/1360] Bathtub, water closet, stall shower and lavatories	each	9775	4.89
Rain Water Drainage	Roof drains	S.F floor	1.68	1.68
D30 HVAC				
Energy Supply	Oil fired hot water, basedboard radiations	S.F floor	7.87	7.87
Heating Generating Systems	Air heating system	S.F floor	9.54	9.54
D40 Fire Protection				
Sprinklers	Wet pipes spinkler system, light hazard	S.F floor	2.78	2.78
Standpipes	Standpipes and hose systems, with pumps	S.F floor	0.95	0.95
D50 Electrical				
Electrical Service/Distribution	1600 Ampere service, panel board and feeders	S.F floor	2.45	2.45
Lighitng & Branch Wiring	Incandecent Fixtures, receptacles, swithced, A.C and misc. power	S.F floor	7.59	7.59
Communications & Sercurity	Addressable Alarm Systems, Emergency lighting, Internet and Phone Wiring	S.F floor	1.7	1.7