OPTIMIZATION AND MODELING TOOLS
FOR TELESCOPE HEXAPOD STRUCTURES

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Mechanical Engineering

by Michael Edward Feeney

June 3, 2011
Abstract

OPTIMIZATION AND MODELING TOOLS FOR TELESCOPE HEXAPOD STRUCTURES
Michael Edward Feeney

Hexapod trusses are an important element in many mechanical design systems. The natural frequency and stiffness behavior under geometric and mass variations of such structures is largely undocumented. Furthermore, the ability to quickly model hexapod designs and explore a large design-space in finite element software packages is, in general, time consuming and inefficient. The purpose of this project was to develop software tools that made design-space exploration (modeling and simulation processes) for hexapod structures drastically more efficient. Secondly, the project included an experimental analysis portion to demonstrate the various modal study techniques and to validate finite element analysis predictions. Lastly, the project investigated a specific hexapod design problem as a means of exhibiting the modeling/optimization software tools and to develop an understanding of the natural frequency behavior of hexapods. To this end, the research could be used for the design of telescope secondary support structures and other hexapod optimization engineering problems.
Acknowledgements

I would like to acknowledge the following individuals:

First, I would like to thank my family for the perpetual support throughout my entire academic career. They have motivated me to work hard and stay focused under the most stressful and difficult times.

Secondly, I would like to thank Stephen Padin for introducing me to CCAT at Caltech and for giving me the opportunity to work on such an exciting project. It has truly been an honor working under Steve. His guidance and knowledge of telescope design have been an inspiration and a great asset towards my professional career. Steve is not a professor at Cal Poly, but he was willing to be one of my advisors (a superb one at that) and for that reason he has gone above and beyond all expectations. This thesis project would not have existed without Steve and his continual participation and advisement throughout the year has been invaluable. I have thoroughly enjoyed and appreciated all of the work I have done with Steve and his involvement with the project is something for which I am sincerely grateful.

Thirdly, I would like to thank Peter Schuster for his support and guidance as a professor and an advisor on this project. The finite element analysis courses I had with Dr. Schuster have provided me with critical skills for a large portion of my thesis and will undoubtedly serve me well throughout my career. It would be difficult to imagine having a more suitable and experienced advisor for this project.

Finally, I would like to thank Jim Meagher for his guidance and participation in the project. The experimental validation portion of the project involved the use of the vibrations laboratory and equipment at Cal Poly. Previously, I was unfamiliar with the equipment and techniques for experimental vibration analysis, and Dr. Meagher was instrumental in guiding and teaching me throughout this section of the project.
Contents

List of Figures x

List of Tables xvi

1 Introduction 1

1.1 Overview and Motivation .................................................. 2
1.2 Background Research ...................................................... 3
1.3 Hexapods ................................................................. 4
1.3.1 Definition ............................................................ 4
1.3.2 Degrees of Freedom and Fully Constrained vs. Over Constrained .... 4
1.3.3 Large Telescope Applications ...................................... 5
1.4 Plan of Project ........................................................... 7

2 Finite Element Analysis 8

2.1 Element Types ............................................................ 10
2.2 Stiffness Matrix .......................................................... 11
2.3 Bar Element Formulation ............................................... 15
2.4 Beam Element Formulation .......................................... 17
2.5 Beam Elements in 3D Space .......................................... 21
2.6 Discussion ............................................................... 24

3 Natural Frequency Analysis ............................................... 26
3.1 Governing Equation ......................................................... 26
3.2 Lumped Mass Matrix ....................................................... 27
3.3 Lanczos Eigensolver ......................................................... 28
3.4 Discussion ................................................................. 28

4 Software Development ..................................................... 29
4.1 Finite Element Analysis with Software .................................. 30
4.2 ABAQUS Software .......................................................... 30
  4.2.1 ABAQUS Graphical User Interface .................................. 31
  4.2.2 ABAQUS Code Environment .......................................... 32
4.3 MATLAB Integration with ABAQUS ..................................... 33
  4.3.1 Node Generation Function ........................................... 35
  4.3.2 Main Node Generation ................................................. 36
  4.3.3 Main Element Generation .............................................. 37
  4.3.4 Triangular Reinforcement Node and Element Generation .......... 38
  4.3.5 Cross Reinforcement Node and Element Generation ............... 38
  4.3.6 Vertical Node and Element Generation .............................. 39
  4.3.7 Input File Generation .................................................. 40
  4.3.8 Main Executable File ................................................... 42

5 Experimental Validation .................................................. 44
5.1 Physical Model: Construction ........................................... 44
5.2 Physical Model: Software Predictions .................................. 48
  5.2.1 ABAQUS Simulations ................................................ 49
  5.2.2 SolidWorks Simulation ............................................... 50
5.3 Hardware ................................................................. 51
5.4 Modal Analysis: Impact Hammer ....................................... 53
5.5 Modal Analysis: Shake Table ........................................... 58
A.7 Triangular Reinforcement Element Generation ........................................ 108
A.8 Cross Reinforcement Node Generation .................................................. 110
A.9 Cross Reinforcement Element Generation ............................................. 111
A.10 Vertical Node Generation ................................................................. 112
A.11 Vertical Element Generation .............................................................. 113
A.12 Mass Search Tool .................................................................................. 114
A.13 Frequency Search Tool ........................................................................ 115
A.14 Diameter and Thickness Calculator ...................................................... 116
List of Figures

1.1 ABAQUS Hexapod base model. ................................................................. 2
1.2 ABAQUS Hexapod reinforced model. ....................................................... 2
1.3 Three-Dimensional space degrees of freedom (Cartesian Coordinates) ........ 4
1.4 Cassegrain mirror configuration [7]. ...................................................... 6
1.5 ALGOR prototype model of CCAT telescope (Image courtesy of David Woody [6]). This figure shows the primary support, secondary support, and incoming ray behavior. ................. 7

2.1 Two-node Bar and Beam Element [9]. ....................................................... 10
2.2 Multi-node Two-Dimensional Planar Elements [9]. ................................ 11
2.3 Three-Dimensional Solid Elements [9]. .................................................. 11
2.4 Two-node Spring Element [9]. ............................................................... 11
2.5 Two-node Spring Elements in series [9]. .................................................. 14
2.6 Two-node Bar Element arbitrarily oriented in two-dimensional space [9]. ........ 15
2.7 Timoshenko Beam differential section [9]. .............................................. 18
2.8 Differential Beam Element with distributed loading, moments, and shear forces [9]. .... 19
2.9 Three-Dimensional Beam Element with all degrees of freedom shown [9]. ......... 22

4.1 ABAQUS CAE graphical user interface. .................................................. 31
4.2 ABAQUS command line description. ...................................................... 33
4.3 Geometry variables for the hexapod-tripod structure. .............................. 34
4.4 Flowchart showing how the individual functions are used. ......................... 35
4.5 Example execution of the node_gen.m matlab file. Results are plotted using the scatter3 command in MATLAB.

4.6 Scatter3 plot from MATLAB isolating the main structural nodes.

4.7 Base model hexapod-tripod structure rendered in ABAQUS with node (red) and element (black) labels.

4.8 Triangular reinforcement nodes isolated from the rest of the structural nodes. Plotted using scatter3 in MATLAB.

4.9 Cross reinforcement nodes isolated from the rest of the structural nodes. Plotted using scatter3 in MATLAB.

4.10 Vertical nodes isolated from the rest of the structural nodes. Plotted using scatter3 in MATLAB.

4.11 All structural nodes superimposed together using MATLAB scatter3 function. Main nodes, triangular reinforcement nodes, cross reinforcement nodes, and vertical nodes colored green, blue, red, and yellow respectively.

4.12 Reinforced hexapod-tripod model from ABAQUS GUI showing the three unique structural member definitions.

4.13 Example input text file showing the different parameter definitions.

4.14 Example batch file, created from MATLAB, to be sent to ABAQUS command module.

5.1 ABAQUS Hexapod base model.

5.2 Ling Electronics shake table surface bolt pattern schematic.

5.3 K&S Stock #1149 brass tubing.

5.4 Enhanced view of the unbeveled physical model leg endings.

5.5 Image of the soldered joints on the physical model (left) and the ABAQUS rendered node intersection (right).

5.6 Image of the L-Brackets brazed to the ends of the physical model legs. Note that the hexapod feet were brazed to the plates where as the other joints were soldered because of large heat dissipation issues.

5.7 Completed physical model bolted to the Ling Electronics shake table.

5.8 Different locations for the accelerometer. This image shows where the point mass was positioned in ABAQUS and where the accelerometer was positioned on the physical model.

5.9 ABAQUS results from the various simulations of the physical model. (a)-(d) correspond to accelerometer point mass locations (see Figure 5.8). The only significant change in natural frequency occurred when placing the point mass midspan of the leg member (most likely due to the fact that this was the excitation location for the first mode).
5.10 **ABAQUS** mode shape result for the simulations.  
5.11 **SolidWorks** three-dimensional sketch and corresponding weldment model.  
5.12 **SolidWorks** first mode natural frequency result and corresponding mode shape. The first mode natural frequency was 194.74 Hz.  
5.13 A view of the various hardware tools used in the experimental analyses.  
5.14 Modal hammer analysis strike locations for the upper long span members.  
5.15 Modal hammer analysis strike locations for the upper central triangle members.  
5.16 Modal hammer analysis frequency response plot (Transfer Function $H_{1.2}$ vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the upper long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.  
5.17 Modal hammer analysis frequency response plot (Transfer Function $H_{1.2}$ vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the upper central triangle members. The accelerometer was oriented parallel with the shake table. The amplitude response was very low and the first mode natural frequency was difficult to extrapolate from this plot.  
5.18 Modal hammer analysis frequency response plot (Transfer Function $H_{1.2}$ vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the upper long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.  
5.19 Modal hammer analysis frequency response plot (Transfer Function $H_{1.2}$ vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the lower long span members. The accelerometer was oriented perpendicular with the shake table. The first amplitude peak represents the first mode natural frequency.  
5.20 Modal hammer analysis frequency response plot (Transfer Function $H_{1.2}$ vs. frequency). The accelerometer was positioned at the midspan (see Figure 5.8 (d)) and the strike location was on one of the upper long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.  
5.21 Modal hammer analysis frequency response plot (Transfer Function $H_{1.2}$ vs. frequency). The accelerometer was positioned at the midspan (see Figure 5.8 (d)) and the strike location was on one of the lower long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.  
5.22 Ling Electronics shake table setup. Shake table and drive system (upper left). Controller system (upper right). Accelerometer locations (bottom).  
5.23 Shake table analysis frequency response plot (Transfer Function $H_{1.2}$ vs. frequency). The structure accelerometer was positioned on the upper node intersection point shown in Figure 5.8 (b) and the shake table was sent a white noise signal.
5.24 Shake table analysis frequency response plot (Transfer Function $H_{1,2}$ vs. frequency). The structure accelerometer was positioned on the lower node intersection point shown in Figure 5.8 (c) and the shake table was sent a white noise signal. ...................................................... 61

5.25 Shake table analysis frequency response plot (Transfer Function $H_{1,2}$ vs. frequency). The structure accelerometer was positioned in the midspan location shown in Figure 5.8 (d) and the shake table was sent a white noise signal. ...................................................... 61

5.26 Cumulative plot of both testing methods (Hammer and Shake Table) along with the simulation results from ABAQUS (5% geometry variation). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)). ...................................................... 62

5.27 Cumulative plot of both testing methods (Hammer and Shake Table) along with the simulation results from ABAQUS (5% geometry variation). The accelerometer was positioned at the lower node intersection (see Figure 5.8 (c)). ...................................................... 63

5.28 Cumulative plot of both testing methods (Hammer and Shake Table) along with the simulation results from ABAQUS (5% geometry variation). The accelerometer was positioned at the midspan location (see Figure 5.8 (d)). ...................................................... 63

6.1 Geometry variables for the hexapod-tripod structure. ...................................................... 67

6.2 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. overall radius of hexapod ($r$). ...................................................... 68

6.3 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. center triangle length ($u_1$). ...................................................... 69

6.4 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. lower hexapod height ($u_2$). ...................................................... 69

6.5 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. lower/upper hexapod spacing ($u_3$). ...................................................... 70

6.6 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. member 1 thickness. ...................................................... 70

6.7 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. member 1 diameter. ...................................................... 71

6.8 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. material density. ...................................................... 71

6.9 Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. Modulus of Elasticity. ...................................................... 72

6.10 Reinforced Model natural frequency vs. member 2 thickness. ...................................................... 72

6.11 Reinforced Model natural frequency vs. member 2 diameter. ...................................................... 73

6.12 Reinforced Model natural frequency vs. member 3 thickness. ...................................................... 73

6.13 Reinforced Model natural frequency vs. member 3 diameter. ...................................................... 74
7.1 Base Model first mode natural frequency vs. $m_{1t}$. The various series show the natural frequency with different $u_3$ values. The blockage was limited to 1% and the mass varied from 100 kg to 300 kg (larger $m_{1t}$ yielded larger mass. $m_{1d}$ was kept fixed to a diameter that corresponded to 1% blockage. 

7.2 Base Model first mode natural frequency vs. $m_{1t}$. The various series show the natural frequency with different $u_3$ values. The blockage was limited to 2% and the mass varied from 200 kg to 300 kg (larger $m_{1t}$ yielded larger mass. $m_{1d}$ was kept fixed to a diameter that corresponded to 2% blockage. 

7.3 Base Model first mode natural frequency vs. $m_{1t}$. This plot, combined with Figures 7.1 and 7.2, shows that there is a maximum natural frequency vs. thickness. Figures 7.1 and 7.2 don’t show this limit because only practical thicknesses were used.

7.4 Base Model mode shapes for three different $u_3$ values, (a) $u_3 = 0.5$, (b) $u_3 = 1$, (c) $u_3 = 2$.

7.5 This plot shows the frequency response for all of the simulations that were run with 1% blockage. Each point within a series corresponds to a different combination of member diameter and thickness (see 7.21) and each series represents a specific $u_3$ value.

7.6 This plot shows the frequency response for all of the simulations that were run with 2% blockage. Each point within a series corresponds to a different combination of member diameter and thickness (see 7.21) and each series represents a specific $u_3$ value.

7.7 This plot shows two cases: 1% and 2% blockage with a fixed $u_3 = 1$ m. This is the same data from Figures 7.5 and 7.6, however, the $u_3 = 1m$ cases are isolated from the rest of the results. There are discontinuities in the 1% blockage case due to the fact that some of the diameter and thickness combinations caused errors in ABAQUS. This error was only when the thickness was larger than the diameter of the member.

7.8 This plot shows a portion of the simulation results and the corresponding natural frequency behavior for various member thickness and diameters (2% Blockage). Each series corresponds to a unique $u_3$ value as indicated in the legend. The arrows indicated the member diameter and thickness changes. $m_{2t}$ and $m_{3t}$ were held at specific values (25% of $m_{1t}$ for this case).

7.9 $m_{2t}/m_{1t} = 0.25$ and $m_{3t}/m_{1t} = 0.45$. See Figure 7.8 for a detailed description of the plot.

7.10 $m_{2t}/m_{1t} = 0.25$ and $m_{3t}/m_{1t} = 0.65$. See Figure 7.8 for a detailed description of the plot.

7.11 $m_{2t}/m_{1t} = 0.45$ and $m_{3t}/m_{1t} = 0.25$. See Figure 7.8 for a detailed description of the plot.

7.12 $m_{2t}/m_{1t} = 0.45$ and $m_{3t}/m_{1t} = 0.45$. See Figure 7.8 for a detailed description of the plot.

7.13 $m_{2t}/m_{1t} = 0.65$ and $m_{3t}/m_{1t} = 0.25$. See Figure 7.8 for a detailed description of the plot.

7.14 $m_{2t}/m_{1t} = 0.65$ and $m_{3t}/m_{1t} = 0.45$. See Figure 7.8 for a detailed description of the plot.

7.15 $m_{2t}/m_{1t} = 0.65$ and $m_{3t}/m_{1t} = 0.65$. See Figure 7.8 for a detailed description of the plot.

7.16 $m_{2t}/m_{1t} = 0.65$ and $m_{3t}/m_{1t} = 0.65$. See Figure 7.8 for a detailed description of the plot.

7.17 MATLAB three-dimensional scatter plot with a colorbar representing frequency. $u_3 = 1$ m and $m_{2t}/m_{1t} = 0.25$. 

xv
7.18 MATLAB three-dimensional scatter plot with a colorbar representing frequency. $u_3 = 1$ m and $m_{2t}/m_{1t} = 0.45$. ................................................................. 89

7.19 MATLAB three-dimensional scatter plot with a colorbar representing frequency. $u_3 = 1$ m and $m_{2t}/m_{1t} = 0.65$. ................................................................. 90

7.20 MATLAB three-dimensional scatter plot with a colorbar representing frequency. $u_3 = 1$ m and $m_{2t}/m_{1t} = 0.95$. ................................................................. 90

7.21 Member thickness and diameter variations displayed in a truncated excel worksheet. .... 91

7.22 Reinforced model mode shapes. Higher natural frequencies tend to have a more distributed mode shape (right), whereas lower natural frequencies tend to have a localized mode shape on the long span member (left). ......................................................... 92
## List of Tables

5.1 Design constraints for Hexapod Optimization. .............................................. 49

6.1 Base and Reinforced Hexapod Versions and corresponding variables. .............. 66

6.2 Default parameters for the general study of the base, scaled base, and reinforced hexapod structures. ................................................................. 68

7.1 Design constraints for Hexapod Optimization. .............................................. 77

7.2 Base and Reinforced Hexapod Versions and corresponding variables for optimization. 78
List of Tables
List of Figures
Chapter 1

Introduction

In mechanical design problems, it is common to build computer prototypes of structures as a means of testing the practicality and structural performance of a design. Computer models are significantly easier to construct and manipulate than physical structures (especially for large and complicated geometries). Depending on the mechanical application, it can be essential that a structure is designed optimally for a particular task. Building and modifying physical structures for optimization purposes are more expensive and time consuming than simply modifying the parameters on a computer model. The computer software packages that are commonly used in mechanical design problems are Computer Aided Design (CAD) and Finite Element Analysis (FEA) programs. Over many years of development, these types of programs have become a very useful tool for a variety of engineering problems. Users are able to build a realistic representation of a physical model and run a variety of analyses to test it’s strength, heat properties, vibration response, etc. On the other hand, there are some downsides when using CAD and FEA software to optimize a structure. In some optimization scenarios, multiple variations of a specific design are investigated as a means of finding an optimal configuration. Unfortunately, using standalone FEA and CAD programs to accomplish this task can be inefficient. This thesis discusses a particular structural design problem, the difficulties associated with optimization using CAD and FEA programs, and a proposed solution.
1.1 Overview and Motivation

The structural behavior of hexapods is dependent on geometric, material, and mass variables. The motivation behind this project was to build an understanding of how these variables affect the natural frequency of hexapod structures by developing modeling and optimization software. The hexapod designs under investigation are seen in Figure 1.1. In addition, models like the one seen in Figure 1.2, with reinforcements added, were also an area of interest. These types of structures are being investigated for radio telescope secondary supports. With the constant improvement of radio telescope camera systems (finer sensitivity and higher observation frequencies), low blockage, light weight, and rigid secondary supports are becoming increasingly important. In the past, blockage was less of a concern due to the higher noise of the receiving cameras and, as a result, the secondary supports were designed for those conditions.

![ABAQUS Hexapod base model](image1.png)

Figure 1.1: ABAQUS Hexapod base model.

![ABAQUS Hexapod reinforced model](image2.png)

Figure 1.2: ABAQUS Hexapod reinforced model.
1.2 Background Research

Before delving into the core components of the thesis, previous studies on optimization techniques and secondary support design theory will be discussed. It was essential to know what type of structural natural frequency optimization techniques had already been studied, and the types of secondary support systems used for radio telescopes. As for natural frequency analysis optimization, the most closely related research pertained to optimization of tripod and quadrupod structures. Furthermore, there was very little information found pertaining to natural frequency analysis and optimization of hexapod structures.

In one study by DOWEC (Dutch Offshore Wind Energy Converter Project), the natural frequency and stiffness of tripod designs were investigated. The tripods in the study were designed to support a wind turbine mounted at the top of the structure and the base was partially submersed in the ocean. The natural frequency of the tripod structure was designed “such that resonance due to wave excitation and wind excitation at the rotor and blade-passing frequencies [were] avoided” [4]. The DOWEC design study was a helpful supplement to the thesis as it was an example optimization problem of a relatively similar structure to a hexapod. Additionally, the project helped illustrate how to present multi-variable optimization data with the use of surface plots.

A study by Lamb and Olver presented the mathematical formulation for planar and spherical wave blockage of secondary support structures for telescopes [8]. The article was closely related to the work in this thesis as it discussed the different blockage effects from secondary support structures for telescopes. The formulations for spherical and planar wave blockage in the article were for quadrupod structures, however, the information was still very useful. Further work for this thesis project could investigate structures that mount within the primary mirror (as opposed to completely outside), in which case spherical blockage would have to be incorporated in the calculations.

Claydon and Dang’s article “The Choice of Subreflector Support Geometry for Earth Station Antennas” [1] discussed the various interference or scattering properties of subreflector (secondary) support structures. Despite the fact that the study was based on quadrupod structures, it analyzed the scattering effects from different cross-sectional member profiles. The information in the article would be useful for an extension to this thesis as alternative cross-section profiles could be investigated.

Last of all, various sources were used to gain an understanding of the finite element method and vibration analysis theory. The ABAQUS Online Documentation provided useful information about beam theory and natural frequency analysis as it pertained to finite element analysis software [10]. This infor-
information helped determine the proper techniques for analyzing natural frequency for beam type structures. Clough’s Dynamics of Structures was a useful source for understanding the basic theory of vibration analysis with regards to beams [2]. Lastly, Logan’s A First Course in the Finite Element Method and Cook’s Concepts and Applications of Finite Element Analysis were useful sources for the basic formulation of the finite element method.

1.3 Hexapods

1.3.1 Definition

By definition, hexapod is a generic term used to describe a six legged support system. In most cases, hexapods are used to support an object mounted at the joining ends of the legs as a means of providing rigidity and stability. There are numerous variations of hexapod designs, however, for this project, the discussion will be limited to the structures seen in Figures 1.1 and 1.2. These particular designs consist of two layered hexapods attached with vertical members at the center triangular portion of the structures. The leg members from the hexapods span from the center triangular structure to three evenly spaced locations.

1.3.2 Degrees of Freedom and Fully Constrained vs. Over Constrained

The hexapod design under investigation (Figure 1.1) is considered fully-constrained and not over-constrained (in the translational degrees of freedom). In order to grasp the concept of constraints, it is first essential to define degrees of freedom.

Figure 1.3: Three-Dimensional space degrees of freedom (Cartesian Coordinates)
A degree of freedom is the ability of an object to translate or rotate in a given physical space. As seen in Figure 1.3, three-dimensional space has six degrees of freedom, namely, translation and rotation about the $x$, $y$, and $z$ axes. Constraining an entity, in mechanical terms, is to eliminate a particular degree of freedom. A hexapod, consisting of rods with ball-joint ends, fully constrains, but does not over constrain a rigid body. A rigid body has six degrees of freedom (see Figure 1.3) and the six hexapod rods are just sufficient to constrain these translation and rotation displacements.

Structures with less than six constraints cannot fully support a rigid body. For example, a monopod only constrains loading along its length. A tripod with rods fixed with ball joint ends does not fully support a rigid body, but it can fully support a point mass (because a point does not have rotational degrees of freedom, only translational). On the contrary, a tripod with fixed end supports can fully support a rigid body, however, the cross-section has to be sufficiently large to resist bending. Quadrupods, are inefficient structures because they over constrain the secondary mirror with ball joint or fixed supports.

### 1.3.3 Large Telescope Applications

For simple applications such as small optical telescopes, surveying equipment, etc. tripod supports have shown to be an effective method for simple stabilization. Tripods are a typical choice for the support mechanism because the instrument can be treated as a point mass and therefore there is little need for bending stiffness. Because of the fact that these structures are relatively small (on the order of a couple meters), the design process is relatively trivial. Stabilization becomes more difficult when the size of the system increases because larger structures have higher gravitational and thermal deformations. For smaller structures, such as small optical telescopes, these effects are relatively negligible due to the large stiffness to mass ratio.

In large terrestrial telescopes, such as the Caltech Sub-millimeter Observatory, James Clerk Maxwell Telescope, and the Heinrich Hertz Sub-millimeter Telescope, complex secondary support structures are incorporated into the designs. These 10 meter diameter telescopes are Cassegrain designs (see Figure 1.4, in which the secondary mirror is positioned at a fixed from the primary mirror.)
In order to achieve good image quality, the telescope structure must maintain alignment of the primary and secondary mirrors in the presence of gravitational and thermal deformations, wind buffeting, and acceleration due to scanning the telescope across the sky. Sub-millimeter wavelength observation times are typically hours long which makes it essential for the telescope mirrors to remain aligned. One of the key components of a Cassegrain telescope is the secondary support structure as seen in Figure 1.5. The secondary support holds the secondary mirror away from the primary mirror at a specific distance. These types of structures are typically more complicated than tripod supports because, being that the secondary mirror is a rigid body (not a point mass), bending must be resisted. When designing a secondary support structure, there are three primary concerns:

1. **Natural Frequency / Stiffness**: Determines how much the support will deflect when the telescope accelerates and wind blows on the secondary.

2. **Ray Blockage**: The amount of loss of the incoming light due to the structural obstruction.

3. **Weight**: The structure has to be light enough such that it does not significantly deform the primary support.

For most large telescopes in existence, quadrupod (four legged) secondary support structures are used, however, these configurations generally over-constrain the secondary mirror. The new 20-30 meter class of optical telescopes have hexapod supports. In the CCAT secondary support shown in Figure 1.5, the hexapod terminates at three points on short towers that attach to the primary support. Deformation of the primary moves the three points, but the hexapod does not impart large torques on the secondary.
To address the goal, a project plan was developed consisting of three major tasks:

1. Develop a MATLAB code that is capable of efficiently generating ABAQUS input files. The ABAQUS input files contain the information that describe hexapod models similar to Figure 1.1 and Figure 1.2 for Finite Element Analysis. The code will be referred to as MATLAB-interfaced-ABAQUS (MIA).

2. Construct a physical model of a simple hexapod design similar to Figure 1.1 and conduct experimental vibration analysis. The purpose behind the physical model is to validate software predictions with experimental testing.

3. Compile a database that defines the effects on natural frequency from basic geometric, material, and mass variations. Investigate a design scenario where many simulations are run from the MATLAB-interfaced-ABAQUS optimization software.

Before delving into the results and technical methods of the project, it is important that the reader understand the rudimentary theory behind Finite Element Analysis and Vibrations. Chapter 2 and Chapter 3 discuss the fundamentals of the Finite Element Method and Natural Frequency Analysis, respectively, as they pertain to this thesis.
Chapter 2

Finite Element Analysis

Finite Element Analysis or the Finite Element Method (FEA or FEM) is defined as, “a numerical method for solving problems of engineering and mathematical physics” by breaking the problem up into many individual elements [9]. The Finite Element Method is a powerful tool for a variety of physics problems including heat transfer, fluid dynamics, and electromagnetic potential, but for this project FEA will only be discussed in terms of structural analysis. This chapter is designed to give the reader an overview of FEA, define the stiffness matrix, derive the equations for the simplest of elements, and finally derive the equations for beams (the elements used in the project). It is important to note that the mathematical derivations and the majority of the figures have been borrowed from [9].

The core principals of the theory are as follows:

- Analytical solutions, mathematical expressions that yield desired quantities, are valid for every point in a continuum [9]. In terms of structures, a continuum is the material body of the system. Typical analytical solutions for structural analysis describe the system’s displacement and stresses under various loading conditions.

- Typically the solutions are high order partial differential equations and thus unobtainable for most structures. Simple structures generally have comprehensible mathematical solutions, but more intricate systems have impractical and incalculable expressions.

- To work around complicated partial differential equations, the finite element method breaks down the
system into smaller bodies or elements which are interconnected by nodes or points. This process is called discretization and represents the idea of generating algebraic equations for individual pieces of the system and solving them simultaneously. The method by which the differential equations are replaced with algebraic equations is with the use of the Finite-Difference Method. That is, by making the element size small enough, one can approximate a first order derivative via the following equation

\[
f'(a) = \lim_{h \to 0^+} \frac{f(a + h) - f(a)}{h} \approx \frac{f(a + h) - f(a)}{h}
\]

For structural analysis, typical solutions find displacements at the nodal coordinates and the stresses within the elements. The process begins by dividing the structure into individual pieces or elements and choosing the correct element type. There are a variety of element types that are appropriate for different systems. The next step is choosing a displacement function which is the mathematical formula describing the motion of material within each element. These functions are often linear, quadratic, cubic, and higher order polynomials. After the finding the displacement functions, the relationships between the strain/displacement and stress/strain have to be derived. Following those relationships, the element stiffness matrix and force equilibrium equations are derived. The stiffness matrix relates the material displacement with the loads applied to the system. Once the stiffness matrices are found for each element, the global stiffness matrix is compiled by linking all of the elements together at the nodes. This process is done by superposition and continuity between the nodes. The boundary conditions, which are constraints on individual nodes or surfaces, are applied and the unknown degrees of freedom (generalized displacements) are determined. With the calculation of the global displacements, strains and stresses can be determined. Finally, the results can be interpreted and validated.

For simple systems that have few elements and nodes, finite element analysis can be done by hand calculations. When analyzing larger systems with many nodes, computers are used to do the tedious and numerous calculations. Common programs for linear, static structural finite element analysis include ABAQUS, ALGOR, SolidWorks Simulation, and ANSYS.

In summary, the generalized Finite Element Method works in the process as follows:

1. Select Element Type
2. Select a Displacement Function
3. Determine Strain/Displacement and Stress/Strain Relationships
4. Derive the Element Stiffness Matrix and Elemental Equations
5. Assemble the Elements to Obtain the Global Stiffness Matrix and Global Equations
6. Apply Boundary Conditions to Solve for Unknown Degrees of Freedom
7. Solve for Desired Stresses and Strains
8. Interpret the Results

### 2.1 Element Types

As discussed previously, there are a variety of element types used in the Finite Element Method. Different element types are appropriate for different structures. Determining what type of element to use in a structural analysis is a very critical part of FEA. A general rule of FEA is to simplify the problem as much as possible while maintaining the accuracy of the desired data. Below is a list of relevant element types.

- **Bars**: Useful for deflections in truss structures. Bars can only resist loading in the axial direction (2D and 3D Space). See Figure 2.1
- **Beams**: Useful for deflections due to bending (2D and 3D Space). Beams can take axial and transverse loading. See Figure 2.1
- **2D-Elements**: Useful for Plain Strain and Plane Stress problems. See Figure 2.2
- **3D-Elements**: Useful for complex geometries that require detailed information throughout the solid body. See Figure 2.3

![Figure 2.1: Two-node Bar and Beam Element [9].](image_url)
2.2 Stiffness Matrix

In classical mechanics of materials, Hooke’s Law is,

\[ f = kd \] \hspace{1cm} (2.2)

where \( f \) is the force on an element, \( d \) is the local coordinates \((u,v,w)\) displacements, and \( k \) is the stiffness. The simplest form of the stiffness matrix is that of a spring element. A spring element can only resist force in the axial direction. For a single spring with two nodes, the following is the standard procedure of the Finite Element Method:

1. Select Element Type: Spring

2. Select a Displacement Function: A spring has only two degrees of freedom (displacements and or rotations), \( d_{1x} \) and \( d_{2x} \) along the axial direction. The subscript 1 or 2 refers to the node number while
the subscript $x$ refers to the $x$ direction. For a linear spring, the displacement function is,

$$u(x) = a_1 + a_2x$$ (2.3)

where $a_1$ and $a_2$ are the coefficients for the two nodes with one degree of freedom each. In general, the number of coefficients corresponds to the number of degrees of freedom in the element. In matrix form, the displacement function is,

$$[u] = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$ (2.4)

To express $u$ as a function of the nodal displacements $d_{1x}$ and $d_{2x}$, the geometric conditions must be applied. This process is as follows,

$$u(x = 0) = d_{1x} = a_1$$ (2.5)

$$u(x = L) = d_{2x} = a_2L + d_{1x}$$ (2.6)

$$a_2 = \frac{d_{2x} - d_{1x}}{L}$$ (2.7)

Now that the constants $a_1$ and $a_2$ are expressed in terms of the degrees of freedom $d_{1x}$ and $d_{2x}$ and $L$ (node separation length), the displacement function can be written as,

$$u = d_{1x} + \frac{d_{2x} - d_{1x}}{L}x$$ (2.8)

In matrix form, the equation is,

$$[u] = \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} \begin{bmatrix} d_{1x} \\ d_{2x} \end{bmatrix}$$ (2.9)

where $\begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix}$ is called the shape function matrix $[N]$. The shape function matrix relates the nodal displacements $(d_{1x}, d_{2x})$ to displacement throughout the element.

3. Determine Strain/Displacement and Stress/Strain Relationships: The difference of the nodal displacements gives the overall deformation of the spring. The expression that relates the strain to the displacement is,

$$\delta = u(x = L) - u(x = 0) = d_{2x} - d_{1x}$$ (2.10)
4. Derive the Element Stiffness Matrix and Elemental Equations: From Figure 2.4,

\[ f_{1x} = -T; f_{2x} = T \] (2.11)

where \( T \) is the force due to extension or contraction of the spring. Substituting 2.10 and 2.11 into 2.2 gives

\[ T = -f_{1x} = k (d_{2x} - d_{1x}) \] (2.12)

and

\[ T = f_{2x} = k (d_{2x} - d_{1x}) \] (2.13)

hence

\[
\begin{bmatrix}
  f_{1x} \\
  f_{2x}
\end{bmatrix} =
\begin{bmatrix}
  k_{1,1} & -k_{1,2} \\
  -k_{2,1} & k_{2,2}
\end{bmatrix}
\begin{bmatrix}
  d_{1x} \\
  d_{2x}
\end{bmatrix}
\] (2.14)

The \( k \) matrix in this expression is called the element stiffness matrix. In general, this matrix is symmetric and square.

5. Assemble the Elements to Obtain the Global Stiffness Matrix and Global Equations: With only one spring, the global stiffness matrix is the same as the element matrix; however, if there were multiple springs linked together, the assembly process is expressed by,

\[
K = [k] = \sum_{e=1}^{n} k^e
\] (2.15)

\[
F = \{f\} = \sum_{e=1}^{n} f^e
\] (2.16)

These expressions explain that the global stiffness matrix and global force vector are the sums of all element stiffnesses (denoted by \( k^e \)) and element forces (denoted by \( f^e \)) respectively.

6. Apply Boundary Conditions to Solve for Unknown Degrees of Freedom: In general, the stiffness matrix is singular (no inverse exists) until boundary conditions are applied to constrain the system. The formulation above was for a single spring element without any boundary conditions.

To show the process of linking multiple elements together, consider a two element spring system shown in Figure 2.5. The same process for other element types is used, the only difference being additional degrees of
freedom. For element 1, the equations are given in matrix form as,

\[
\begin{bmatrix}
  f_{1x}^{(1)} \\
  f_{3x}^{(1)}
\end{bmatrix} =
\begin{bmatrix}
  k_1 & -k_1 \\
  -k_1 & k_1
\end{bmatrix}
\begin{bmatrix}
  d_{1x}^{(1)} \\
  d_{3x}^{(1)}
\end{bmatrix}.
\] (2.17)

For element 2, the equations are given in matrix form as,

\[
\begin{bmatrix}
  f_{3x}^{(2)} \\
  f_{2x}^{(2)}
\end{bmatrix} =
\begin{bmatrix}
  k_2 & -k_2 \\
  -k_2 & k_2
\end{bmatrix}
\begin{bmatrix}
  d_{3x}^{(2)} \\
  d_{2x}^{(2)}
\end{bmatrix}.
\] (2.18)

In these equations, the subscripts 1 and 2 on the \( k \) terms denote the different spring stiffnesses. For continuity or compatibility, it is clear that the displacements and forces at node 3 must follow,

\[
d_{3x}^{(1)} = d_{3x}^{(2)} = d_{3x}^{(2.19)}
\]

\[
f_{3x} = f_{3x}^{(1)} + f_{3x}^{(2)}
\] (2.20)

\[
f_{2x} = f_{2x}^{(2)}
\] (2.21)

\[
f_{1x} = f_{1x}^{(1)}
\] (2.22)

After assembly, the resulting equations in matrix form are,

\[
\begin{bmatrix}
  f_{1x} \\
  f_{2x} \\
  f_{3x}
\end{bmatrix} =
\begin{bmatrix}
  k_1 & 0 & -k_1 \\
  0 & k_2 & -k_2 \\
  -k_1 & -k_2 & k_1 + k_2
\end{bmatrix}
\begin{bmatrix}
  d_{1x} \\
  d_{2x} \\
  d_{3x}
\end{bmatrix}.
\] (2.23)

Notice that in this form, the matrix equations are unsolvable until the boundary condition \( d_{1x} = 0 \) is applied. Otherwise, there are too many unknowns for the given number of equations. Furthermore, from statics, \( f_{1x} = -f_{2x} - f_{3x} \). Finally to solve for the nodal displacements \( d_{2x} \) and \( d_{3x} \), the stiffness matrix must
be inverted and multiplied by the forces as seen in the following equation,

$$
\begin{bmatrix}
  d_{2x} \\
  d_{3x}
\end{bmatrix}
= 
\begin{bmatrix}
  k_2 & -k_2 \\
  -k_2 & k_1 + k_2
\end{bmatrix}^{-1}
\begin{bmatrix}
  f_{2x} \\
  f_{3x}
\end{bmatrix}
$$

(2.24)

2.3 Bar Element Formulation

Bar elements are very similar to spring elements in that both types can only resist loading in the axial
direction. The primary difference between the two is the introduction of stress and strain. Once again,
following the Finite Element procedure:

1. Select Element Type: Bar

![Figure 2.6: Two-node Bar Element arbitrarily oriented in two-dimensional space [9].](image)

2. Select a Displacement Function: Identical to the spring element Equation 2.3.

3. Determine Strain/Displacement and Stress Strain Relationships: For bar elements, the strain is only
   in the axial direction and is defined as the derivative of the displacement function. The strain/dis-
   placement relationship is,

$$
\epsilon_x = \frac{du(x)}{dx} = \frac{d_{2x} - d_{1x}}{L}
$$

(2.25)
where \( u(x) \) is the displacement function, \( d_{1x} \) and \( d_{2x} \) are the degrees of freedom, and \( L \) is the length of the element. The stress/strain expression is,

\[
\sigma_x = E\varepsilon_x
\]  

(2.26)

where \( E \) is the modulus of elasticity (a material property).

4. Derive the Element Stiffness Matrix and Elemental Equations: From strength of materials of linear elastic solids, the relationship of force and stress is,

\[
T = A\sigma_x
\]  

(2.27)

where \( T \) is the tension or compression force and \( A \) is the cross-sectional area. Substitution of the \( \sigma_x \) the equation becomes,

\[
T = AE \frac{d_{2x} - d_{1x}}{L}
\]  

(2.28)

For a single element Equation 2.28

\[
\begin{bmatrix}
  f_{1x} \\
  f_{2x}
\end{bmatrix} = \frac{AE}{L} \begin{bmatrix}
  1 & -1 \\
  -1 & 1
\end{bmatrix} \begin{bmatrix}
  d_{1x} \\
  d_{2x}
\end{bmatrix}
\]  

(2.29)

so the stiffness matrix is,

\[
k = \frac{AE}{L} \begin{bmatrix}
  1 & -1 \\
  -1 & 1
\end{bmatrix}
\]  

(2.30)

5. Assemble the Elements to Obtain the Global Stiffness Matrix and Global Equations: Similar to the methods shown in the spring example, the global stiffness matrix and equations can be assembled for multiple elements. If the elements are oriented in different angles, simple transformation matrices are used to convert all displacements and forces into a universal or global coordinate system. The stiffness derivations above were for a single one dimensional element. For a single bar element oriented in two
dimensions (see Figure 2.6, a second displacement $d_y$ is introduced and the stiffness matrix is,

$$[k] = \frac{AE}{L} \begin{bmatrix}
  d_{1x} & d_{1y} & d_{2x} & d_{2y} \\
  C^2(\theta) & C(\theta)S(\theta) & -C^2(\theta) & -C(\theta)S(\theta) \\
  C(\theta)S(\theta) & S^2(\theta) & -C(\theta)S(\theta) & -S^2(\theta) \\
  -C^2(\theta) & -C(\theta)S(\theta) & C^2(\theta) & C(\theta)S(\theta) \\
  -C(\theta)S(\theta) & -C(\theta)S(\theta) & C(\theta)S(\theta) & S^2(\theta)
\end{bmatrix} \quad (2.31)$$

where $C(\theta)$ and $S(\theta)$ are cosine and sine operators, respectively. To link multiple elements together, each individual elemental stiffness matrix has to be derived using the matrix equation above. The elemental matrices are then linked by matrix expansion and addition of the corresponding degrees of freedom.

### 2.4 Beam Element Formulation

A beam is defined as, “a long, slender structural member generally subjected to transverse loading that produces significant bending effects as opposed to twisting or axial effects” [9]. There are three degrees of freedom associated with a beam, namely, displacement in the x and y directions and rotation in the x-y plane. The two approaches to beam theory are the Euler-Bernoulli and the Timoshenko methods. The Timoshenko theory is, “higher order than the Euler-Bernoulli theory, [and] it is known to be superior in predicting the transient response of the beam” [5]. Furthermore, Timoshenko beam theory accounts for bending and transverse shear deformations. The primary difference between the two methods is that, unlike Euler-Bernoulli beams, Timoshenko beams do not restrict plane sections to remain plane in the deformed state. Following the Finite Element derivation:

1. Select Element Type: Beam

2. Select a Displacement Function: The displacement function requires some initial derivation. The shear and bending deformations are represented by $\beta(x)$ and $\phi(x)$ functions respectively (as seen in Figure
The slope of the deflected curve at point $x$ is given by

$$\frac{dv}{dx} = \phi(x) + \beta(x).$$

(2.32)

The relationship between the bending moment and the bending deformation is,

$$M(x) = EI \frac{d\phi(x)}{dx}.$$  

(2.33)

where $E$ is the modulus of elasticity and $I$ is the moment of inertia. The relationship between the shear force and the shear deformation is,

$$V(x) = k_s AG\beta(x).$$

(2.34)

where $k_s A$ is the shear area and $G$ is the shear modulus of Elasticity. The shear strain, $\gamma_{yz}$ is represented by the function $\beta(x)$ and may be written as,

$$\gamma_{yz} = \beta(x) = \frac{dv}{dx} - \phi,$$

(2.35)

from Eq. 2.32. By examining a differential beam element section under a uniform load (see Figure 2.8), the two governing differential equations are found to be,

$$\frac{d}{dx} \left[ k_s AG\left( \frac{dv}{dx} - \phi \right) \right] = -w.$$  

(2.36)
where \( w \) is a force per unit length.

\[
\frac{d}{dx}(EI\frac{d^2\phi}{dx^2}) + k_sAG\left(\frac{dv}{dx} - \phi\right) = 0 \tag{2.37}
\]

The first of these equations represents the summation of the shear forces. It is formulated by taking the derivative of the Eq. 2.34 with respect to \( x \) and substituting Eq. 2.35 for \( \beta(x) \). This expression is equal to the uniform loading \( w \). The second equation shows the summation of the moments using similar substitutions. For beams, the displacement formulation is assumed to be a cubic function given by

\[
v(x) = a_1x^3 + a_2x^2 + a_3x + a_4 \tag{2.38}
\]

where \( a_1, a_2, a_3, \) and \( a_4 \) are constants. The transverse shear \( \gamma_{yz} \) is assumed to be a constant given by,

\[
\gamma_{yz} = c \tag{2.39}
\]

To solve for the constant \( c \), the cubic displacement function Eq. 2.38, the slope function Eq. 2.32, the bending moment-curvature Eq. 2.33, and the shear force-strain Eq. 2.34 are substituted into

\[
V = \frac{dM}{dx} \tag{2.40}
\]

Then,

\[
c = 6a_1g \tag{2.41}
\]
where \( g = \frac{EI}{k AG} \) and \( \phi \) is,

\[
\phi(x) = a_3 + 2a_2x + (3x^2 + 6g)a_1
\] (2.42)

Solving for the constants \( a_1, a_2, a_3, \) and \( a_4 \) in terms of the nodal displacements \( d_1x, d_2x, d_1y, \phi_1, \) and \( \phi_2, \) the result is

\[
a_1 = \frac{2d_1y + L\phi_1 - 2d_2y + L\phi_2}{L(L^2 + 12g)}
\] (2.43)

\[
a_2 = \frac{-3Ld_1y - (2L^2 + 6g)\phi_1 + 3Ld_2y + (-L^2 + 6g)\phi_2}{L(L^2 + 12g)}
\] (2.44)

\[
a_3 = \frac{-12gd_1y + (L^3 + 6gL)\phi_1 + 12gd_2y - 6gL\phi_2}{L(L^2 + 12g)}
\] (2.45)

\[
a_4 = d_1y
\] (2.46)

Substituting these constants into the cubic function yields the final displacement function for the Timoshenko beam.

3. Determine Strain/Displacement and Stress/Strain Relationships: This step was incorporated into the derivation of the displacement function.

4. Derive the Element Stiffness Matrix and Elemental Equations: Solving the forces and moments at the ends of the beam element the following relationships are found,

\[
f_{1y} = V(0) = 6EIa_1 = \frac{EI(12d_1y + 6L\phi_1 - 12d_2y + 6L\phi_2)}{L(L^2 + 12g)}
\] (2.47)

\[
m_1 = -m(0) = -2EIa_2 = \frac{EI[6Ld_1y + (4L^2 + 12g)\phi_1 - 6Ld_2y + (2L^2 - 12g)\phi_2]}{L(L^2 + 12g)}
\] (2.48)

\[
f_{2y} = -V(L) = \frac{EI(-12d_1y - 6L\phi_1 + 12d_2y - 6L\phi_2)}{L(L^2 + 12g)}
\] (2.49)

\[
m_2 = m(L) = \frac{EI[6Ld_1y + (2L^2 - 12g)\phi_1 - 6Ld_2y + (4L^2 + 12g)\phi_2]}{L(L^2 + 12g)}
\] (2.50)
The stiffness matrix can be extracted from these equations in matrix form below,

\[
[k] = \frac{EI}{L(L^2 + 12g)} \begin{bmatrix}
    12 & 6L & -12 & 6L \\
    6L & (4L^2 + 12g) & -6L & (2L^2 - 12g) \\
    12 & -6L & 12 & -6L \\
    6L & (2L^2 - 12g) & -6L & (4L^2 + 12g)
\end{bmatrix}
\]

(2.51)

5. Assemble the Elements to Obtain the Global Stiffness Matrix and Global Equations: Same process as described in the bar element section. The angle of each element is adjusted via the rotation matrix.

2.5 Beam Elements in 3D Space

The previous equations described planar beam elements and planar loading. To conclude the discussion on the Finite Element Method, the following derivations define beam elements for arbitrary orientations in three dimensional space. There are six degrees of freedom associated with each node for a beam in three dimensional space:

1. Translation along the x-axis \( d_x \)
2. Translation along the y-axis \( d_y \)
3. Translation along the z-axis \( d_z \)
4. Rotation about the x-axis \( \phi_x \)
5. Rotation about the y-axis \( \phi_y \)
6. Rotation about the z-axis \( \phi_z \)

As seen in Figure 2.9, there are two bending planes \( x - z \) and \( x - y \) and a torsional plane \( y - z \). The other deformation is axial tension and compression along the x-axis. Each one of the planes has a corresponding stiffness matrix.
After superposition of all the bending and torsional planes, the final stiffness matrix for a two-node Timoshenko beam element with constant cross-section is,

\[
\begin{align*}
  k_{1,1} &= k_{7,7} = -k_{1,7} = -k_{7,1} = \frac{AE}{L} \\
  k_{2,2} &= -k_{2,8} = -k_{8,2} = 12kGAEI_y \frac{12EI_y + kGAL^2}{L(12EI_y - kGAL^2)^2} \\
  k_{2,6} &= k_{2,12} = k_{6,2} = k_{12,2} = 6kGAEI_y \frac{12EI_y + kGAL^2}{(12EI_y - kGAL^2)^2} \\
  k_{3,3} &= k_{3,9} = k_{9,3} = 12kGAEI_z \frac{12EI_z + kGAL^2}{L(12EI_z - kGAL^2)^2} \\
  k_{3,5} &= k_{3,11} = k_{5,3} = -6kGAEI_z \frac{12EI_z + kGAL^2}{(12EI_z - kGAL^2)^2} \\
  k_{4,4} &= -k_{4,10} = -k_{10,4} = \frac{G(I_y + I_z)}{L} \\
  k_{5,5} &= 4EI_z \left[\frac{(kGA)^2L^4 + 3kGAL^2EI_z + 36(EI_z)^2}{L(12EI_z - kGAL^2)^2}\right] \\
  k_{5,9} &= k_{9,5} = \frac{6kGAEI_z(12EI_z + kGAL^2)}{(12EI_z - kGAL^2)^2} \\
  k_{5,11} &= k_{11,5} = -\frac{2EI_z [72(EI_z)^2 - (kGA)^2L^4 - 30kGAL^2EI_z]}{L(12EI_z - kGAL^2)^2} \\
  k_{6,6} &= 4EI_y \left[\frac{(kGA)^2L^4 + 3kGAL^2EI_y + 36(EI_y)^2}{L(12EI_y - kGAL^2)^2}\right] \\
  k_{6,8} &= k_{8,6} = -\frac{6kGAEI_y(12EI_y + kGAL^2)}{(12EI_y - kGAL^2)^2}
\end{align*}
\]
\[
\begin{align*}
\text{k}_{6,12} &= k_{12,6} - \frac{2EI_y \left[ -(kGA)^2 L^4 - 30kGAL^2 EI_y + 72(EI_y)^2 \right]}{L(12EI_y - kGAL^2)^2} \\
\text{k}_{8,8} &= \frac{12kGAEI_y(12EI_y + kGAL^2)}{L(12EI_y - kGAL^2)^2} \\
\text{k}_{8,12} &= k_{12,8} = -\frac{6kGAEI_y(12EI_y + kGAL^2)}{(12EI_y - kGAL^2)^2} \\
\text{k}_{9,9} &= \frac{12kGAEI_z(12EI_z + kGAL^2)}{L(12EI_z - kGAL^2)^2} \\
\text{k}_{9,11} &= k_{11,9} = \frac{6kGAEI_z(12EI_z + kGAL^2)}{(12EI_z - kGAL^2)^2} \\
\text{k}_{11,11} &= \frac{4EI_z \left[ -(kGA)^2 L^4 + 3kGAL^2 EI_z + 36(EI_z)^2 \right]}{L(12EI_z - kGAL^2)^2} \\
\text{k}_{12,12} &= \frac{4EI_y \left[ -(kGA)^2 L^4 + 3kGAL^2 EI_y + 36(EI_y)^2 \right]}{L(12EI_y - kGAL^2)^2}
\end{align*}
\]

\[
[k] =
\begin{bmatrix}
k_{1,1} & 0 & 0 & 0 & 0 & 0 & k_{1,7} & 0 & 0 & 0 & 0 & 0 \\
0 & k_{2,2} & 0 & 0 & 0 & k_{2,6} & 0 & k_{2,8} & 0 & 0 & 0 & k_{2,12} \\
0 & 0 & k_{3,3} & 0 & k_{3,5} & 0 & 0 & 0 & k_{3,9} & 0 & k_{3,11} & 0 \\
0 & 0 & 0 & k_{4,4} & 0 & 0 & 0 & 0 & 0 & k_{4,10} & 0 & 0 \\
0 & 0 & k_{5,3} & 0 & k_{5,5} & 0 & 0 & 0 & k_{5,9} & 0 & k_{5,11} & 0 \\
0 & k_{6,2} & 0 & 0 & 0 & k_{6,6} & 0 & k_{6,8} & 0 & 0 & 0 & k_{6,12} \\
k_{7,1} & 0 & 0 & 0 & 0 & 0 & k_{7,7} & 0 & 0 & 0 & 0 & 0 \\
0 & k_{8,2} & 0 & 0 & 0 & k_{8,6} & 0 & k_{8,8} & 0 & 0 & 0 & k_{8,12} \\
0 & 0 & k_{9,3} & 0 & k_{9,5} & 0 & 0 & 0 & k_{9,9} & 0 & k_{9,11} & 0 \\
0 & 0 & 0 & k_{10,4} & 0 & 0 & 0 & 0 & 0 & k_{10,10} & 0 & 0 \\
0 & 0 & k_{11,3} & 0 & k_{11,5} & 0 & 0 & 0 & k_{11,9} & 0 & k_{11,11} & 0 \\
0 & k_{12,2} & 0 & 0 & 0 & k_{12,6} & 0 & k_{12,8} & 0 & 0 & 0 & k_{12,12}
\end{bmatrix}
\]  

(2.53)
The stiffness matrix above is for a single element in local coordinates. To change this matrix from a local to a global coordinate system, the following transformation process below is used.

\[
[k_{\text{global}}] = [T^T] [k_{\text{local}}] [T]
\]  

(2.54)

In this equation \([T]\) is the transformation matrix and is equal to

\[
[T] = \begin{bmatrix}
\lambda_{3x3} & & \\
& \lambda_{3x3} & \\
& & \lambda_{3x3}
\end{bmatrix}
\]

(2.55)

where \(\lambda\) is

\[
\lambda_{3x3} = \begin{bmatrix}
l & m & n \\
-m & l & 0 \\
-ln & -mn & D
\end{bmatrix}
\]

(2.56)

and

\[
l = \cos(\theta_{x\hat{x}})
\]

(2.57)

\[
m = \cos(\theta_{y\hat{x}})
\]

(2.58)

\[
n = \cos(\theta_{z\hat{x}})
\]

(2.59)

\[
D = (l^2 + m^2)^{1/2}
\]

(2.60)

where \(\theta_{x\hat{x}}, \theta_{y\hat{x}}, \text{and } \theta_{z\hat{x}}\) are the angles between the global \(x,y,z\) axes and the local \(x\) axis, \(\hat{x}\).

### 2.6 Discussion

Having an understanding of the basic formulation of the finite element method was an important part of this project. Without a fundamental knowledge of FEA theory, it is difficult to understand what takes place within FEA software. Furthermore, in order to model a structure properly using an FEA software package,
one has to know what types of elements to use and their basic mathematical formulations. Without this information, the user could easily create an invalid or inefficient model. This chapter was a discussion of the basic element types and the corresponding mathematics for static analyses. The next chapter of the thesis discusses vibration analysis theory, including the fundamental equations and the applications within FEA software. In addition, it explains how the static finite element analysis theory is implemented for dynamic analyses, specifically, natural frequency studies.
Chapter 3

Natural Frequency Analysis

With the finite element formulation of beams in place, the next step is understanding the process by which the fundamental (natural) frequencies are calculated. Natural frequencies are dependent on the system’s mass and stiffness. They are the frequencies at which forced vibrations generate the largest displacement amplitudes. There are numerous methods for calculating the natural frequencies of structures, but the discussion will be limited to the method used for two node beams in ABAQUS. Calculating the natural frequencies and mode shapes of a system is done by solving an eigenvalue problem [3]. ABAQUS uses the Lanczos Algorithm for the solving the eigenvalues and the lumping technique for generating the mass matrix.

3.1 Governing Equation

The eigenvalue problem for natural modes of a finite element system is given by

\[
\mu^2 [M] + \mu [C] + [K] \{\phi\} = 0
\] (3.1)

where \([M]\) is the mass matrix, \([C]\) is the damping matrix, \([K]\) is the stiffness matrix, \(\mu\) is the eigenvalue, and \(\{\phi\}\) is the eigenvector. When analyzing the natural frequencies, the damping is disregarded. By definition, the natural frequency is the frequency by which the system vibrates without the influence of a forcing function. Calculation of the natural frequency neglects damping because it is the unforced vibration.
Furthermore, \( \mu \) becomes an imaginary eigenvalue, \( \mu = i\omega \), where \( \omega \) is the natural frequency. Thus Equation 3.1 becomes,

\[
[ -\omega^2 [M] + [K] ] \{ \phi \} = 0
\]  

\( (3.2) \)

### 3.2 Lumped Mass Matrix

As discussed previously, the finite element method divides a system into individual pieces or elements. Each element contains a specific number of nodes (2 nodes per beam element for the cases under investigation). Just as the stiffness matrix defines the distributed stiffness throughout the structure, the same has to be done with the mass for a dynamic analysis. The lumped mass matrix technique “assumes that the entire mass is concentrated at the points at which the translational displacements are defined” [2]. In other words the mass is evenly divided and placed at the two nodes of the beam. As discussed in the previous chapter, for a beam in three-dimensional space, there are a total of six degrees of freedom, namely, the three translational displacements and the three corresponding rotations. The lumped mass matrix for a single three-dimensional beam element is,

\[
[m] = \begin{bmatrix}
1 & d_{1x} & d_{1y} & d_{1z} & \phi_{1x} & \phi_{1y} & \phi_{1z} & d_{2x} & d_{2y} & d_{2z} & \phi_{2x} & \phi_{2y} & \phi_{2z} \\
1 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]  

\( (3.3) \)
where \( m_1 = m_2 = \frac{m_{element}}{2} \). As seen in the mass matrix above, the off diagonal terms are zero due to the fact that “an acceleration of any mass point produces an inertial force at that point only” [2]. Furthermore, “the mass associated with any rotational degree of freedom will be zero because of the assumption that the mass is lumped in points which have no rotational inertia” [2]. For multiple elements, the mass matrix is simply expanded corresponding to the degrees of freedom associated with stiffness matrix. The lumped mass matrix approach yields slightly lower natural frequency results and uses far less memory than other methods. Assuming the elements are sufficiently small, the lumped mass matrix technique is a valid option for two-node beam elements.

### 3.3 Lanczos Eigensolver

The Lanczos Eigensolver is a method ABAQUS uses to solve the governing eigenvalue problem. In large systems with thousands of elements, solving the eigenvalue problem directly with matrix inversion is computationally expensive. The Lanczos method is an iterative numerical process that uses a shifting and stepping procedure to find the converged eigenvalues and eigenvectors. The details of the Lanczos Algorithm can be found in [10].

### 3.4 Discussion

The information discussed in this chapter provides some of the internal mathematics and techniques behind a natural frequency analysis using FEA software. Understanding the limitations and mathematics of the various natural frequency analysis methods is important for properly simulating a model. This chapter concludes the background theoretical information of the thesis and the following chapters discuss the main components of the project and a conclusion section.
Chapter 4

Software Development

Finite Element Analysis (FEA) and Computer Aided Design (CAD) programs are very powerful tools for modeling and analyzing structures. These types of software enable the user to develop prototype designs without having to construct expensive physical models and perform time consuming experiments. Although useful and advantageous, there are some caveats with regards to programs like ABAQUS and SolidWorks. Depending on the complexity of the design, modeling a structure in CAD or FEA software can be quite tedious. In most cases, users of CAD and FEA programs build and modify only a handful of designs for analysis. While it is true that programs like ABAQUS and SolidWorks can do some forms of design space exploration (such as section and material property variations) relatively easily, significant geometry variations are rather difficult to perform. In ABAQUS and SolidWorks, the user has to perform numerous tasks to configure a model for Finite Element Analysis. Often times, if the geometry is altered the user has to rebuild the entire model. Reconstructing the model repeatedly is very inconvenient. The fundamental goal of this project was to develop a technique to efficiently explore the design space of hexapod structures. MATLAB was coupled with ABAQUS to avoid the issues associated with standalone FEA software. The sections that follow describe the basic FEA process from a computer standpoint, the ABAQUS environment, and the coupling of ABAQUS with MATLAB.
4.1 Finite Element Analysis with Software

The procedures for performing Finite Element Analysis on a computer are the following:

1. Develop an Initial Layout of the Problem:
   - Define the problem
   - Compile information about the problem
   - Calculate an analytical solution if feasible

2. Model
   - Build the model by defining the geometry and element type
   - Define the material parameters
   - Define the loads, boundary conditions, and other constraints

3. Mesh (Discretization)
   - Divide the model into nodes (Seeding)
   - Divide the model into elements (Meshing)

4. Analysis and Setup
   - Determine the type of analysis to be performed (Loading, Buckling, Modal Analysis, etc)
   - Apply the appropriate settings in the program to perform the analysis

5. Execution (Run by the Computer)
   - Check inputs
   - Define element stiffness matrices
   - Assemble global stiffness matrix
   - Apply boundary conditions
   - Invert stiffness matrix and solve for displacements
   - Calculate parameters for the chosen analysis type (Stress/Strain, Modes/Frequencies, etc)

6. Post-Processing
   - Verification of results
   - Perform Mesh Convergence (Reducing the mesh size until the results converge)

4.2 ABAQUS Software

ABAQUS has two modeling environments, namely, the Graphical User Interface (GUI) and the command module. In both cases, an input file is generated which contains all of the necessary information describing the geometry and FEA analysis parameters of the model. Upon execution, ABAQUS reads the input file and
performs the commands within the code. The GUI allows the user to construct a model interactively with the built in CAD tools, define the FEA parameters for the study, and then automatically generate the input file or using other file creation software. The command module requires the user to build the model and set up the analysis parameters by directly coding the input file. For simple structures, coding the input file is a viable option to construct a model, but for more complicated geometries it is very difficult. Figure 4.1 shows the Graphical User Interface and Figure 4.2 shows the coding environment. Note that Figure 4.1 shows a simple tripod structure in the GUI and Figure 4.2 is not the corresponding input file, but rather a very basic description of the key elements within a generic input file.

4.2.1 ABAQUS Graphical User Interface

![AB AQUS GUI](image)

Figure 4.1: ABAQUS CAE graphical user interface.

As seen in Figure 4.1, the ABAQUS GUI environment is composed of several modules. The Parts module is the CAD interface where individual components can be drawn with geometric parameters. The Material module
is where the modulus of elasticity, density, Poisson’s ratio and other material parameters are defined. The Section and Profile modules define the type of element and cross-sectional geometry for beams, respectively. The step module is used to define the type of simulation to be run, such as frequency analysis, buckling, loading, etc. The Assembly module is where the complete structure is constructed from the individual parts. To define connections between different nodes of different parts, the Constraint module is used. Finally, the Job module is used to submit a model for analysis (this is the step where the input file is created).

Using the GUI for large design space exploration is problematic because of the issues that can arise from the Constraint module. After an assembly is built and the constraints have been applied, alterations of the components in the Parts module cause serious errors. Typically, the constraints are used to join the individual parts together. These constraints become corrupt when the geometry changes and thus have to be deleted and redefined. Furthermore, ABAQUS only allows planar sketching, which forces the user to make several components and join them together using constraints in the Assembly module.

### 4.2.2 ABAQUS Code Environment

Figure 4.2 displays the very basic components that comprise the input file. Depending on the complexity of the geometry and meshing parameters of the model, the input file can be very large. For instance, the input file for the simple hexapod structure seen in Figure 4.1 amounts to several pages of coding. The components of the code include node, element, material, section, boundary condition, and analysis definitions.
### 4.3 MATLAB Integration with ABAQUS

With an understanding of the difficulties associated with ABAQUS, a second program, MATLAB, was implemented as a solution. MATLAB is an interpreter coding language with many useful features including the ability to read and write text files, generate graphs, and run Windows executable files within the MATLAB Command Window. The purpose of the MATLAB code was to simplify and optimize the structural design space exploration procedure. In order to accomplish this task, the MATLAB code had to be capable of the following:

1. **Define Nodal Coordinate Matrix**: Create a matrix or matrices of individual nodes that describe the geometry of the structure.

2. **Define Element Matrix**: Create a matrix or matrices that describes how the nodes are connected. Connecting nodes are referred to as elements.
3. Assign Element Types and Section Profiles
4. Apply Material Parameters
5. Apply Boundary Conditions
6. Organize All Data into the Proper ABAQUS Input File (.inp) Format

The MATLAB code evolved throughout the course of the project and eventually arrived to a well organized modular program. The main calling program, calling.m, acts as the executable file and reads in the inputs from a text file. Calling.m utilizes several other sub functions that help to create the final input files. The remainder of this chapter describes the different functions that the main executable file, calling.m, uses to generate a final ABAQUS input file. Figure 4.3 helps to define some of the key variables associated with the functions, and is a useful reference for the discussion. The flowchart shown in Figure 4.4 is a general representation of how the code works.

Figure 4.3: Geometry variables for the hexapod-tripod structure.
4.3.1 Node Generation Function: node_gen.m

The first program written was a function that created evenly spaced nodes between two points in 3D space. The function, node_gen.m, receives four inputs: (1) coordinates 1, (2) coordinates 2, (3) element size, and (4) position in matrix. The position in the matrix parameter ensures that nodes are not created twice. Essentially this function creates a parametric equation of a line in space connected by the two points and evenly divides it into nodes with corresponding 3D coordinates. Figure 4.5 shows an example of how the
function works. The two points, \( p_1 \) and \( p_2 \), are connected via a series of evenly spaced nodes defined by the array \( t \). Node\_gen.m is a useful function for discretizing line segments and is used frequently in other parts of the code.

\[
\begin{align*}
&x = \{1,2,3\}; \\
&y = \{4,5,6\}; \\
&z = \{7,8,9\}; \\
&t = \{10,11,12\}; \\
&\text{Nodes} = \text{node\_gen}(p_1, p_2, t, x, y, z) \\
&\text{scatter3(Nodes(:,1), Nodes(:,2), Nodes(:,3))}
\end{align*}
\]

Figure 4.5: Example execution of the node\_gen.m matlab file. Results are plotted using the scatter3 command in MATLAB.

### 4.3.2 Main Node Generation: main\_node\_gen.m

The second function created was used to define the *main nodes* of the tripod structure, which are shown in Figure 4.6. The main nodes are only the long span members and the central triangle portion of the structure. The main\_node\_gen.m function receives the input geometry: \( d \), \( u_1 \), \( u_2 \), \( u_3 \), \( t_1 \), and \( t_2 \) to produce a matrix of node numbers and corresponding coordinates (scalars values except for \( t_1 \) and \( t_2 \)). These variables are defined in Figure 4.3. Main\_node\_gen.m takes advantage of structural symmetry by utilizing a rotational transformation matrix applied to inputs. The four scalar inputs \( d \), \( u_1 \), \( u_2 \), and \( u_3 \) are converted to vectors within the code and then, with the application of the rotational transformation matrix, the rest of the reference vectors are generated.
4.3.3 Main Element Generation: main_element_gen.m

Main_element_gen.m is a function that takes the main node matrix and creates a matrix of numbered elements. This function was designed to connect the main nodes together using the corresponding identification number associated with each node point. Due to the complicated geometry of the structure, main_element_gen.m had to be able to account for the position within the main node matrix such that it could intelligently create elements from the proper adjacent nodes. This issue is made apparent in Figure 4.7. Essentially, it is impossible to draw the structure using a continuous line of nodes and therefore it was necessary to be able to keep track of node locations when generating elements.
4.3.4 Triangular Reinforcement Node and Element Generation: tri_node_gen.m and tri_element_gen.m

After the main nodes and elements are generated, the next set of functions, tri_node_gen.m and tri_element_gen.m, create the triangular reinforcement members that connect along the span members. The triangular reinforcement members are shown in Figure 4.8. Tri_node_gen.m takes in four inputs: (1) main_nodes, (2) $t_1$, (3) $t_2$, and (4) $nt$. Main_nodes is the matrix that contains the node numbers and coordinates for the main nodes of the structure (previously defined). $t_1$ and $t_2$ are the seed sizes (number of nodes along a line) and $nt$ specifies the number of triangular reinforcements per span. Tri_nodes_gen.m interprets the main_nodes matrix and the number of reinforcement triangles and fills in evenly spaced reinforcement triangles along the main spans of the structure. The resulting nodes produced are connected together via tri_element_gen.m. The purpose of excluding the triangular reinforcements from the set of main nodes and elements was to allow the simpler structures to be created without the reinforcements.

![Figure 4.8: Triangular reinforcement nodes isolated from the rest of the structural nodes. Plotted using scatter3 in MATLAB.](image)

4.3.5 Cross Reinforcement Node and Element Generation: cross_node_gen.m and cross_element_gen.m

The next set of nodes and elements generated are the cross reinforcement members. The cross reinforcement nodes are shown in Figure 4.9. The procedure for creating the nodes and elements is identical to the
triangular reinforcement section. The cross reinforcements were treated as separate nodes from the main structure to allow for exclusion from the system.

![Image of isolated cross reinforcement nodes](image.png)

Figure 4.9: Cross reinforcement nodes isolated from the rest of the structural nodes. Plotted using scatter3 in MATLAB.

### 4.3.6 Vertical Node and Element Generation: vertical_node_gen.m and vertical_element_gen.m

With the nodes and elements defined for the main structure, triangular reinforcements, and cross reinforcements, the next set of members defined are the vertical struts. The vertical struts connect the upper and lower tripod sections together. The purpose of excluding this portion of the structure from the main nodes was due to an issue with ABAQUS. When defining beam section orientation, a normal is defined such that ABAQUS understands how the cross-sectional area is oriented relative to the beam. For vertical members, a unique normal has to be defined. It was most convenient to define two normals, one for non-vertical members and one for vertical members, and to isolate these nodes and elements. The vert_node_gen.m and vert_element_gen.m functions operate identically to the main_node_gen.m and main_element_gen.m functions. Figure 4.10 shows the vertical nodes isolated from the rest of the structure. Using superposition, all of the nodes are compiled together in Figure 4.11.
4.3.7 Input File Generation: input_file_gen.m

When all of the nodes and elements are generated, the information is sent to a function called input_file_gen.m. At this point, there are two matrices for the nodes, and four matrices for the elements. The main nodes, triangular reinforcement nodes and cross reinforcement nodes are compiled into one large matrix called Nodes_h and the vertical nodes are kept in a separate matrix, Nodes_v (to enable unique normal deformations for cross-sectional beam profiles). The elements corresponding to main, triangular, cross, and vertical members are kept in unique matrices, main_elements, tri_elements, cross_elements, and vert_elements, respectively. Input_file_gen.m takes these matrices as inputs along with three sets of member diameters and thicknesses. There are three different types of diameters and thicknesses for the three different types of members defined in Figure 4.12. The other inputs for the function input_file_gen.m are density, modulus of elasticity, Poisson’s ratio, and point mass locations. Input_file_gen.m generates the proper formatting for the input file by utilizing a series of printing commands in MATLAB. Three important printing functions used in input_file_gen.m are dlmwrite (for printing delimited matrices), fprintf (for printing text), and sprintf (for printing text with changing variables). The input file is saved into a uniquely named text file to be sent to ABAQUS.
Figure 4.11: All structural nodes superimposed together using MATLAB scatter3 function. Main nodes, triangular reinforcement nodes, cross reinforcement nodes, and vertical nodes colored green, blue, red, and yellow respectively.

Figure 4.12: Reinforced hexapod-tripod model from ABAQUS GUI showing the three unique structural member definitions.
4.3.8 Main Executable File: calling.m

The program that orchestrates all of the functions previously described is the main executable file, calling.m. Upon execution, calling.m generates a directory, ABAQUS Jobs, to save all of the generated data. It then reads in a text file, inputs.txt, which houses all of the geometric and material information for the tripod structures. Each line of inputs.txt corresponds to a unique job (see Figure 4.13). A job, in terms of ABAQUS, represents a unique hexapod-tripod design with a set of defined geometric, mass, and material parameters.

Calling.m cycles through each input line and generates a uniquely named input file for each job. Once all of the input files are generated, calling.m then creates a .bat file which is a batch file containing the names of all of the input files and the proper syntax for ABAQUS to read and execute each job consecutively. In summary, the calling.m program reads in a text file of inputs similar to inputs.txt, generates a series of input files (.inp format), and finally creates a .bat file to be sent to ABAQUS. An example batch file corresponding to the input.txt file above is shown in Figure 4.14. The function “abaqus” indicates to the Windows system command prompt to execute abaqus and “interactive” ensures that each job will be run consecutively versus in parallel.

Figure 4.13: Example input text file showing the different parameter definitions.

Calling.m cycles through each input line and generates a uniquely named input file for each job. Once all of the input files are generated, calling.m then creates a .bat file which is a batch file containing the names of all of the input files and the proper syntax for ABAQUS to read and execute each job consecutively. In summary, the calling.m program reads in a text file of inputs similar to inputs.txt, generates a series of input files (.inp format), and finally creates a .bat file to be sent to ABAQUS. An example batch file corresponding to the input.txt file above is shown in Figure 4.14. The function “abaqus” indicates to the Windows system command prompt to execute abaqus and “interactive” ensures that each job will be run consecutively versus in parallel.
Figure 4.14: Example batch file, created from MATLAB, to be sent to ABAQUS command module.

After the batch file is submitted, ABAQUS produces several files for each job during the analyses. The file that contains the results of the analyses is the .dat file. Calling.m also calls two other functions, frequency_search.m and mass_search.m, not previously discussed. These files are post processing tools that search through all of the ABAQUS .dat files to find the Natural Frequencies and overall masses from each job. The code for these functions can be seen in the Appendix.
Chapter 5

Experimental Validation

Using MATLAB and ABAQUS to explore the design space of tripod structures is an efficient and useful method, but the results require validation from experimental testing. There are numerous techniques for measuring the natural frequencies of structures. The vibrations laboratory at California Polytechnic State University, San Luis Obispo has two types of modal analysis systems: (1) Impulse Hammer and (2) Slip Table. In both systems, an input force or displacement is applied to the structure and an accelerometer is used to measure the response. For the Impulse Hammer method, the model is struck with a force transducer hammer and accelerometer mounted on the structure measures the vibration response. The shake table technique applies a random noise input displacement to the base of the model and an accelerometer mounted on the structure measures the vibration response. This chapter will discuss the physical model construction process, software predictions (ABAQUS and SolidWorks), testing hardware, and the experimental analyses.

5.1 Physical Model: Construction

The physical model was designed after the basic tripod seen in Figure 5.1. Despite the fact that this design excludes the triangular and cross reinforcement members seen in Figure 1.2, it was deemed sufficient for validating the results of the Finite Element Analysis software. Furthermore, this model captures the key geometric components for the tripods under investigation, hence, for validation purposes, there was no major
gain by increasing the complexity of the geometry. Lastly, the MATLAB-interfaced-ABAQUS code was capable of generating tripods with or without the reinforcement members, so software predictions could easily be made.

Figure 5.1: ABAQUS Hexapod base model.

After deciding on the basic layout for the model, the next phase in the construction process was determining the dimensions and material. It was critical to make a model that fit within the constraints of the modal analysis hardware. In other words, the model had to fit onto the slip table and be capable of mounting onto the table surface with bolts. Figure 5.2 shows the overall surface dimensions and bolt pattern of the slip table used in the experiments. The radius, $r$, of the tripod was chosen to be 13 inches such that, at any orientation on the table, the ends of the tripod legs would not be positioned over one of the tapped bolt holes (see Figures 4.3 and 5.2 for reference). The height and upper/lower section spacing, $u_2$ and $u_3$, were less of a concern and were chosen arbitrarily (see Figure 4.3 for reference).
Similar to the simulations, the physical model used circular cross-section profiles for the structural members. In addition, several materials were considered for the design. Initially, carbon fiber was chosen, but due to the complicated and, often times, inconsistent orthotropic material properties, it was discarded as an option. The introduction of material inconsistencies into the physical model would have detracted from the main objective of the experimental validation. For practical reasons, brass tubing was chosen for the material of the physical model. Brass tubing is both inexpensive and relatively easy to modify for construction purposes. Furthermore, brass can be soldered or brazed instead of welded which is very convenient for joining thin-walled members. As seen in Figure 5.3, the outer diameter of the chosen tubing was 0.25 inches with a wall thickness of 0.014 inches.
The tubes were cut to the proper lengths and the ends were modified to join well at the intersections. The ends of the tripod legs were kept unbeveled because of the acute angle, as seen in Figure 5.4. Figure 5.5 (left) shows where the joints were soldered together using flux-core solder and a soldering iron. In the software simulations, at the intersections of beam elements, the cross-sectional profiles are allowed to overlap (see Figure 5.5 (right)). Accordingly, the two-dimensional profiles of the members are for aesthetic purposes only and the members intersect at a single point. This is one of the major discrepancies between the computer simulation and the physical model. The mounting plates, used to secure the model to the table, were the last components added to the structure. L-Brackets were used to attach the span members to the plates. Due to the large surface area and heat dissipation, a propane torch was used to braze the brackets to the plates. Figure 5.6 shows the steel plates and L-Bracket attachment with the proper holes drilled for
mounting onto the slip table. The final configuration is shown in Figure 5.7 mounted onto the slip table surface.

Figure 5.6: Image of the L-Brackets brazed to the ends of the physical model legs. Note that the hexapod feet were brazed to the plates where as the other joints were soldered because of large heat dissipation issues.

5.2 Physical Model: Software Predictions

Once again, the purpose of the physical model was to validate the ABAQUS FEA results. In addition, the physical model was analyzed in SolidWorks as a means of cross-validating softwares. Because the model was not built with perfect accuracy, measurements were taken with calipers to estimate the geometric variations. It was determined that approximately 5% variation of the main structural dimensions was an adequate estimation. Table 5.1 shows the mean value for the dimensions of the physical model.
Figure 5.7: Completed physical model bolted to the Ling Electronics shake table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Average Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>33 cm</td>
</tr>
<tr>
<td>( u_1 )</td>
<td>8.55 cm</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>12 cm</td>
</tr>
<tr>
<td>( u_3 )</td>
<td>4.75 cm</td>
</tr>
<tr>
<td>( m_{1t} )</td>
<td>0.03556 cm</td>
</tr>
<tr>
<td>( m_{1d} )</td>
<td>0.635 cm</td>
</tr>
<tr>
<td>( \rho )</td>
<td>0.00874 kg cm(^{-3})</td>
</tr>
<tr>
<td>( E )</td>
<td>1.1E9 N cm(^{-2})</td>
</tr>
<tr>
<td>( \nu )</td>
<td>0.331</td>
</tr>
</tbody>
</table>

Table 5.1: Design constraints for Hexapod Optimization.

5.2.1 ABAQUS Simulations

The MATLAB software, used to interface with ABAQUS, generated a series of input files of the physical model with 5% geometric uncertainties. All of the files were run in ABAQUS and the results were compiled for comparison with experimental results. It was important to understand whether or not the accelerometer mass would have an effect on the natural frequency and, as a result, four series of ABAQUS simulations were run:

1. Model without accelerometer point mass
2. Model with accelerometer point mass on upper node intersection
3. Model with accelerometer point mass on lower node intersection
4. Model with accelerometer point mass on upper span member

Figure 5.8 shows the four series with the accelerometer point mass locations and Figure 5.9 shows the corresponding results of the simulations. Finally, Figure 5.10 shows the first mode shape.

![Diagram](image)

Figure 5.8: Different locations for the accelerometer. This image shows where the point mass was positioned in ABAQUS and where the accelerometer was positioned on the physical model.

5.2.2 SolidWorks Simulation

One SolidWorks model was created as a supplement to the validation process. The SolidWorks simulation used beam elements and there was no point mass included. The dimensions of the model were taken from the average of the physical model geometry measurements. The technique for creating a frame structure in SolidWorks is typically done with weldments, which is accomplished by creating a 3D sketch of the structure and applying a section profile to the line drawing. This process is seen in Figure 5.11. Weldments allow the user to create intricate frame structures without having to use complicated extrusion and cutting features. Unlike the ABAQUS models, the ends of the tripod legs do not come to a single point due to the issues associated with weldment rendering. Despite this discrepancy, the results were nearly identical to the ABAQUS simulation without a point mass. The results of the first mode natural frequency is shown in Figure 5.12 below.
Figure 5.9: ABAQUS results from the various simulations of the physical model. (a)-(d) correspond to accelerometer point mass locations (see Figure 5.8). The only significant change in natural frequency occurred when placing the point mass midspan of the leg member (most likely due to the fact that this was the excitation location for the first mode).

Figure 5.10: ABAQUS mode shape result for the simulations.

### 5.3 Hardware

The list below is a description of the various hardware components used for the modal hammer and shake table analyses.
Figure 5.11: *SolidWorks* three-dimensional sketch and corresponding weldment model.

Figure 5.12: *SolidWorks* first mode natural frequency result and corresponding mode shape. The first mode natural frequency was 194.74 Hz.

1. Endevco 30927 Impulse Hammer
2. Ling Electronics Slip Table
3. PCB 338C04 Accelerometer
4. PCB 4-Channel, Line-Powered, ICP Sensor Signal Conditioner
5. Dactron LDS Focus II Data Acquisition
6. Dell Laptop with RT Pro 6.21
5.4 Modal Analysis: Impact Hammer

After constructing the physical model and predicting the natural frequencies in ABAQUS and SolidWorks, the next phase of the validation process was the experimental testing. Initially, the impact hammer technique was used to obtain a survey of the physical model natural frequency behavior. The impact hammer method is a relatively simple approach for analyzing the natural frequencies of small, lightweight structures. Essentially, a measured force is applied to the model via the impact hammer and the response is measured by the accelerometer mounted on the structure. The two signals, the impact force and the acceleration, are processed by a data acquisition system and vibration analysis software. The following is a description of the impact hammer modal analysis setup:

- **Hardware Configuration**

  1. The physical model was mounted onto the shake table as a means of conveniently anchoring the system by the tripod legs.
  2. The PCB 338C04 accelerometer was mounted onto a specified location on the structure using hot glue.
  3. The ICP cables attached to the impact hammer (Channel 1) and accelerometer (Channel 2) were connected to the PCB Signal Conditioner and BNC cables were used to attach the Signal Conditioner to the Dactron LDS Focus II DAQ.
4. The Dactron LDS Focus II was attached to the Dell Laptop via USB.

- RT Pro 6.21 Configuration

1. The Real-Time Measurement: Digital Signal Analyzer module was chosen for the analysis in RT Pro.
2. The two channels, Impact Hammer (Channel 1) and Accelerometer (Channel 2) were configured in the signal setup. Time Capture, Fast Fourier Transform, and Power Spectra were selected for both signals.
3. In Channel Parameter Settings, the Impact Hammer (Channel 1) was set to 0.3 Maximum Voltage and ICP 0.7 AC Coupling. The accelerometer was set to 1.0 Maximum Voltage and ICP 0.7 AC Coupling. All of the other parameters were kept as default.
4. The Sampling Parameters were set to Forced/Exponential and the frequency span was set to 500 Hz and 1024 points.
5. Averaging was set to Linear with 4 Averaged Frames. This setting required the user to strike the system 4 times with the impact hammer to obtain a result.
6. The trigger was set for Channel 1 Analog Input at a 2% level. This configuration set a minimum input force on the hammer before a strike would be registered.
7. An FRF Bode Window (Transfer Function) was configured between Channels 1 and 2 (H1,2,1(f)). The Bode Window measures ratio of the impact hammer and accelerometer signals.
8. The MDA Export Setup was configured to automatically export the Bode data after each 4-averaged sample. The output was chosen to be .uff ASCII (a standard text file with three columns: (1) Frequency, (2) FRF Amplitude, and (3) Phase).

For a given location of the accelerometer, each member of the physical model (excluding the vertical members) was struck four times and the data was recorded for each. Figure 5.14 shows the acquisition process for the upper long span members and Figure 5.15 shows the same process for the upper central triangle members (this procedure was repeated for the corresponding lower members).

Figure 5.14: Modal hammer analysis strike locations for the upper long span members.
In order to capture a well averaged survey of the natural frequencies, the impact hammer modal analysis data was collected from multiple orientations and locations of the single axis accelerometer. Positioning the accelerometer at various orientations and locations on the structure influenced the amplitude and frequency of the response. A portion of the results from the impact hammer analysis are shown in Figures 5.16-5.21.

Figure 5.16: Modal hammer analysis frequency response plot (Transfer Function $H_{1,2}$ vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the upper long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.
Figure 5.17: Modal hammer analysis frequency response plot (Transfer Function H1,2 vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the upper central triangle members. The accelerometer was oriented parallel with the shake table. The amplitude response was very low and the first mode natural frequency was difficult to extrapolate from this plot.

Figure 5.18: Modal hammer analysis frequency response plot (Transfer Function H1,2 vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the lower long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.
Figure 5.19: Modal hammer analysis frequency response plot (Transfer Function $H_{1,2}$ vs. frequency). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)) and the strike location was on one of the upper long span members. The accelerometer was oriented perpendicular with the shake table. The first amplitude peak represents the first mode natural frequency.

Figure 5.20: Modal hammer analysis frequency response plot (Transfer Function $H_{1,2}$ vs. frequency). The accelerometer was positioned at the midspan (see Figure 5.8 (d)) and the strike location was on one of the upper long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.
Figure 5.21: Modal hammer analysis frequency response plot (Transfer Function H1,2 vs. frequency). The accelerometer was positioned at the midspan (see Figure 5.8 (d)) and the strike location was on one of the lower long span members. The accelerometer was oriented parallel with the shake table. The first amplitude peak represents the first mode natural frequency.

5.5 Modal Analysis: Shake Table

Following the impact hammer modal analysis, the shake table was used as a supplemental technique to measure the natural frequencies of the physical model. The shake table method uses two accelerometers, one attached to the table surface and another attached to the structure. The table and structure accelerometers measured the input and response accelerations respectively. The two signals from the accelerometers were processed using the DAQ system previously described in the modal hammer analysis section. The following is a description of the hardware and software setup for the shake table analysis.

- **Hardware Configuration**
  1. The physical model was mounted onto the shake table.
  2. The PCB 338C04 accelerometer was mounted onto a specified location on the structure using hot glue.
  3. The PCB U308B accelerometer was mounted to the shake table using the threaded stud.
  4. The ICP cables attached to the table accelerometer (Channel 1) and structure accelerometer (Channel 2) were connected to the PCB Signal Conditioner and BNC cables were used to attach the Signal Conditioner to the Dactron LDS Focus II DAQ input channels.
  5. A third BNC cable was attached from the Dactron LDS Focus II output channel to the shake table controller input signal channel.
6. The Dactron LDS Focus II was attached to the Dell Laptop via USB.
7. The electrical switches and compressed air were turned on to enable power to the shake table.

- RT Pro 6.21 Configuration

1. The Real-Time Measurement: Digital Signal Analyzer module was chosen for the analysis in RT Pro.
2. The two channels, table accelerometer (Channel 1) and structure accelerometer (Channel 2) were configured in the signal setup. Time Capture, Fast Fourier Transform, and Power Spectra were selected for both signals.
3. In Channel Parameter Settings, both accelerometers were set to 1.0 Maximum Voltage and ICP 0.7 AC Coupling. All of the other parameters were kept as default.
4. The Sampling parameters were set to Forced/Exponential and the frequency span was set to 500 Hz and 1024 points.
5. Averaging was set to Linear with 40 Averaged Frames. This setting required the shake table to cycle 40 times to obtain a result.
6. The Waveform Source output channel was set to Random White Noise and the amplitude was set to 1 Volt.
7. An FRF Bode Window (Transfer Function) was configured between Channels 1 and 2 (H1,2,1(f)). The Bode Window measures ratio of the impact hammer and accelerometer signals.
8. The MDA Export Setup was configured to automatically export the Bode data after each 4-averaged sample. The output was chosen to be .uff ASCII (a standard text file with three columns: (1) Frequency, (2) FRF Amplitude, and (3) Phase).

The structure accelerometer was positioned at various locations and orientations on the model and the data were recorded. Figure 5.22 shows the two accelerometers used in the shake table analysis. Figures 5.23, 5.24, and 5.25 show a portion of the results of the shake table analysis.
Figure 5.22: Ling Electronics shake table setup. Shake table and drive system (upper left). Controller system (upper right). Accelerometer locations (bottom).

Figure 5.23: Shake table analysis frequency response plot (Transfer Function H1_2 vs. frequency). The structure accelerometer was positioned on the upper node intersection point shown in Figure 5.8 (b) and the shake table was sent a white noise signal.
Figure 5.24: Shake table analysis frequency response plot (Transfer Function $H_{1 \rightarrow 2}$ vs. frequency). The structure accelerometer was positioned on the lower node intersection point shown in Figure 5.8 (c) and the shake table was sent a white noise signal.

Figure 5.25: Shake table analysis frequency response plot (Transfer Function $H_{1 \rightarrow 2}$ vs. frequency). The structure accelerometer was positioned in the midspan location shown in Figure 5.8 (d) and the shake table was sent a white noise signal.
5.6 Discussion and Conclusions

The experimental validation was a three part process: (1) Construction of the Physical Model, (2) ABAQUS and SolidWorks Simulations, and (3) Experimental Vibration Analysis. The computational and experimental modal analysis samples were compiled into an Excel spreadsheet and the results were compared. Individual plots, similar to the ones seen in Sections 5.4 and 5.5, were made for each averaged sample. The first peak amplitude value for each sample was recorded into another database as to compare the overall first mode natural frequencies of the various experimental analyses. Figures 5.26, 5.27, and 5.28 show the collected data from the ABAQUS simulations and experimental vibration tests. It is important to note that the data were separated by location of the accelerometer. In other words, the ABAQUS simulations for the point mass location seen in Figure 5.8 (b) was compared to the Impact Hammer and Shake Table data with the corresponding accelerometer location. The same procedure was applied to the lower intersection node and the midspan locations. Figures 5.26, 5.27, and 5.28 correspond to the upper node, lower node, and midspan locations respectively.

![Graph showing data comparison](image)

Figure 5.26: Cumulative plot of both testing methods (Hammer and Shake Table) along with the simulation results from ABAQUS (5% geometry variation). The accelerometer was positioned at the upper node intersection (see Figure 5.8 (b)).
The impact hammer modal analysis produced relatively consistent results with respect to the simulations, however, the excited mode depended on the strike location and accelerometer placement. Because the input force was applied to a single member on the structure, the first mode natural frequency was not always excited. As seen in Figures 5.10 and 5.12, the first natural frequency mode shape occurred on the upper long span members. Having the accelerometer positioned on the upper node intersections (as seen in
Figure 5.8 (b)) and striking the upper long span members (as seen in Figure 5.14) typically excited the first mode. The general response for this configuration is shown in Figure 5.16. With the accelerometer in the same location and striking the lower, long span members, only sometimes was the first mode excited. In addition, striking the center triangle members (upper or lower) typically bypassed the first mode and the resultant amplitude response was rather low (see Figure 5.17). Figures 5.18 and 5.19 show the resultant frequency response for the accelerometer positioned on the lower node intersection point seen in Figure 5.8 (c). In general, there was not a significant difference between the cases where the accelerometer was on the upper or lower node intersection point, but rather the response depended more considerably on the strike location. The last two plots, Figures 5.20 and 5.21 illustrate the effect on natural frequency when positioning the accelerometer midspan of the long, leg members. This configuration reduced the first mode natural frequency and directly correlated with the simulations.

The shake table analysis produced results that were very consistent with the ABAQUS simulations. Unlike the modal hammer analysis (which sent an input force to a single member of the structure), the shake table testing sent a distributed, lateral input displacement to the entire base of the physical model. Consequently, the first mode natural frequency results from the shake table showed very little deviation from the simulation predictions. As described previously in Section 5.5, the accelerometer was attached to the structure at various locations and the shake table vibrated with a random noise signal. Individual frequency response plots, shown in Figures 5.23, 5.24, and 5.25, represent the structure’s vibration response for accelerometer positions: upper node intersection, lower node intersection, and midspan respectively. The results were very similar to the modal hammer analysis; however, the span of frequencies for each case was much smaller. In other words, placing the accelerometer at any of the three upper (or lower) node intersection points produced nearly the same first mode natural frequency results. Similarly, placing the accelerometer on the six upper (or lower) midspan points yielded nearly constant results. In Figures 5.26, 5.27, and 5.28, the consistency and accuracy between the simulation and the shake table results are highlighted.

Finally, the SolidWorks model was used to cross validate software predictions with ABAQUS. The model configured in SolidWorks used the average geometry values from the ABAQUS models, however, the ends of the hexapod-tripod legs were treated as four points instead of one unifying node. As seen in 5.12, the first mode natural frequency was 194 Hz. Despite the minor discrepancy with the hexapod-tripod leg endings, there was no significant difference between the ABAQUS and SolidWorks simulations.

In conclusion, the experimental validation was deemed successful and sufficiently legitimized the MATLAB code and ABAQUS simulations.
Chapter 6

Database 1: Hexapod General Geometry/Material Variations

6.1 Introduction

In order to characterize the natural frequency behavior of an arbitrary hexapod structure, a database was created showing the individual effect of each geometric and material parameter. Both the base and reinforced versions of the hexapod structures (see Figures 1.1 and 1.2) were studied and the variables listed in Table 6.1 were the parameters of interest. The structures were tested without the inclusion of point masses (to simulate a secondary mirror).

An arbitrary default configuration was chosen with values listed in Table 6.2 and the individual parameters were varied independently. In other words, all geometric and material quantities were maintained at the default configuration while altering one variable of interest at a time. The default configuration used material parameters similar to carbon fiber reinforced polymer and the geometry was scaled to the size of a secondary support for a large telescope. The purpose of this study was to understand how the natural frequency of hexapod structures (base and reinforced versions) varied with respect to physical dimensions and material properties. Furthermore, it was of interest to know how the scale of the structure affected the natural frequency response. In order to characterize the general natural frequency response of hexapod structures,
it was important to show if the scale of the geometric and material parameters would change the general natural frequency trend. The list below describes the analyses that were run.

1. Base Model with default parameters from Table 6.2 varying each parameter individually.

2. Scaled Base Model with default parameters from Table 6.2 varying each parameter individually. This model had geometric parameters that were 10x smaller than the original Base Model and the material was kept constant.

3. Reinforced Model with default parameters from Table 6.2 varying each parameter individually.

### 6.2 Model Geometry and Material Parameters

As mentioned previously, Database 1 was designed to show the effects on natural frequency from individual geometric and material parameters. The three models studied, (1) Base Model, (2) Scaled Base Model, and (3) Reinforced Model, were analyzed using a systematic procedure. Each model maintained the default parameters listed in Table 6.2 and the individual parameters were varied independently. The plots in Section 6.3 show the ranges for each parameter and the corresponding natural frequency response.

<table>
<thead>
<tr>
<th>Base Model</th>
<th>Reinforced Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
<td>$d$</td>
<td>Overall Structure Radius</td>
</tr>
<tr>
<td>$u_1$</td>
<td>$u_1$</td>
<td>Central Triangle Length</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$u_2$</td>
<td>Lower Hexapod Height</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$u_3$</td>
<td>Lower/Upper Hexapod Spacing</td>
</tr>
<tr>
<td>$m_{1t}$</td>
<td>$m_{1t}$</td>
<td>Member 1 Thickness</td>
</tr>
<tr>
<td>$m_{1d}$</td>
<td>$m_{1d}$</td>
<td>Member 1 Diameter</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{2t}$</td>
<td>Member 2 Thickness</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{2d}$</td>
<td>Member 2 Diameter</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{3t}$</td>
<td>Member 3 Thickness</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{3d}$</td>
<td>Member 3 Diameter</td>
</tr>
<tr>
<td>$\rho$</td>
<td>$\rho$</td>
<td>Material Density</td>
</tr>
<tr>
<td>$E$</td>
<td>$E$</td>
<td>Material Modulus of Elasticity</td>
</tr>
</tbody>
</table>

Table 6.1: Base and Reinforced Hexapod Versions and corresponding variables.
Figure 6.1: Geometry variables for the hexapod-tripod structure.

6.3 Database 1: Base Model, Scaled Base Model, and Reinforced Model Results

Figures 6.2-6.13 show the Base Model, Scaled Base Model, and Reinforced Model natural frequency response with respect to individual geometric/material parameters. The Base Model and Reinforced Model are plotted with the actual natural frequency values from the simulations, whereas the Scaled Base Model is plotted with scaled natural frequency and geometric parameter values. Figure 6.2 (upper left, upper right, and lower right) show the results of the Base Model, Scaled Base Model, and Reinforced Model on independent graphs with the values from the simulations. Figure 6.2 (lower right) shows the same information, however, the Scaled Base Model data are scaled appropriately. All of the remaining plots follow the superimposed formatting except for the last four plots which show how the reinforcing members affected the natural frequency for the Reinforced Model.
Table 6.2: Default parameters for the general study of the base, scaled base, and reinforced hexapod structures.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Base Model</th>
<th>Scaled Base Model</th>
<th>Reinforced Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>12.5 m</td>
<td>1.25 m</td>
<td>12.5 m</td>
</tr>
<tr>
<td>$u_1$</td>
<td>3 m</td>
<td>0.3 m</td>
<td>3 m</td>
</tr>
<tr>
<td>$u_2$</td>
<td>5.09 m</td>
<td>0.509 m</td>
<td>5.09 m</td>
</tr>
<tr>
<td>$u_3$</td>
<td>1 m</td>
<td>0.1 m</td>
<td>1 m</td>
</tr>
<tr>
<td>$m_{1t}$</td>
<td>0.004 m</td>
<td>0.0004</td>
<td>0.004 m</td>
</tr>
<tr>
<td>$m_{1d}$</td>
<td>0.119 m</td>
<td>0.0119 m</td>
<td>0.119 m</td>
</tr>
<tr>
<td>$m_{2t}$</td>
<td></td>
<td></td>
<td>0.004 m</td>
</tr>
<tr>
<td>$m_{2d}$</td>
<td></td>
<td></td>
<td>0.119 m</td>
</tr>
<tr>
<td>$m_{3t}$</td>
<td></td>
<td></td>
<td>0.004 m</td>
</tr>
<tr>
<td>$m_{3d}$</td>
<td></td>
<td></td>
<td>0.119 m</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1550 kg m$^{-3}$</td>
<td>1550 kg m$^{-3}$</td>
<td>1550 kg m$^{-3}$</td>
</tr>
<tr>
<td>$E$</td>
<td>4.0E11 Pa</td>
<td>4.0E11 Pa</td>
<td>4.0E11 Pa</td>
</tr>
</tbody>
</table>

Figure 6.2: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. overall radius of hexapod ($r$).
Figure 6.3: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. center triangle length ($u_1$).

Figure 6.4: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. lower hexapod height ($u_2$).
Figure 6.5: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. lower/upper hexapod spacing ($u_3$).

Figure 6.6: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. member 1 thickness.
Figure 6.7: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. member 1 diameter.

Figure 6.8: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. material density.
Figure 6.9: Base Model, Scaled Base Model, and Reinforced Model natural frequency vs. Modulus of Elasticity.

Figure 6.10: Reinforced Model natural frequency vs. member 2 thickness.
Figure 6.11: Reinforced Model natural frequency vs. member 2 diameter.

Figure 6.12: Reinforced Model natural frequency vs. member 3 thickness.
Figure 6.13: Reinforced Model natural frequency vs. member 3 diameter.
6.4 Discussion and Conclusions

As mentioned previously, the three models tested were the base, scaled, and reinforced configurations. An arbitrary set of dimensions were chosen for each configuration that were convenient for future studies. The base model was used to show the effects of varying the dimensions of a general hexapod structure and the reinforced model was used to show the effects of adding reinforcement elements. The scaled model was employed to show the changes between hexapod size. In every case, the natural frequency of the scaled model was exactly ten times larger than the base configuration. This result is understood with the following equations for simple bar elements:

\[ k = \frac{AE}{L} \]

\[ m = \rho AL \]

\[ \omega = \sqrt{\frac{k}{m}} = \sqrt{\frac{E}{\rho L^2}} \propto \frac{1}{L} \]  \hspace{1cm} (6.1)

As seen in this series of equations, the natural frequency is proportional to the inverse of length parameter when the material parameters are held constant. In other words, because the scaled model was 10x smaller than the base model, the natural frequency was 10x larger.

In nearly every case, the reinforced model produced higher natural frequencies than the base model. As expected, the reinforcing members tended to strengthen the structure and provide additional stiffness. In Figure 6.2, there was an optimum radius configuration for the base and reinforced versions of the hexapod. This peak point was near the transition from a stable hexapod structure to a vertical tower truss because of decreased bending stiffness. Figure 6.3 showed that variations of the center triangle length had little effect on the base and scaled models, however, there was an optimum configuration for the reinforced models. This characteristic was true for \( u_2 \) and \( u_3 \) variations as seen in Figures 6.4 and 6.5. Alterations of \( m_{1t} \) and \( m_{1d} \) are shown in Figures 6.6 and 6.7. The base model was completely defined by Member 1, but the reinforced model had two other member types. Changing the thickness \( m_{1t} \) for the reinforced model had a significantly stronger effect on the natural frequency than the the base model. For the base model, increasing the thickness added additional mass to the system and lowered the natural frequency. Decreasing the thickness for both systems tended to increase the natural frequency up to a certain point. Increasing the member diameter \( m_{1d} \) increased the natural frequency of the systems drastically.

The natural frequency response for variations of the density and modulus of elasticity are shown
in Figure 6.8 and 6.9. Increasing the density caused a decrease in natural frequency because of added mass. Increasing the modulus of elasticity improved the stiffness for both models and the natural frequency response improved. For this variation, the natural frequency scaled as $\sqrt{\frac{E}{\rho}}$ because the geometry was held constant (see Equation 6.1). The remainder of the plots show the natural frequency response for variations of the reinforcing members. The base model was excluded because it lacked these members. The thickness variations for Members 2 and 3 shows that there are optimal configurations. The natural frequency response as a function of Member 2 and 3 diameters is a bit less obvious. One would expect that larger diameters would have produced higher natural frequencies, however, the opposite happens. This outcome is most likely due to the fact that the other members are being excited first. As seen in Figure 6.11 increasing Member 2 diameter causes the natural frequency response to asymptotically approach a specific value. This is because Member 2 becomes increasingly stiff, but because the other members are staying constant, they are being excited before Member 2 (the same result was found for Member 3). Because the natural frequency behavior was the only factor of interest, buckling was not considered in the thickness and diameter analyses.

The results of Database 1 provide useful information about natural frequency response as a function of a single parameter. This information is helpful in understanding more complicated, multi-variable variations in Database 2. The following chapter discusses the optimization process for the base and reinforced versions of the hexapod structure.
Chapter 7

Database 2: Hexapod Optimization

This chapter shows the optimization process for the Base and Reinforced Hexapod Models. To limit the number of variables, and consequently the design space, a scenario was chosen with a specific set of geometric and material constants. The criteria for the baseline design is listed in Table 7.1. In addition, three 333.33 kg point masses were placed at the lower triangular section of the hexapod structure to simulate a 1000 kg secondary mirror.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>12.5 m</td>
</tr>
<tr>
<td>$u_1$</td>
<td>3.0 m</td>
</tr>
<tr>
<td>$u_2$</td>
<td>5.09 m</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1550 kg m$^{-3}$</td>
</tr>
<tr>
<td>$E$</td>
<td>4.0E11</td>
</tr>
<tr>
<td>$\nu$</td>
<td>0.28</td>
</tr>
<tr>
<td>Blockage</td>
<td>1-2%</td>
</tr>
<tr>
<td>Mass</td>
<td>200-400 kg</td>
</tr>
</tbody>
</table>

Table 7.1: Design constraints for Hexapod Optimization.
Table 7.2: Base and Reinforced Hexapod Versions and corresponding variables for optimization.

### 7.1 Base Model Optimization

The first optimization was done using the Base Model (see Figure 1.1) configuration. It was important to know the maximum natural frequency that could be generated with the Base Model so that the results could be compared with the Reinforced Model. The Base Model is significantly less complicated than the Reinforced Model due to the fact that are fewer variables. The parameters that were allowed to be altered for the two models are shown in Table 7.2.

<table>
<thead>
<tr>
<th>Base Model</th>
<th>Reinforced Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_3$</td>
<td>$u_3$</td>
</tr>
<tr>
<td>$m_{1t}$</td>
<td>$m_{1t}$</td>
</tr>
<tr>
<td>$m_{1d}$</td>
<td>$m_{1d}$</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{2t}$</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{2d}$</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{3t}$</td>
</tr>
<tr>
<td>N/A</td>
<td>$m_{3d}$</td>
</tr>
</tbody>
</table>
Figure 7.1: Base Model first mode natural frequency vs. $m_{1t}$. The various series show the natural frequency with different $u_3$ values. The blockage was limited to 1% and the mass varied from 100 kg to 300 kg (larger $m_{1t}$ yielded larger mass. $m_{1d}$ was kept fixed to a diameter that corresponded to 1% blockage.
Figure 7.2: Base Model first mode natural frequency vs. $m_{1t}$. The various series show the natural frequency with different $u_3$ values. The blockage was limited to 2% and the mass varied from 200 kg to 300 kg (larger $m_{1t}$ yielded larger mass. $m_{1d}$ was kept fixed to a diameter that corresponded to 2% blockage.
Figure 7.3: Base Model first mode natural frequency vs. $m_{1t}$. This plot, combined with Figures 7.1 and 7.2, shows that there is a maximum natural frequency vs. thickness. Figures 7.1 and 7.2 don’t show this limit because only practical thicknesses were used.

Figure 7.4: Base Model mode shapes for three different $u_3$ values, (a) $u_3 = 0.5$, (b) $u_3 = 1$, (c) $u_3 = 2$.

7.2 Reinforced Model Optimization

Optimization of the Reinforced Model was a simple process to execute, but it is not so easy to explain. There are seven variables in the Reinforced Model (see Table 7.2), so the design space has seven dimensions. Unfortunately, the results can only be plotted with three or four dimensions at a time.

The optimization process went as follows:
1. Developed a set of inputs that efficiently explored the design space of the Reinforced Hexapod Model. In order to fit within the blockage and mass constraints, it was essential to control the member group diameters and masses. A technique was developed to base Member 2 and Member 3 corresponding diameter and thickness \((m_{2d}, m_{3d}, m_{2t}, m_{3t})\) respectively) off of a percentage of Member 1 diameter and thickness \((m_{1d} \text{ and } m_{1t})\). As seen in Appendix A.14 the diameter calculation was based off of the blockage. For a given \(m_{1d}\), an equation for blockage was solved to generate a set of diameters for Members 2 and 3. Once the diameters were calculated for each member group, the thicknesses were calculated to fit within the mass constraint. Using a very similar technique, the thicknesses, \(m_{2t}\) and \(m_{3t}\) were based off a percentage of \(m_{1t}\). Choosing the percentages for each case was done iteratively. With \(m_{1d}\) and \(m_{1t}\) set to 100%, the ranges for the other members were 25%, 45%, 65%, and 95%. Every possible combination was tested, which resulted in 256 cases. The remaining variable, \(u_3\), was set at 0.5, 1, 1.5, 2, 2.5, and 3 m. Two blockages were used, 1% and 2%. The total number of simulations that were run is shown in the equation,

\[
S = DTC \times UV \times BV = 256 \times 6 \times 2 = 3072 \quad (7.1)
\]

DTC is the number of diameter and thickness combinations, \(UV\) is the number of \(u_3\) values, and \(BV\) is the number of blockage values.

2. Once the inputs were set up in excel, they were imported into MATLAB and the automated simulation process was started.

3. Finally, the results were interpreted and organized.

Figures 7.5 and 7.6 are selected portions of the simulation results. All frequency response plots are of the first mode.
Figure 7.5: This plot shows the frequency response for all of the simulations that were run with 1% blockage. Each point within a series corresponds to a different combination of member diameter and thickness (see 7.21) and each series represents a specific $u_3$ value.

### 7.3 Discussion and Conclusions

The optimization process was split into two sections: (1) Base Model Optimization and (2) Reinforced Model Optimization. Several design constraints were applied to the system to limit the complexity and size of the design space. The design criteria and available variables are shown in Tables 7.1 and 7.2, respectively.

#### 7.3.1 Base Model

The results from the base model optimization were relatively simple to interpret because there were only three parameters to vary: $u_3$, $m_{1d}$, and $m_{1t}$. As seen in Figures 7.1 and 7.2, 1% and 2% blockage values were tested. Member 1 diameter was set constant by the blockage value, and the thickness was varied such that it stayed within the mass criteria range. Member 1 thickness, $m_{1t}$, and lower/upper hexapod spacing, $u_3$, were varied and the results are shown in Figure 7.1 and 7.2. Increasing the thickness in all cases tended to decrease the natural frequency response because the added mass did not improve the stiffness. Figure
Figure 7.6: This plot shows the frequency response for all of the simulations that were run with 2% blockage. Each point within a series corresponds to a different combination of member diameter and thickness (see 7.21) and each series represents a specific $u_3$ value.

7.3 shows that decreasing the thickness of Member 1 eventually causes the base model natural frequency to drop. This transition can be defined by the point when the secondary point masses dominate. Furthermore, decreasing the lower/upper hexapod spacing, $u_3$, improved the natural frequency. Because there were no reinforcement members, the natural frequency was dominated by the length of the long span members. Increasing $u_3$ forced the upper long span members to be longer with lower natural frequency. Mode 1 shape was dominated by the upper long span members, so it is clear why decreasing $u_3$ caused higher natural frequency results (see Figure 7.4). Increasing the blockage from 1% to 2% caused an increase in Member 1 diameter and, as expected, the resultant natural frequency response was improved.

### 7.3.2 Reinforced Model

Figure 7.8 is a sample of the overall data plot for 2% blockage. As seen in the figure, the member diameters and thicknesses were changed systematically. The labels in Figures 7.8-refimg:d2sectional2 illustrate how the natural frequency changes with respect to $m_{2d}$, $m_{3d}$, and $u_3$ (given a fixed $m_{2f}$ and $m_{3f}$ percentage value).
Figure 7.7: This plot shows two cases: 1% and 2% blockage with a fixed $u_3 = 1m$. This is the same data from Figures 7.5 and 7.6, however, the $u_3 = 1m$ cases are isolated from the rest of the results. There are discontinuities in the 1% blockage case due to the fact that some of the diameter and thickness combinations caused errors in ABAQUS. This error was only when the thickness was larger than the diameter of the member.

Figure 7.8: This plot shows a portion of the simulation results and the corresponding natural frequency behavior for various member thickness and diameters (2% Blockage). Each series corresponds to a unique $u_3$ value as indicated in the legend. The arrows indicated the member diameter and thickness changes. $m_{2t}$ and $m_{3t}$ were held at specific values (25% of $m_{1t}$ for this case).

Cross comparing the figures shows how the data varies with $m_{2t}$ and $m_{3t}$ percentage values. Within each of the sectional plots, increasing $m_{2d}$ improved the natural frequency where as increasing $m_{3d}$ tended to
Figure 7.9: $m_{2t}/m_{1t} = 0.25$ and $m_{3t}/m_{1t} = 0.45$. See Figure 7.8 for a detailed description of the plot.

Figure 7.10: $m_{2t}/m_{1t} = 0.25$ and $m_{3t}/m_{1t} = 0.65$. See Figure 7.8 for a detailed description of the plot.

Figure 7.11: $m_{2t}/m_{1t} = 0.45$ and $m_{3t}/m_{1t} = 0.25$. See Figure 7.8 for a detailed description of the plot.
Figure 7.12: \( m_{2t}/m_{1t} = 0.45 \) and \( m_{3t}/m_{1t} = 0.45 \). See Figure 7.8 for a detailed description of the plot.

Figure 7.13: \( m_{2t}/m_{1t} = 0.45 \) and \( m_{3t}/m_{1t} = 0.65 \). See Figure 7.8 for a detailed description of the plot.

Figure 7.14: \( m_{2t}/m_{1t} = 0.65 \) and \( m_{3t}/m_{1t} = 0.25 \). See Figure 7.8 for a detailed description of the plot.
Figure 7.15: $m_{2t}/m_{1t} = 0.65$ and $m_{3t}/m_{1t} = 0.45$. See Figure 7.8 for a detailed description of the plot.

Figure 7.16: $m_{2t}/m_{1t} = 0.65$ and $m_{3t}/m_{1t} = 0.65$. See Figure 7.8 for a detailed description of the plot.

decrease the natural frequency. Furthermore, there was an optimal value for $u_3$, which, in most cases, was $u_3 = 1 \, \text{m}$. Changing the $u_3$ value also affects the orientation of the triangular reinforcement members and therefore there is a coupled relationship between the two parameters. The optimal value for $u_3$ was most likely dependent on this orientation angle. Cross comparing the plots shows that changes in $m_{2t}$ and $m_{3t}$ have little effect on the natural frequency trend. In general decreasing the thicknesses increase the natural frequency, but only up to a certain point.

Figures 7.17-7.20 show the same results from the sectional plots, but the information was displayed using a color-bar scatter plot. Figure 7.21 is a truncated Excel Spreadsheet that shows how the Member 2 and 3 thicknesses and diameters were based off a percentage Member 1.
Figure 7.17: MATLAB three-dimensional scatter plot with a colorbar representing frequency. \( u_3 = 1 \) m and \( m_{2t}/m_{1t} = 0.25 \).

Figure 7.18: MATLAB three-dimensional scatter plot with a colorbar representing frequency. \( u_3 = 1 \) m and \( m_{2t}/m_{1t} = 0.45 \).
Figure 7.19: MATLAB three-dimensional scatter plot with a colorbar representing frequency. $u_3 = 1$ m and $m_{2t}/m_{1t} = 0.65$.

Figure 7.20: MATLAB three-dimensional scatter plot with a colorbar representing frequency. $u_3 = 1$ m and $m_{2t}/m_{1t} = 0.95$. 

90
![Figure 7.21: Member thickness and diameter variations displayed in a truncated excel worksheet.](image-url)
Figure 7.22: Reinforced model mode shapes. Higher natural frequencies tend to have a more distributed mode shape (right), where as lower natural frequencies tend to have a localized mode shape on the long span member (left).
Chapter 8

Conclusions

The primary motivation for the project was centered around the development of software tools that quickly and efficiently explore the natural frequency behavior of hexapod structures. There are numerous applications for hexapod support structures and in many cases the stiffness, weight, profile area, and natural frequency are important factors in the design process.

A Cassegrain telescope requires a support structure to hold the secondary mirror in place. The most common secondary support truss is a quadrupod, but many new telescopes designs use a hexapod to fully, but not over, constrain the secondary. The tools described in this thesis were developed to optimize the geometry of hexapod secondary support structures.

In most mechanical design problems, engineers use CAD and FEA software packages for modeling and analyzing prototype structures. While this method is cost effective and beneficial, it is difficult to explore a large design space. What is missing is a means of quickly optimizing the geometry of a structure. Such an optimization is the subject of this thesis.

The project was comprised of three parts:

1. Developed a MATLAB code that was capable of efficiently generating ABAQUS input files. The ABAQUS input files contained the information that described hexapod models similar to Figure 1.1 and Figure 1.2 for Finite Element Analysis.

2. Constructed a physical model of a simple hexapod design similar to Figure 1.1 and conduct experimental
vibration analysis. The purpose behind the physical model was to validate software predictions with experimental testing.

3. Compiled a database that defined the effects on natural frequency from basic geometric, material, and mass variations. Investigated a design scenario where many simulations were run from the MATLAB and ABAQUS optimization software.

8.1 Software Development

Developing the optimization software was initially challenging and time consuming, but the final version is modular, modifiable, and expandable. The modular characteristic (multiple functions rather than one large file) allows the user to add more features onto a basic hexapod design. The code allows for rapid design space exploration of hexapod truss models, but it can be expanded and generalized for other types of structures.

8.2 Experimental Validation

To validate the accuracy of the MATLAB-interfaced-ABAQUS program a scaled model of the base hexapod structure was built (see Figure 1.1). Two modal analysis methods, impact hammer and shake table, were performed on the model and the results agreed with the simulated predictions from ABAQUS. The modal hammer technique was less consistent than the shake table. A SolidWorks model of the structure was used to cross check software packages. The results were identical to the ABAQUS simulation.

8.3 Databases

The project produced two databases describing the natural frequency performance of hexapod structures. Database 1 was a study of how the natural frequency of general hexapod structures (base and reinforced models) varied with geometric and material variations. Database 2 was an optimization study for the base and reinforced hexapod models. This database showed how the utilization of the MATLAB-interfaced-ABAQUS program could systematically explore the design space of hexapod structures and find optimal geometry
configurations with high natural frequencies.

8.4 Overall

The project demonstrated that software tools driving a commercial FEA package can be used to explore a large design space. The software tools allow the user to quickly generate many hexapod structures and natural frequencies are obtained from the commercial FEA package. The approach can be used for any structures, not just hexapods. The software tools created in this project could be expanded into a comprehensive package covering a variety of common structures. At the very least, one could easily expand the code to explore more complicated hexapod structures e.g. including the tower bases in Figure 1.5.
Appendices
Appendix A

MATLAB Code

A.1 Main Executable

```matlab
%% File: calling.m
%
% Description: This file is the main calling program for generating a
% specified amount of input file for ABAQUS. Calling.m takes data from a
% text file for a specified amount of cases. In addition calling.m also
% creates a batch file for multi job submission in ABAQUS. Executes the
% generated jobs in the ABAQUS command module.
%
% 01-01-11 Michael E. Feeney Original code.
% 01-18-11 Michael E. Feeney Updated for text file input.
% 01-23-11 Michael E. Feeney Triangular supports and seed
% 02-07-11 Michael E. Feeney Updated for point masses
% 02-08-11 Michael E. Feeney Updated for multi mx_d mx_t

% Initial Input Variables:
% inputs.txt - text file with specified parameters for each case

clear screen variables and directories
clc;
close all;
clear all;
if ~exist('Abaqus Jobs','dir')
    rmdir('Abaqus Jobs','s');
end
	%read-in text file for input data
[d,u1,u2,u3,m1t,m1d,m2t,m2d,m3t,m3d,p,E,v,nt,tri,cross]...
textread('inputs.txt');

% Seed Size for Mesh
```
t1 = 0:.05:1;
t2 = 0:.2:1;

% Loop length based on the number of input cases from inputs.txt
counter = length(u1);

% Main Loop
for r=1:counter
    % Reciprocating clear of variables and initialization
    clear tri_nodes cross_nodes main_nodes main_elements epos tri_nodes...
    tri_elements remember1 remember2
    tri_nodes = [];
    tri_elements = [];
    cross_nodes = [];
    cross_elements = [];

    % Main nodes and elements generation and point mass locations
    [main_nodes,lt_p1,lt_p2,lt_p3] = main_node_gen(d(r),u1(r),u2(r),...
        u3(r),t1,t2);
    main_elements = main_element_gen(main_nodes,t1,t2);

    % Triangular reinforcement nodes and elements generation
    if tri(r) == 1
        e_pos = length(main_elements(:,1));
        tri_nodes = tri_node_gen(main_nodes,t1,t2,nt(r));
        tri_elements = tri_element_gen(tri_nodes,t2,nt(r),e_pos);
    end

    % Cross reinforcement nodes and elements generation
    if cross(r) == 1 && tri(r) == 1
        n_pos = length(main_nodes(:,1))+length(tri_nodes(:,1));
        cross_nodes = cross_node_gen(main_nodes,n_pos,t1,t2);
        e_pos = tri_elements(end,1);
        cross_elements = cross_element_gen(cross_nodes,t2,e_pos);
    end
    if cross(r) == 1 && tri(r) == 0
        n_pos = length(main_nodes(:,1));
        cross_nodes = cross_node_gen(main_nodes,n_pos,t1,t2);
        e_pos = length(main_elements(:,1));
        cross_elements = cross_element_gen(cross_nodes,t2,e_pos);
    end

    % Vertical nodes and elements generation
    if tri(r) == 1 && cross(r) == 1
        n_pos = length(main_nodes(:,1))+length(tri_nodes(:,1))...
            +length(cross_nodes(:,1));
        vert_nodes = vert_node_gen(main_nodes,n_pos,t1,t2);
        e_pos = cross_elements(end,1);
        vert_elements = vert_element_gen(vert_nodes,t2,e_pos);
    end
    if tri(r) == 0 && cross(r) == 1
        n_pos = length(main_nodes(:,1))+length(cross_nodes(:,1));
        vert_nodes = vert_node_gen(main_nodes,n_pos,t1,t2);
        e_pos = cross_elements(end,1);
        vert_elements = vert_element_gen(vert_nodes,t2,e_pos);
    end
    if tri(r) == 0 && cross(r) == 0
        n_pos = length(main_nodes(:,1));
        vert_nodes = vert_node_gen(main_nodes,n_pos,t1,t2);
        e_pos = length(main_elements(:,1));
        vert_elements = vert_element_gen(vert_nodes,t2,e_pos);
    end
    if tri(r) == 1 && cross(r) == 0
        n_pos = length(main_nodes(:,1))+length(tri_nodes(:,1));
        vert_nodes = vert_node_gen(main_nodes,n_pos,t1,t2);
        e_pos = length(main_elements(:,1));
        vert_elements = vert_element_gen(vert_nodes,t2,e_pos);
    end
end
e_pos = tri_elements(end,1);
vert_elements = vert_element_gen(vert_nodes,t2,e_pos);

%Compilation of all horizontal nodes and deletion of repetitive horizontal nodes
Nodesh = [main_nodes; tri_nodes; cross_nodes];
k = 0;
for i = 1:length(Nodesh(:,1))
    for j = i+1:length(Nodesh(:,1))
        if Nodesh(i,1) == Nodesh(j,1)
            k = k+1;
            remember1(k,1) = j;
        end
    end
end
Nodesh(remember1,:) = [];

%Compilation of all vertical nodes and deletion of repetitive vertical nodes
Nodesv = vert_nodes;
k = 0;
for i = 1:length(Nodesv(:,1))
    for j = 1:length(Nodesh(:,1))
        if Nodesv(i,1) == Nodesh(j,1)
            k = k+1;
            remember2(k,1) = i;
        end
    end
end
Nodesv(remember2,:) = [];

%Point mass identification
k=1;
lt_p1 = lt_p1';
lt_p2 = lt_p2';
lt_p3 = lt_p3';
for i=1:length(Nodesh(:,1))
    if abs(Nodesh(i,2:4) - lt_p1) <= 1e-10
        point_mass1(k,1) = Nodesh(i,1);
        k+1;
    end
end
if abs(Nodesh(i,2:4) - lt_p2) <= 1e-10
    point_mass2(k,1) = Nodesh(i,1);
    k+1;
end
if abs(Nodesh(i,2:4) - lt_p3) <= 1e-10
    point_mass3(k,1) = Nodesh(i,1);
    k+1;
end
point_mass = [point_mass1, point_mass2, point_mass3];

%Creates Input File for ABAQUS
input_file(r,Nodesh,Nodesv,main_elements,tri_elements,cross_elements,...
    vert_elements,m1.t(r),m1.d(r),m2.t(r),m2.d(r),m3.t(r),m3.d(r),...
    p(r),E(r),v(r),point_mass);

%Creates Job-batch.bat file
fid = fopen('Job-batch.bat','wt');
for r=1:counter
    fprintf(fid, sprintf('%n\nabaqus job=Job-%d interactive',r));
    fprintf(fid, '%n');
end
fprintf(fid,')');
fseek(fid,0,-1);
fprintf(fid,'(');
close(fid);
%Places created input files and batch file into a new directory Abaqus Jobs
mkdir('Abaqus Jobs');
movefile('Job*','Abaqus Jobs');
%Executes the batch file into the Abaqus Command Module
system('Job-batch.bat');
%Seeks for natural frequencies and masses of each job and stores
%information in a text file
frequency_search(counter);
mass_search(counter);

A.2 Input File Generator

%% File: input_file.m
%
% Description: This function takes information from the function
% nodes_elements and generates a formatted input file for ABAQUS.
%
% |
| | |||Date|||Author|||Changes|
% 11-05-2011 Michael E. Feeney Original code.
% 01-18-2011 Michael E. Feeney Made into a function/cleaned
% 02-09-2011 Michael E. Feeney Commented and Formatted
%
% Initial Input Variables:
% r - count number
% Nodesh - Horizontal Nodes
% Nodesv - Vertical Nodes
% main_elements - main structure elements
% tri_elements - triangular member elements
% cross_elements - cross member elements
% vert_elements - vertical member elements
% m1_t - Member_1 Thickness
% m1_d - Member_1 Diameter
% m2_t - Member 2 Thickness
% m2_d - Member 2 Diameter
% m3_t - Member 3 Thickness
% m3_d - Member 3 Diameter
% p - Density
% E - Modulus of Elasticity
% v - Poisson Ratio
% point_mass - point mass locations

%% input_file.m Function
function input_file(r,Nodesh,Nodesv,main_elements,...
tri_elements,cross_elements,vert_elements,m1_t,...
m1_d,m2_t,m2_d,m3_t,m3_d,p,E,v,point_mass)
% Checking the existence of tri and cross elements
size1 = length(tri_elements);
size2 = length(cross_elements);

% Printing of the Abaqus job file name and heading
fid=fopen(sprintf('Job-%d.inp',r),'wt');
fprintf(fid,'*Heading
');
fprintf(fid,sprintf('**Job Name: Job-%d.inp
',r));
fprintf(fid,'** Generated by: Abaqus/CAE Student Edition 6.10-2
');
fprintf(fid,'** Author: Michael E. Feeney
');
fprintf(fid,'*Preprint, echo=NO, model=NO, history=NO,>Contact=NO
');

% Part Title
fprintf(fid,'*Part, name=Part-1
');

% Horizontal Nodes printed into the file
fprintf(fid,'*Node, nset=Nodesh
');
dlmwrite(sprintf('Job-%d.inp',r), Nodesh,'newline','pc', '-append');
fprintf(fid,'\n*Node, nset=Nodesv
');
dlmwrite(sprintf('Job-%d.inp',r), Nodesv,'newline','pc', '-append');
fprintf(fid,'*Element, type=B31, elset=main
');
dlmwrite(sprintf('Job-%d.inp',r), main_elements,'newline','pc', '-append');
if size1 \neq 0
    fprintf(fid,'*Element, type=B31, elset=tri
');
dlmwrite(sprintf('Job-%d.inp',r), tri_elements,'newline','pc', '-append');
end
if size2 \neq 0
    fprintf(fid,'*Element, type=B31, elset=cross
');
dlmwrite(sprintf('Job-%d.inp',r), cross_elements,'newline','pc', '-append');
end

% Beam section definitions
fprintf(fid,'\n*Beam Section, elset=main_elements,'
    'material=Material-1, poisson = 0.3,'
    'temperature=GRADIENTS, section=PIPE
');
if size1 \neq 0
    fprintf(fid,sprintf('%2.5f, %2.5f\n0.,0.,1.',m1_d/2,m1_t));
end
if size2 \neq 0
    fprintf(fid,sprintf('%2.5f, %2.5f\n0.,0.,1.',m2_d/2,m2_t));
end

fprintf(fid, 'n*Beam Section, elset=vert_elements,'...
'material=Material-1, poisson = 0.3,'...
'temperature=GRADIENTS, section=PIPE\n');
fprintf(fid, sprintf('%2.5f, %2.5f\n0.0,0.0,m1_d/2,m1_t'));
fprintf(fid, 'n*End Part\n');

%Assembly title
fprintf(fid, 'n*Assembly, name=Assembly\n');
fprintf(fid, 'n*Instance, name=Part-1-1, part=Part-1\n');
fprintf(fid, 'n*End Instance\n');

%Boundary condition definition
k=1;
for j=1:length(Nodesh(:,1))
    if Nodesh(j,4) == 0
        BC(1,k) = Nodesh(j,1);
        k=k+1;
    end
end
fprintf(fid, 'n*Nset, nset=boundary,instance=Part-1-1\n');
dlmwrite(sprintf('Job-%d.inp',r), BC,'newline','pc', '-append');
fseek(fid,0,1);

%Point mass definition
fprintf(fid, 'n*Element, type=mass, elset=point_mass\n');
fprintf(fid, sprintf('1, Part-1-1.%d\n',point_mass(1)));
fprintf(fid, sprintf('2, Part-1-1.%d\n',point_mass(2)));
fprintf(fid, sprintf('3, Part-1-1.%d\n',point_mass(3)));
fseek(fid,0,1);

%fseek(fid,0,1);
fprintf(fid, 'n*Mass, elset=point_mass\n');
fprintf(fid, '333.333,\n');
fprintf(fid, 'n*End Assembly\n\n');

%Material definition
fprintf(fid, 'n*Material, name=Material-1\n');
fprintf(fid, 'n*Density\n');
fprintf(fid, sprintf('%f,\n',p));
fprintf(fid, 'n*Elastic\n');
fprintf(fid, sprintf('%f, %f\n',E,v));
fprintf(fid, 'n*Boundary\n');
fprintf(fid, 'n*End Boundary\n');

%Step definition
fprintf(fid, 'n*Step, name=Step-1, perturbation\n');
fprintf(fid, 'n*Frequency, eigensolver=Lanczos,'...
'acoustic coupling=on, normalization=displacement\n');
fprintf(fid, 'n*Restart, write, frequency=1\n');
fprintf(fid, 'n*Output, field, variable=PRESELECT\n');
fclose(fid);

A.3 Node Generation

%% File: node_gen.m
A.4 Main Node Generation

```matlab
function [Nodes] = node_gen(p1,p2,t,m)

slope = p2-p1;
if m == 1
    k = length(t);
    for i=1:k
        Nodes(i,1) = p1(1)+slope(1)*t(i);
        Nodes(i,2) = p1(2)+slope(2)*t(i);
        Nodes(i,3) = p1(3)+slope(3)*t(i);
    end
end
if m == 0
    t = t(2:end);
    k = length(t);
    for i=1:k
        Nodes(i,1) = p1(1)+slope(1)*t(i);
        Nodes(i,2) = p1(2)+slope(2)*t(i);
        Nodes(i,3) = p1(3)+slope(3)*t(i);
    end
end
end
```
% t2 - seed size of the cross, tri, and vert members

%% Function cross_node_gen.m

function [main_nodes, lt_point_1, lt_point_2, lt_point_3] = main_node_gen(d,u_1,u_2,u_3,t1,t2)

% Rotation Tensor
th1 = 2*pi/3;
th2 = 4*pi/3;
R1 = [cos(th1) -sin(th1) 0; sin(th1) cos(th1) 0; 0 0 1]';
R2 = [cos(th2) -sin(th2) 0; sin(th2) cos(th2) 0; 0 0 1]';

% Centroid coordinate calculation

% Upper triangle coordinates
ut_point_1 = [0, c2, u_2+u_3]';

% Lower triangle coordinates
lt_point_1 = [0, c2, u_2]';

% Span coordinates
s_point_1 = [0, -d, 0]';

% Upper node generation
UpperNodes1 = node_gen(ut_point_1,ut_point_2,t2,1);
UpperNodes2 = node_gen(ut_point_2,ut_point_3,t2,0);
UpperNodes3 = node_gen(ut_point_3,ut_point_1,t2,0);
UpperNodes4 = node_gen(s_point_2,ut_point_3,t1,0);
UpperNodes5 = node_gen(ut_point_3,s_point_1,t1,0);
UpperNodes6 = node_gen(s_point_1,ut_point_2,t1,0);
UpperNodes7 = node_gen(s_point_1,ut_point_3,t1,0);
UpperNodes8 = node_gen(ut_point_1,s_point_3,t1,0);
UpperNodes9 = node_gen(s_point_3,ut_point_1,t1,0);
UpperNodes = [UpperNodes1;UpperNodes2;UpperNodes3;UpperNodes4;...;
UpperNodes5;UpperNodes6;UpperNodes7;UpperNodes8;UpperNodes9];

% Lower node generation
LowerNodes1 = node_gen(lt_point_1,lt_point_2,t2,1);
LowerNodes2 = node_gen(lt_point_2,lt_point_3,t2,0);
LowerNodes3 = node_gen(lt_point_3,lt_point_1,t2,0);
LowerNodes4 = node_gen(s_point_2,lt_point_3,t1,0);
LowerNodes5 = node_gen(ut_point_3,s_point_1,t1,0);
LowerNodes6 = node_gen(s_point_1,lt_point_2,t1,0);
LowerNodes7 = node_gen(s_point_1,lt_point_3,t1,0);
LowerNodes8 = node_gen(ut_point_2,s_point_3,t1,0);
LowerNodes9 = node_gen(s_point_3,lt_point_1,t1,0);
LowerNodes = [LowerNodes1;LowerNodes2;LowerNodes3;LowerNodes4;...;
LowerNodes5;LowerNodes6;LowerNodes7;LowerNodes8;LowerNodes9];

% Main node compilation and numbering
main_nodes1(:,2:4) = [UpperNodes;LowerNodes];
k = length(main_nodes1(:,1));
for i = 1:k
A.5 Main Element Generation

%% File: main_element_gen.m

% Description: This function generates the elements for the main reinforcement members.

% Initial Input Variables:
% main_nodes - nodes of the main members
% t1 - seed size of the main members
% t2 - seed size of the cross, tri, and vert members

%% Function main_element_gen.m

function [main_elements] = main_element_gen(main_nodes,t1,t2)

% Initial matrix definition and length
main_elements = []; % L1 = length(main_nodes(:,1))/2-1;

% Connects the nodes for the upper section into the main element matrix
for i = 1:1:L1
    main_elements(i,2) = main_nodes(i,1);
    main_elements(i,3) = main_nodes(i+1,1);
end

p = i+2;

% Connects the nodes for the lower section into the main element matrix
for i = p:1:L1+2+1
    main_elements(i-1,2) = main_nodes(i,1);
    main_elements(i-1,3) = main_nodes(i+1,1);
end

% Numbers the elements in the main element matrix
L2 = length(main_elements(:,2));
for i = 1:1:L2
    main_elements(i,1) = i;
end
A.6 Triangular Reinforcement Node Generation

```matlab
%% File: tri_node_gen.m

% Description: This function generates the nodes for the triangular reinforcement members.

% 01-14-2011 Michael E. Feeney Original Code
% 02-09-2011 Michael E. Feeney Commented and Formated

%% Initial Input Variables:
main_nodes - nodes of the main structure
t1 - seed size of the main structure members
t2 - seed size of the cross, tri, and vert members
nt - number of reinforcement triangles

function [tri_nodes] = tri_node_gen(main_nodes,t1,t2,nt)

%Initial calculations for spacing in the tri_node matrix
len1 = length(t1);
len2 = length(t2);
shift1 = floor(len1/nt);
shift2 = shift1*2;

%Coordinate definitions for initial triangles
p1 = 1;
p2 = (len2*3-2)*2+(len1*6-6)+shift1;
p3 = (len2*3-2)+shift2;
p4 = len2*2-1;
p5 = (len2*3-2)*2+(len1*8-8)-shift1;
p6 = (len2*3-2)+(len1*2-2)-shift2;
p7 = len2*2-1;
p8 = (len2*3-2)*2+(len1*8-8)+shift1;
p9 = (len2*3-2)+(len1*2-2)+shift2;
p10 = len2;
p11 = (len2*3-2)*2+(len1*10-10)-shift1;
p12 = (len2*3-2)+(len1*4-4)-shift2;
p13 = len2;
p14 = (len2*3-2)*2+(len1*10-10)+shift1;
p15 = (len2*3-2)+(len1*4-4)+shift2;
p16 = 1;
p17 = (len2*3-2)*2+(len1*12-12)-shift1;
p18 = (len2*3-2)+(len1*6-6)-shift2;

%Compilation of the initial triangle coordinates for nt = 5
if nt == 5
```

106
Nodes\textsubscript{t1} = \{p1 p2; p2 p3; p3 p2+shift2; p2+shift2 p3+shift2\};
Nodes\textsubscript{t2} = \{p4 p5; p5 p6; p6 p5-shift2; p5-shift2 p6-shift2\};
Nodes\textsubscript{t3} = \{p7 p8; p8 p9; p9 p8+shift2; p8+shift2 p9+shift2\};
Nodes\textsubscript{t4} = \{p10 p11; p11 p12; p12 p11-shift2; p11-shift2 p12-shift2\};
Nodes\textsubscript{t5} = \{p13 p14; p14 p15; p15 p14+shift2; p14+shift2 p15+shift2\};
Nodes\textsubscript{t6} = \{p16 p17; p17 p18; p18 p17-shift2; p17-shift2 p18-shift2\};

end

%Compilation of the initial triangle coordinates for nt = 10
if nt == 10
    
    Nodes\textsubscript{t1} = \{p1 p2; p2 p3; p3 p2+shift2; p2+shift2 p3+shift2;...
p3+shift2 p2+shift2*2; p2+shift2*2 p3+shift2*2;...
p2+shift2*3; p2+shift2*3 p3+shift2*3;...
p3+shift2*4; p3+shift2*4 p2+shift2*4;...
\}

    Nodes\textsubscript{t2} = \{p4 p5; p5 p6; p6 p5-shift2; p5-shift2 p6-shift2;...
p6-shift2 p5-shift2*2; p5-shift2*2 p6-shift2*2;...
p5-shift2*3; p5-shift2*3 p6-shift2*3;...
p6-shift2*4; p6-shift2*4 p5-shift2*4;...
\}

    Nodes\textsubscript{t3} = \{p7 p8; p8 p9; p9 p8+shift2; p8+shift2 p9+shift2;...
p8+shift2 p8+shift2*2; p8+shift2*2 p9+shift2*2;...
p9+shift2*3; p9+shift2*3 p8+shift2*3;...
p8+shift2*4; p8+shift2*4 p9+shift2*4;...
\}

    Nodes\textsubscript{t4} = \{p10 p11; p11 p12; p12 p11-shift2; p11-shift2 p12-shift2;...
p12-shift2 p12-shift2*2; p12-shift2*2 p11-shift2*2;...
p11-shift2*3; p11-shift2*3 p12-shift2*3;...
p12-shift2*4; p12-shift2*4 p11-shift2*4;...
\}

    Nodes\textsubscript{t5} = \{p13 p14; p14 p15; p15 p14+shift2; p14+shift2 p15+shift2;...
p15+shift2 p15+shift2*2; p15+shift2*2 p14+shift2*2;...
p14+shift2*3; p14+shift2*3 p15+shift2*3;...
p15+shift2*4; p15+shift2*4 p14+shift2*4;...
\}

    Nodes\textsubscript{t6} = \{p16 p17; p17 p18; p18 p17-shift2; p17-shift2 p18-shift2;...
p18-shift2 p18-shift2*2; p18-shift2*2 p17-shift2*2;...
p17-shift2*3; p17-shift2*3 p18-shift2*3;...
p18-shift2*4; p18-shift2*4 p17-shift2*4;...
\}

end

%Node generation for the first line of connecting nodes
len3 = length(Nodes\textsubscript{t1}(1,1));  
Nodes\textsubscript{T1} = [];  
Nodes\textsubscript{T2} = [];  
Nodes\textsubscript{T3} = [];  
Nodes\textsubscript{T4} = [];  
Nodes\textsubscript{T5} = [];  
Nodes\textsubscript{T6} = [];  
a = node\_gen(main\_nodes(Nodes\textsubscript{t1}(1,1),2:4),...  
    main\_nodes(Nodes\textsubscript{t1}(1,2),2:4),t2,1);  
b = node\_gen(main\_nodes(Nodes\textsubscript{t2}(1,1),2:4),...  
    main\_nodes(Nodes\textsubscript{t2}(1,2),2:4),t2,1);  
c = node\_gen(main\_nodes(Nodes\textsubscript{t3}(1,1),2:4),...  
    main\_nodes(Nodes\textsubscript{t3}(1,2),2:4),t2,1);  
d = node\_gen(main\_nodes(Nodes\textsubscript{t4}(1,1),2:4),...  
    main\_nodes(Nodes\textsubscript{t4}(1,2),2:4),t2,1);  
e = node\_gen(main\_nodes(Nodes\textsubscript{t5}(1,1),2:4),...  
    main\_nodes(Nodes\textsubscript{t5}(1,2),2:4),t2,1);  
f = node\_gen(main\_nodes(Nodes\textsubscript{t6}(1,1),2:4),...  
    main\_nodes(Nodes\textsubscript{t6}(1,2),2:4),t2,1);  
Nodes\textsubscript{T1} = [Nodes\textsubscript{T1}; a];  
Nodes\textsubscript{T2} = [Nodes\textsubscript{T2}; b];  
Nodes\textsubscript{T3} = [Nodes\textsubscript{T3}; c];  
Nodes\textsubscript{T4} = [Nodes\textsubscript{T4}; d];  
Nodes\textsubscript{T5} = [Nodes\textsubscript{T5}; e];  
Nodes\textsubscript{T6} = [Nodes\textsubscript{T6}; f];
clear a b c d e f;
for i = 2:1:len3
    a = node_gen(main_nodes(Nodes_t1(i,1),2:4),...
                 main_nodes(Nodes_t1(i,2),2:4),t2,0);
    b = node_gen(main_nodes(Nodes_t2(i,1),2:4),...
                 main_nodes(Nodes_t2(i,2),2:4),t2,0);
    c = node_gen(main_nodes(Nodes_t3(i,1),2:4),...
                 main_nodes(Nodes_t3(i,2),2:4),t2,0);
    d = node_gen(main_nodes(Nodes_t4(i,1),2:4),...
                 main_nodes(Nodes_t4(i,2),2:4),t2,0);
    e = node_gen(main_nodes(Nodes_t5(i,1),2:4),...
                 main_nodes(Nodes_t5(i,2),2:4),t2,0);
    f = node_gen(main_nodes(Nodes_t6(i,1),2:4),...
                 main_nodes(Nodes_t6(i,2),2:4),t2,0);
    Nodes_T1 = [Nodes_T1; a];
    Nodes_T2 = [Nodes_T2; b];
    Nodes_T3 = [Nodes_T3; c];
    Nodes_T4 = [Nodes_T4; d];
    Nodes_T5 = [Nodes_T5; e];
    Nodes_T6 = [Nodes_T6; f];
end

%Compilation of all tri nodes. Numbers the nodes making sure not
%to define a unique node number to a node that already exists in the
%main_nodes matrix
tri_nodes(:,2:4) = [Nodes_T1; Nodes_T2; Nodes_T3; Nodes_T4;...
                   Nodes_T5; Nodes_T6];
k = length(tri_nodes(:,1));
k = length(main_nodes(:,1));
for i = 1:k
    tri_nodes(i,1) = kk;
    kk = kk+1;
end
kk = length(main_nodes(:,1));
for i = 1:k
    for j = 1:k
        if abs(tri_nodes(i,2:4) - main_nodes(j,2:4)) < 1e-10
            tri_nodes(i,1) = main_nodes(j,1);
        end
    end
end

A.7 Triangular Reinforcement Element Generation

%% File: tri_element_gen.m

% Description: This function generates the elements for the cross
% reinforcement members.

% 01-14-2011 Michael E. Feeney Original Code
% 02-09-2011 Michael E. Feeney Commented and Formated
%% Initial Input Variables:
% cross_nodes - nodes of the cross members
% t2 - seed size of the cross, tri, and vert members
% epos - current element number

%% Function cross_element_gen.m
function [tri_elements] = tri_element_gen(tri_nodes,t2,nt,epos)

%Initial calculations for spacing in the tri_element matrix
len1 = length(tri_nodes(:,1));
len2 = length(t2);

%Connecting the tri_nodes into the tri_element matrix for nt = 5
if nt == 5
    len3 = 4*(len2-1);
    start = 1;
    finish = len3;
    while finish <= len1-1
        for i=start:1:finish
            tri_elements(i,2) = tri_nodes(i,1);
            tri_elements(i,3) = tri_nodes(i+1,1);
        end
        start = finish+2;
        finish = start+len3-1;
    end
end

%Connecting the tri_nodes into the tri_element matrix for nt = 10
if nt == 10
    len3 = 8*(len2-1);
    start = 1;
    finish = len3;
    while finish <= len1-1
        for i=start:1:finish
            tri_elements(i,2) = tri_nodes(i,1);
            tri_elements(i,3) = tri_nodes(i+1,1);
        end
        start = finish+2;
        finish = start+len3-1;
    end
end

%Removes skipped lines from tri_element matrix generation
k=0;
for i=1:length(tri_elements)
    if tri_elements(i,2:3)==0
        k=k+1;
        remove(k,1) = i;
    end
end

%Numbers the elements in the tri_element matrix
tri_elements(remove,:) = [];
for i=1:length(tri_elements)
    tri_elements(i,1) = epos+1;
    epos = epos+1;
end
A.8 Cross Reinforcement Node Generation

```matlab
%% File: cross_node_gen.m
% Description: This function generates the nodes for the cross reinforcement members.

% Initial Input Variables:
% main_nodes - nodes of the main structure
% npos - current node number
% t1 - seed size of the main structure members
% t2 - seed size of the cross, tri, and vert members

%% Function cross_node_gen.m
function cross_nodes = cross_node_gen(main_nodes, npos, t1, t2)

% Initial calculations for spacing in the cross_nodes matrix
len1 = length(t1);
len2 = length(t2);

% Coordinates for the first leg
p1 = 1;
p2 = (len2*3-2)*2+(len1*8-8)-floor(len1*.25);
p3 = (len2*3-2)+floor(len1*.5);
p4 = (len2*3-2)*2+(len1*8-8)-floor(len1*.75);

% Coordinates for the second leg
p5 = len2*2-1;
p6 = (len2*3-2)*2+(len1*10-10)-floor(len1*.25);
p7 = (len2*3-2)+(len1*2-2)+floor(len1*.5);
p8 = (len2*3-2)*2+(len1*10-10)-floor(len1*.75);

% Coordinates for the third leg
p9 = len2;
p10 = (len2*3-2)*2+(len1*12-12)-floor(len1*.25);
p11 = (len2*3-2)+(len1*4-4)+floor(len1*.5);
p12 = (len2*3-2)*2+(len1*12-12)-floor(len1*.75);

% Node completion for the cross members of each leg 1, 2, 3
Nodes_c1 = [p1 p2; p2 p3; p3 p4];
Nodes_c2 = [p5 p6; p6 p7; p7 p8];
Nodes_c3 = [p9 p10; p10 p11; p11 p12];

% Node generation of the first line of connecting nodes
len3 = length(Nodes_c1(:,1));
Nodes_C1 = [Nodes_c1];
Nodes_C2 = [Nodes_c2];
Nodes_C3 = [Nodes_c3];
a = node_gen(main_nodes(Nodes_c1{1,1},2:4),
             main_nodes(Nodes_c1{1,2},2:4),t2,1);
b = node_gen(main_nodes(Nodes_c2{1,1},2:4),
             main_nodes(Nodes_c2{1,2},2:4),t2,1);
c = node_gen(main_nodes(Nodes_c3{1,1},2:4),
             main_nodes(Nodes_c3{1,2},2:4),t2,1);
Nodes_C1 = [Nodes_C1; a];
```

110
A.9 Cross Reinforcement Element Generation
%function [cross_elements] = cross_element_gen(cross_nodes,t2,e_pos)

%Initial calculations for spacing in the cross_element matrix
len1 = length(cross_nodes(:,1));
len2 = length(t2);
len3 = 3*(len2-1);
start = 1;
finish = len3;

%Connecting the cross_nodes into the cross_element matrix
while finish<len1-1
    for i=start:1:finish
        cross_elements(i,2) = cross_nodes(i,1);
cross_elements(i,3) = cross_nodes(i+1,1);
    end
    start = finish+2;
    finish = start+len3-1;
end

%Removes skipped lines from cross_element matrix generation
k=0;
for i=1:1:length(cross_elements(:,1))
    if cross_elements(i,2:3)==0
        k=k+1;
        remove(k,1) = i;
    end
end
cross_elements(remove,:) =[];

%Numbers the elements in the cross_element matrix
for i=1:1:length(cross_elements)
cross_elements(i,1) = e_pos+1;
    e_pos = e_pos+1;
end

A.10 Vertical Node Generation

%% File: vert_node_gen.m
%% Description: This function generates the nodes for the vertical
%% reinforcement members.

%% Initial Input Variables:
% main_nodes - nodes of the main structure
% n_pos - current node number
% t1 - seed size of the main structure members
% t2 - seed size of the cross, tri, and vert members

%% Function vert_node_gen.m
function [vert_nodes] = vert_node_gen(main_nodes,n_pos,t1,t2)

%Initial calculations for spacing in the vert_node matrix
len1 = length(t1);
len2 = length(t2);

%Coordinate definitions for upper and lower vertices of the triangles
p1 = 1;
p2 = (len2*3-2)+(len1*6-5);
p3 = len2;
p4 = p2 + len2 -1;
p5 = (len2*2-1);
p6 = p4 + len2 -1;

%Node generation for the three vertical members
Nodes_V1 = node_gen(main_nodes(p1,2:4),main_nodes(p2,2:4),t2,1);
Nodes_V2 = node_gen(main_nodes(p3,2:4),main_nodes(p4,2:4),t2,1);
Nodes_V3 = node_gen(main_nodes(p5,2:4),main_nodes(p6,2:4),t2,1);
vert_nodes(:,2:4) = [Nodes_V1; Nodes_V2; Nodes_V3];

%Compilation of all vert nodes. Numbers the nodes making sure not
%to define a unique node number to a node that already exists in the
%main_nodes matrix
len3 = length(vert_nodes(:,1));
for i = 1:1:len3
    vert_nodes(i,1) = n_pos +1;
n_pos=n_pos+1;
end
kk = length(main_nodes(:,1));
for i = 1:1:len3
    for j = 1:1:kk
        if abs((vert_nodes(i,2:4) - main_nodes(j,2:4))) < 1e-10
            vert_nodes(i,1) = main_nodes(j,1);
        end
    end
end
end

A.11 Vertical Element Generation

%% File: vert_element_gen.m

% Description: This function generates the elements for the vertical
% reinforcement members.

% Initial Input Variables:
% vert_nodes - nodes of the vertical members
% t2 - seed size of the cross, tri, and vert members
% epos - current element number

% 01-14-2011 Michael E. Feeney Original Code
% 02-09-2011 Michael E. Feeney Commented and Formated
A.12 Mass Search Tool
%Prints the Job Number and searches for location of dat file where the
%masses are listed.
for i = 1:counter
    fid = fopen(sprintf('Job-%d.dat',i));
    string1 = ' TOTAL MASS OF MODEL';
    string2 = '** EXECUTION IS TERMINATED **';
    while strcmp(tline,string1)\neq 1;
        tline = fgetl(fid);
        if strcmp(tline,string2)==1
            break
        end
    end
    if strcmp(tline,string2)\neq 1
        tline = fgetl(fid);
        mass = fgetl(fid);
        fclose(fid);
        fid = fopen('Masses.txt','a');
        fprintf(fid,mass,'n');
        fprintf(fid,\n);
        fclose(fid);
    end
end

%Reads in generated data and plots natural frequencies vs. job number
% [a] = textread('Masses.txt');
% figure(2)
% plot(1:1:counter,a);
% cd ..
end

A.13 Frequency Search Tool
%% Function frequency_search.m

function frequency_search(counter)

%Changes Directory to Aabqus Jobs
cd 'Abaqus Jobs';

%Prints the Job Number and searches for location of dat file where the
%natural frequencies are listed.
for i = 1:counter
    fid = fopen(sprintf('Job-%d.dat',i));
    tline = fgetl(fid);
    string1 = ' (RAD/TIME) (CYCLES/TIME)';
    string2 = ' ** EXECUTION IS TERMINATED **';
    while strcmp(tline,string1)\n        tline = fgetl(fid);
        if strcmp(tline,string2)==1
            break\n        end\n    end
    if strcmp(tline,string2)\n        tline = fgetl(fid);
        tline = fgetl(fid);
        if strcmp(tline,string2)==1
            fclose(fid);
            fid = fopen('Frequencies.txt','a');
            fprintf(fid,Frequency1,'\n');
            fprintf(fid,'\n');
            fclose(fid);
        else
            fclose(fid);
        end
    end
end

%Reads in generated data and plots natural frequencies vs. job number
% [a] = textread('Frequencies.txt');
% figure(1)
% plot(1:1:counter,a(:,4));
cd ..
end

A.14 Diameter and Thickness Calculator

function [M1] = input_gen(u_1,u_2,u_3,L12,L13,d,B,M,p)
clc;
var1 = [.25 .25; .25 .45; .25 .65; .25 .95; .45 .25; .45 .45; .45 .65;]
\[c = u_1 \times \sqrt{3}/6;\]
\[L_1 = d - c;\]
\[L_2 = \sqrt{((u_1/2)^2 + L_1^2)};\]
\[\theta_1 = \arcsin(L_1/L_2);\]
\[L_3 = \sqrt{L_2^2 + (u_2 + u_3)^2};\]
\[L_4 = \sqrt{L_2^2 + u_2^2};\]
\[s_1 = \sqrt{l_1^2 + u_1^2 - 2*l_1*u_1}\cos(\theta_1);\]
\[s_2 = \sqrt{l_2^2 + (l_1 + l_2)^2 - 2*l_2*(l_1 + l_2)\cos(\theta_2)};\]
\[A = \pi \times d^2;\]
\[x = \text{length}(\text{var1}(;1));\]
\[\text{for } i=1:1:x\]
\[\quad M1(i,2) = (A*B)/(6*L2+3*u_1+3*(s1+s2+s3)*\text{var1}(i,2));\]
\[\quad M1(i,4) = \text{var1}(i,1)*M1(i,2);\]
\[\quad M1(i,6) = \text{var1}(i,2)*M1(i,2);\]
\[\text{end}\]
\[\]
\[\]
\[\text{for } i=1:1:j-1\]
\[\quad M2 = M1;\]
\[\quad M3 = [M3;M2];\]
\[\text{end}\]
\[\]
\[LL1 = 6*(u_1+L3+L4)+3*u_2;\]
\[\]
\[x = \text{length}(M1(:,1));\]
\[\text{syms } \text{m1t}\]
\[\text{M5 = [ ];}\]
\[\text{for } j=1:1:x\]
\[\quad \text{for } i=1:1:x\]
\[\quad\quad \text{eq} = (LL1*0.785*(M1(i,2)^2 - (M1(i,2)-2*m1t)^2)...\]
\[\quad\quad \text{LL2*0.785*(M1(i,4)^2 - (M1(i,4)-2*m1t*var1(j,1))^2)...}\]
\[\quad\quad \text{LL3*0.785*(M1(i,6)^2 - (M1(i,6)-2*m1t*var1(j,2))^2))*p = 300;}\]
\[\quad\quad \text{coo} = \text{eval(}\text{solve(eq));}\]
\[\quad\quad M1(i,1) = \text{min(coo)};\]
\[\quad\quad M1(i,3) = \text{var1}(j,1)*M1(i,1);\]
\[\quad\quad M1(i,5) = \text{var1}(j,2)*M1(i,1);\]
\[\quad\text{end}\]
\[\quad M4 = M1;\]
\[\text{end}\]
\[\text{M5 = [M5;M4];}\]
\[\text{end}\]
\[\]
\[\text{clear M1}\]
\[\text{M1(:,1) = M5(:,1)};\]
\[\text{M1(:,2) = M3(:,2)};\]
\[\text{M1(:,3) = M5(:,3)};\]
\[\text{M1(:,4) = M3(:,4)};\]
M1(:,5) = M5(:,5);
M1(:,6) = M3(:,6);
dlmwrite('DiameterThickness.txt', M1)
end
Bibliography


