Brigham Young University
BYU ScholarsArchive

Theses and Dissertations

2006-12-06

Large 3-D Deflection and Force Analysis of Lateral Torsional Buckled Beams

Robert Parley Chase
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

 Part of the Mechanical Engineering Commons

BYU ScholarsArchive Citation

Chase, Robert Parley, "Large 3-D Deflection and Force Analysis of Lateral Torsional Buckled Beams" (2006). *Theses and Dissertations*. 1040.
https://scholarsarchive.byu.edu/etd/1040

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

LARGE 3-D DEFLECTION AND FORCE ANALYSIS OF LATERAL TORSIONAL BUCKLED BEAMS

by

Robert P. Chase Jr.

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering
Brigham Young University
December 2006

Copyright © 2006 Robert P. Chase Jr.

All Rights Reserved

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Robert P. Chase Jr.

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

_____	_____
Date	Robert H. Todd, Chair
_____	_____
Date	Larry L. Howell
_____	_____
Date	Spencer P. Magleby

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Robert P. Chase Jr. in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

_____ _____

Date

Robert H. Todd
Chair, Graduate Committee

Accepted for the Department

_____

Matthew R. Jones
Graduate Coordinator

Accepted for the College

_____

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

ABSTRACT

LARGE 3-D DEFLECTION AND FORCE ANALYSIS OF
LATERAL TORSIONAL BUCKLED BEAMS

Robert P. Chase Jr.
Department of Mechanical Engineering
Master of Science

This thesis presents research on the force and deflection behavior of beams with rectangular cross-sections undergoing lateral torsional buckling. The large 3-D deflection path of buckling beam tips was closely approximated by circular arcs in two planes. A new chain algorithm element was created from pseudo-rigid-body segments and used in a chain calculation that accurately predicted the force deflection relationship of beams with large 3-D deflections.

ACKNOWLEDGMENTS

My thanks go out to many people who helped me with this thesis. I am grateful to David Hall, president of Novatek, who sponsored me in my research. I am also thankful for the guidance and counsel of Bob Todd, my advisor. And finally, I am deeply grateful for the loving and patient help from my beautiful editor and wife, Kathryn.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Thesis Statement

Compliant mechanisms have often been modeled in two dimensions using rectangular or radial coordinates. These models are common because they are easy to visualize and understand. However, many useful 3-D mechanisms have not been invented because their enabling design tools have not been created and their special defining properties have yet to be discovered. This thesis focuses on defining the special deflection properties of lateral torsional buckling and creating a 3-D chain algorithm with pseudo-rigid-body model elements that accurately predicts 3-D forces and large deflections of beams. The algorithm would be useful for solving many general beam problems and could predict cases of lateral torsional buckling.

A good example of lateral torsional buckling can be readily understood with a cantilever beam that has a slender, rectangular cross-section. If a force is placed at the tip of the beam so that it is roughly perpendicular to the long sides of the rectangular cross section, the beam will bend in the direction of the force, which can be easily predicted and modeled in two dimensions, as seen in Figure 1.1.

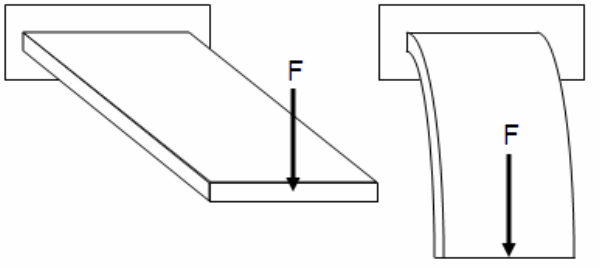


Figure 1.1: Bending beams modeled in 2-D

What happens if the force is again placed at the tip of the beam but this time it is exactly perpendicular to the short sides of the rectangular cross-section? If the force is small enough, then the beam will act similarly to the two-dimensional case. It will deflect in the direction of the force. If the force were large enough, the beam would twist a very tiny amount because of imperfections in the beam, as shown in Figure 1.2.

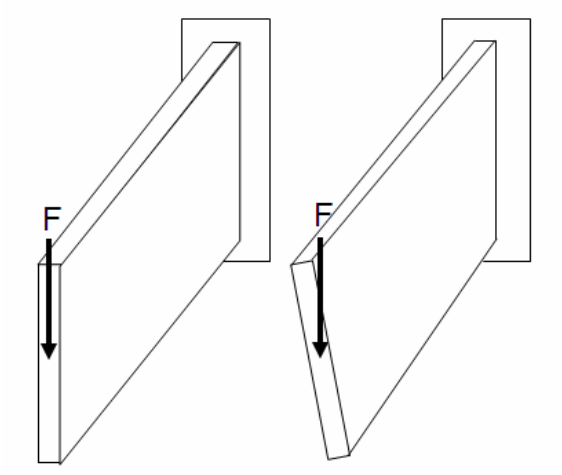


Figure 1.2: A beam deflecting in the direction of the force, resulting in twisting because of imperfections in the beam

If however, the beam had been theoretically perfect and the applied load had a tiny error, the same phenomena would be observed. This twisting in the beam would be accompanied with a very small tip deflection almost orthogonal to the applied force. In this case both the twisting and the nearly orthogonal deflection are extremely small and the beam's stiffness holds it in position. Assuming the applied load begins to increase, the beam would twist more, causing more of the applied load to be felt on the slender direction of the beam. The beam would, in turn, deflect nearly orthogonal to the load, causing an even greater torque felt through the beam. Finally, enough of the load would have been felt by the slender sides of the beam that it would no longer be stiff enough to hold its original position but would move to one side, finding a new resting position (see Figure 1.3).

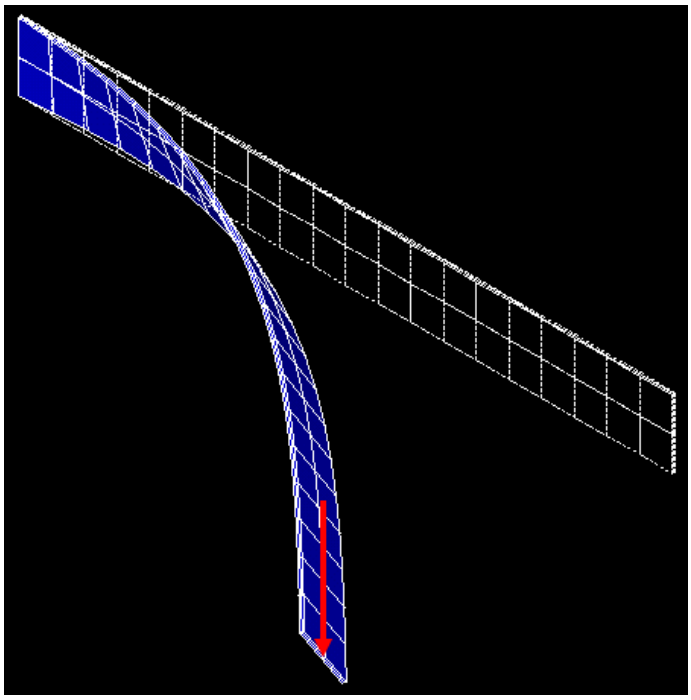


Figure 1.3: Lateral torsional buckling

This phenomenon of twisting and lateral movement is known as lateral torsional buckling. The above example describes the causes of this type of buckling. However, the amount of twisting and orthogonal deflection is both a function of location on the beam as well as the applied force. Aside from the fixed end of the cantilever beam, no part of the beam remains in its original plane. The beam has twisted and bent in different amounts along its length.

Modeling large displacements of lateral torsional buckling for future use in mechanisms may appear to be complicated. However, this thesis will present some ideas and methods that allow lateral torsional buckling to be visualized and understood with much more ease.

1.2 Thesis Objectives

Thesis objectives will include the following:

1. Summarize previous work done in the following areas:
    - Lateral torsional buckling
    - Buckling of compliant ortho-planar springs
    - A 3-D pseudo-rigid-body model

- Chain algorithms
- Method of superposition
- Combining chain algorithms with pseudo-rigid-body models

2. Characterize the following for rectangular, prismatic cantilever beams undergoing large deflections of lateral torsional buckling due to a single tip force:
    - A simplified method to understand 3-D large deflections
    - Beam tip rotations
    - The causes for variations in deflection paths and tip rotations
3. Create a 3-D Chain algorithm with the following properties:
    - Uses pseudo-rigid-body elements
    - Allows for force and moment boundary conditions along the length of the beam
    - Accurately predicts lateral torsional buckling large deflection paths
    - Accurately predicts lateral torsional buckling force-deflection relationship

1.3 Thesis Outline

Chapter 1 presents a statement of the problem, the research objectives, and the outline for each chapter.

Chapter 2 overviews what others have already done in this field of research. It explores research in compliant mechanisms, lateral torsional buckling, chain algorithms, superposition, and pseudo-rigid-body models.

Chapter 3 proposes a displacement analysis method that will find the defining properties of lateral torsional buckling and a method to make a 3-D chain algorithm with pseudo-rigid body model elements.

Chapter 4 uses a displacement analysis method to define the properties of lateral torsional buckling and presents the results and a case study.

Chapter 5 focuses on the design of a 3-D chain algorithm with pseudo-rigid-body model elements, and presents a simplified method to understand beam rotations, followed by a case study.

Chapter 6 explores the implications, advantages, and disadvantages of this research. Recommendations for further research are then noted.

Chapter 2

Previous Research

2.1 Introduction to Four Areas of Mechanics

Past research related to this research comes from four different areas of mechanics. The four areas will first be briefly described and then discussed in more detail.

The first area is the theory behind lateral torsional buckling (see Figure 2.1). Lateral torsional buckling of a beam occurs when an in-plane force causes an out-of-plane deflection with axial twisting. This type of buckling is caused by a high resistance to bending in one plane and low resistance to bending in the other, imperfections in the beam, and a large enough force pushing in a direction perpendicular to the beam's stiffer moment of inertia.

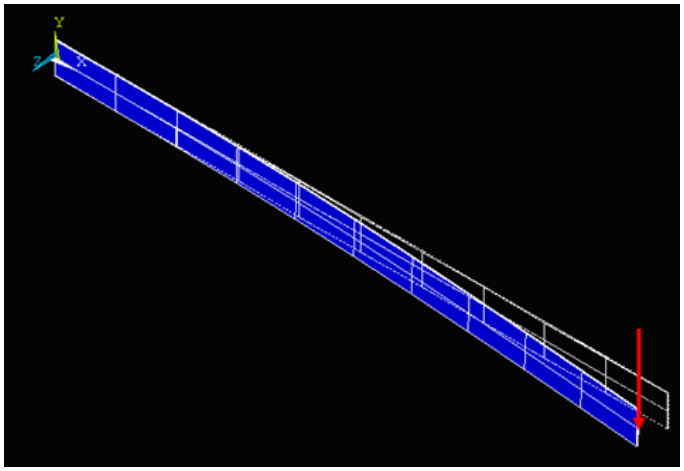


Figure 2.1: An in-plane force causes an out-of-plane deflection

The second area of mechanics that this project will address is the theory behind many compliant mechanisms: 2-D pseudo-rigid-body models. Combining lateral torsional buckling theory and 2-D pseudo-rigid-body model theory is not enough to produce an accurate 3-D pseudo-rigid-body model, but they provide a base for more knowledge to expand from.

The third area of mechanics is the chain algorithm. "The chain algorithm is so named because it requires discretization of the object being modeled into beam elements and analyzes each element in succession" [1]. The chain algorithm also depends on small deflection formulas to describe the displacements for each small beam segment. This theory allows different parts of a beam to undergo different types of displacements.

The fourth is the method of superposition. Superposition is a simple concept used to find the deflection and rotation of beams by breaking the problem down into components. It can be summed up as follows: "the deflection of a beam produced by several different loads acting simultaneously can be found by superposing the deflections produced by the same loads acting separately" [2]. This method for solving problems has been proven to be straightforward and accurate.

2.2 Lateral Torsional Buckling

Research on the stability of rods has traditionally focused on finding the critical buckling load that will cause a beam to buckle out of plane. (Rod, in this case, means a beam with "the cross-section where a rectangle of which one pair of sides is much longer than the other pair" [3].) The main goal in studying this type of buckling has been to build safe public structures such as bridges and buildings. In these cases it is best for the beam to remain rigid with minimal deflection. Lateral deflection in a building or bridge could quickly lead to catastrophic failure.

Lateral torsional buckling has been investigated using Kirchhoff's theory of rod stability with combined states of stress. In an article Love states, "Since the treatment of [lateral torsional rod stability] is very complicated, we approximate the equations by considering small deviations from the unstrained rectilinear state, and study the stability of the solutions by applying the linearized criterion for stability" [4]. Small deflections and a linearized criterion for stability are assumed. The linearized criterion for stability approximates stress resultants and stress couples based on Saint Venant's formulas along with other formulas that require modifications in order to yield close approximations to lateral torsional buckling. In Love's article about the linearized

criterion, he recommends using an unextended central line for a first approximation when bending and twisting are important features [5]. An unextended central line refers to keeping the beam-tip distance from the plane of the beam root, or constrained end of the cantilever beam, constant throughout deflection. Although the literature search for this thesis project has only found lateral torsional buckling equations that use the above assumptions, it should be noted that Love inferred that a better approximation could be made after the first approximation.

2.3 Compliant Mechanisms

The purpose of a pseudo-rigid-body model is to provide a simple method of analyzing systems that undergo large, nonlinear deflections. Pseudo-rigid-body models have been used with great success in predicting force-deflection characteristics of compliant mechanisms in two dimensions [1]. Currently, Pseudo-rigid-body models have been defined for many two-dimensional cases where it is assumed that all deflections remain in a single plane. The key discovery that makes a pseudo-rigid-body model so useful is the near-circular deflection path of beams. This enables engineers to replace complex models of flexible members with a much simpler model of two-dimensional rigid links and springs.

Although mechanisms are often designed in a single plane, it may be necessary to consider how a mechanism will react to out-of-plane forces. A good understanding of out-of-plane forces can lead to increased performance while making a design more robust in a three-dimensional world. For instance, a compliant ortho-planar spring or COPS (see Figure 2.2) is a good example of a mechanism where analyzing out-of-plane forces would help increase performance and understand limitations. [6]

After being stamped out of a single sheet of material, the compliant ortho-planar spring can then be used in a continuously variable transmission (CVT), as shown in Figure 2.2. As the center platform moves up and down the shaft, colored black in Figure 2.2, the legs deflect and provide a restoring force. While this spring performs its function, it also spins. Centripetal forces change the spring constant and deflection paths of the legs. If the centripetal forces are strong enough, the legs undergo lateral torsional buckling (see Figure 2.3), meaning the spring legs would twist and buckle out of the plane of the centripetal distributed force. This could be damaging or dangerous depending on the application.

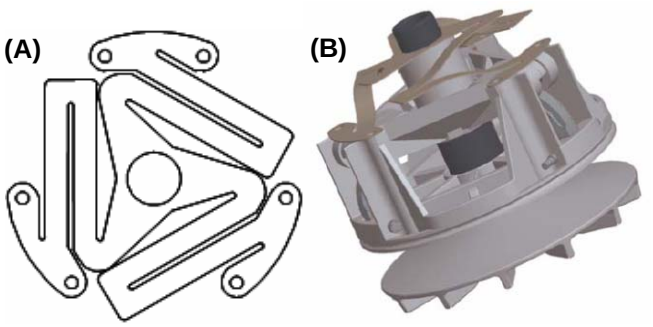


Figure 2.2: (A) Compliant ortho-planar spring (COPS) [6]
(B) CVT-driven clutch with coil spring replaced by a COPS [6]

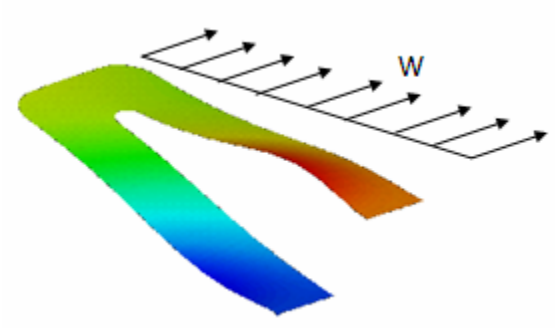


Figure 2.3: Centripetal forces modeled as a distributed load *W* cause the spring leg to undergo lateral torsional buckling [6]

A 2-D pseudo-rigid-body model could not predict the ortho-planar spring leg's out-of-plane buckling behavior. If a 3-D pseudo-rigid-body model existed that captured the lateral torsional buckling behavior of a rectangular beam undergoing large deflections, it would provide engineers with a useful design tool and be the groundwork for exciting, new inventions.

The closest related work and inspiration for this thesis comes from a paper by Rasmussen et al., in which the idea of expanding the 2-D pseudo-rigid-body model into three dimensions is introduced [3]. The 3-D model was investigated after it was discovered that ortho-planar spring legs designed with 2-D pseudo-rigid-body models will buckle out of plane because of centripetal forces, and therefore not follow the 2-D approach. The paper "explores the possibility of deriving a 3D PRBM for non-follower, end loaded, rectangular cross-section cantilever beams" [3]. As with other pseudo-rigid-body models, it is assumed the beam will deflect in a circular arc. He reasons that a beam with a circular cross-section, if deflected in 3-D, would follow a spherical shell within

some limit. This can easily be proven with a 2-D pseudo-rigid-body model and a transformation of the coordinate system. The next step is deflecting a rectangular beam the same way. It is successfully shown that the deflection of a rectangular beam follows a spherical shell with no more than 5 percent error over a given range. This means a 3-D pseudo-rigid-body could be designed to accurately approximate 3-D beam deflections (see Figure 2.4).

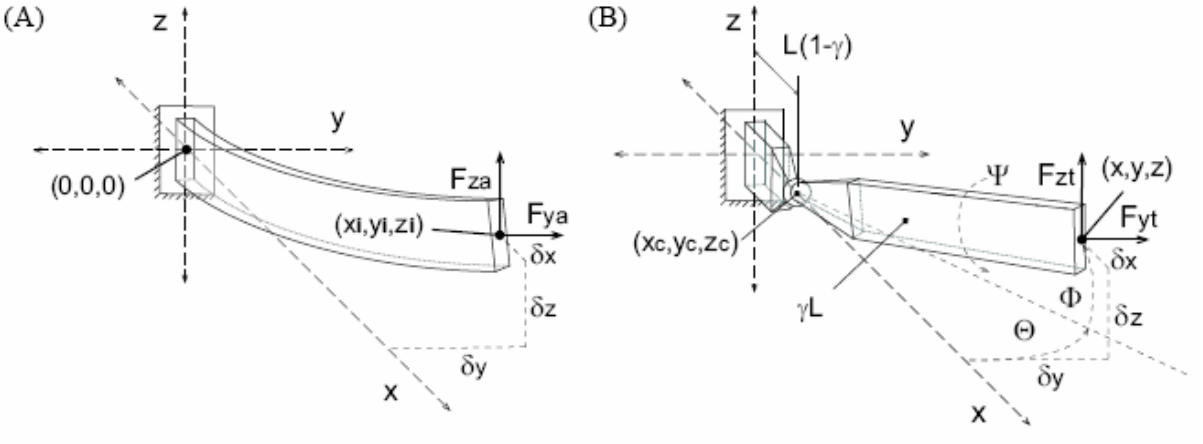


Figure 2.4: (A) Deflected beam
(B) 3-D pseudo-rigid-body model replacement of beam [3]

Research related to lateral torsional buckling has been focused on predicting the initiation of the buckling process. No literature has been found that describes the elastic post buckling response of a cantilever beam beyond small deflection approximations.

More importantly there is no simple design tool that can quickly and accurately predict the lateral torsional buckling phenomena essential to rectangular cantilevered beams subjected to 3-D forces (see Figure 2.5). Aside from the paper by Rasmussen, no research has been found that delves into a 3-D pseudo-rigid-body model. If a 3-D pseudo-rigid-body model is to be created, much force and displacement information will need to be added to the current knowledge base of beam stability. However, beam stability theory deals with why the phenomenon occurs while pseudo-rigid-body models focus solely on predicting beam behavior for design purposes.

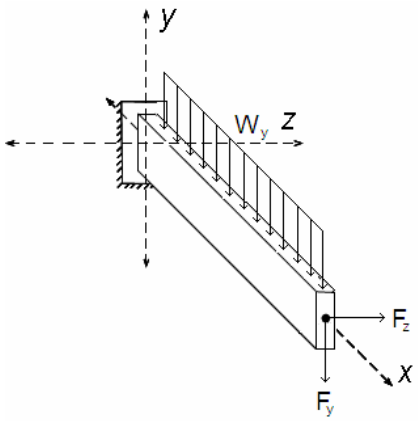


Figure 2.5: Example of 3-D forces

2.4 Chain Algorithm

The chain calculation procedure is an alternative to finite element analysis and can find solutions efficiently in many applications. "A numerical chain algorithm uses the same basic elements and stiffness matrix theory as those of conventional finite elements, but uses a different technique to combine and solve the resulting equations" [1]. Each beam element is calculated in succession as a cantilever beam. Once an element's deflections are solved, all succeeding elements or those closer to the beam tip are rotated and lined up starting at the deflected element's tip.

Figure 2.6, below, illustrates how a single element is analyzed. Starting at the beam root, the deflection path of each element is calculated. At element *i*, the beam has undergone rigid-body translation because of the deflection of previous elements. It has also undergone rigid-body rotation $\Delta\Theta_{i-1}$ because of the elastic deflection caused by previous elements. The forces are broken down into the element's local coordinate system and any moment is also added to the beam tip. "The directions of the loads are not permitted to change during deformation, therefore 'follower' forces are not included" [7]. The loading problem is then solved at the element level, causing a rotation $\Delta\theta_i$ at the beam tip node *i*.

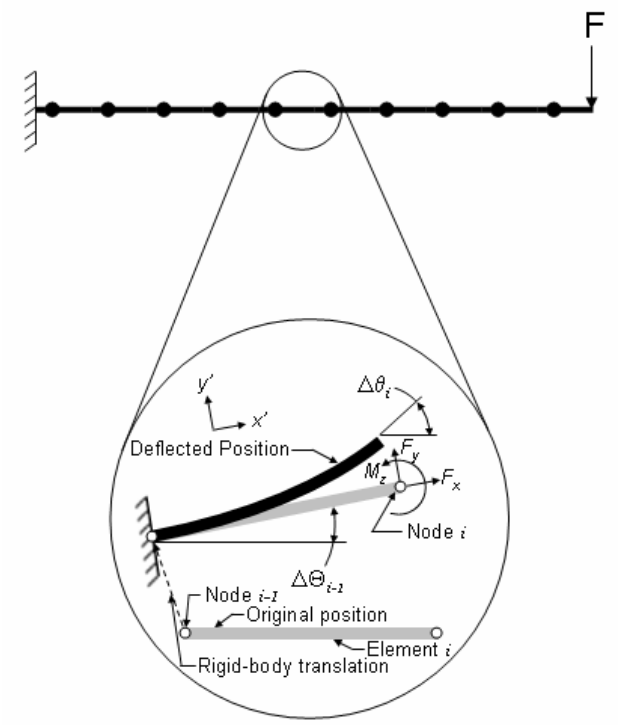


Figure 2.6: Analysis of a single chain algorithm element

Once an element has been analyzed, forces and moments on elements down the line need to be recalculated with the latest position and rotation information, otherwise these inaccuracies will create considerable error. Iterating through the succession of elements can be repeated until the desired accuracy is achieved. Load increments can be adjusted for better efficiency. Load increments also need to be small enough to allow the chain algorithm to converge instead of become unstable. In general "if a large number of load increments are used, a small number of iterations at the end are needed, and vice versa" [1].

Different boundary conditions can be used in conjunction with the chain algorithm. "Boundary conditions may not be limited to nodal displacements when using chain calculations with Newton-Raphson iterations. Axial stresses, transverse loads, strain energy, bending moments, and other parameters may be used as boundary conditions as well" [8]. Using the Newton-Raphson method to solve a chain algorithm with a variety of boundary conditions requires making an initial guess at the correct solution. "Choosing an initial guess within the domain results in a guaranteed convergence to the root in that domain" [8]. The domain describes part of the design space in which the Newton-Raphson method will be searching. If the

design space is very "bumpy," the guess needs to be close enough to the correct domain to allow the optimization method to converge to the right solution. Otherwise, "if the estimate is not in that domain, the method may either diverge or converge to another root" [8].

Hill proposes two schemes to enhance the convergence of the Newton-Raphson method. First, increase the domain, and second, find an estimate in the correct domain. This method of applying different types of boundary conditions is "straightforward, economical, and can be easily implemented on small computer systems. It is, therefore, found suitable for use, with the Newton-Raphson iteration as the shooting method, for compliant mechanism analysis and synthesis problems" [8].

Thus far, the approach described has been for a 2-D chain algorithm. Nahvi, in his dissertation, describes how to make a 3-D chain algorithm. He begins by replacing sine and cosine with three Eularian angles. The Eularian angles allow a vector to undergo a series of three rotations that retain the magnitude of the vector but rotate to a new position with respect to a global coordinate system. These angles are used in a rotation matrix that computes the three rotations simultaneously. However, this method of rotation adds a challenge because successive rotations around a set of axes are non-commutative. "In other words, a general three-dimensional, nonlinear beam element formulation cannot be derived by a simple extension of the two-dimensional formulation, because three-dimensional analysis deals with large rotations that are not vector quantities" [9]. It is very cumbersome to calculate the 3-D deflection in a beam element and then try to systematically find the three Eularian angles that describe that rotation. This problem is overcome by using semitangential rotations. Semitangential rotations are commutative and thus, vector quantities. They are basically infinitesimal rotations less than about 10 degrees. Equation 2.1 illustrates how the semitangential rotation method simplifies the ε_x rotation about the X axes using a rotation matrix.

$$[\varepsilon_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varepsilon_x & \sin\varepsilon_x \\ 0 & -\sin\varepsilon_x & \cos\varepsilon_x \end{bmatrix} \cong \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \varepsilon_x \\ 0 & -\varepsilon_x & 1 \end{bmatrix} \tag{2.1}$$

Large matrices describing deflections and force deflection relationships on single elements were expounded by Nahvi but will not be discussed in this thesis because they diverge from the pseudo-rigid-body model methodology. The 3-D chain cantilever beam study was performed

using a circular cross-sectioned beam. Different combinations of forces and moments were placed at the beam tip in three cases and compared to the finite element analysis method. Difference errors between 0.1% and 5.6% were recorded [9]. "Thus, it may be concluded that the proposed chain algorithm is appropriate for use in solving large-deflection, three-dimensional cantilevered beam problems consisting of straight beam elements" [9].

2.5 Method of Superposition

"With few exceptions, the effect (stress, strain, or deflection) produced on an elastic system by any final state of loading is the same whether the forces that constitute that loading are applied simultaneously or in any given sequence and in the result of the effect that the several forces would produce if each acted singly" [10]. Exceptions occur when a set of forces create a displacement that would actually not happen. This would be the case, for example, when a beam subjected to an axial and transverse force is deflected appreciably in the transverse direction by the transverse force. The result would be that the axial force would cause bending, as shown in Figure 2.7. If the model were broken down into its components using the method of superposition, as shown in Figure 2.8, the axial force would only apply tension to the beam, lengthening it.

The actual bending phenomenon would appear to only be a function of the transverse force, which would be false. In considering the implications of superposition for a large deflection case such as lateral torsional buckling, care must be taken that no unwanted interactions occur between the simplified model components. The successful implementation of superposition in this thesis project can greatly increase the scope of the 3-D pseudo-rigid-body model, making it a more general tool for designers.

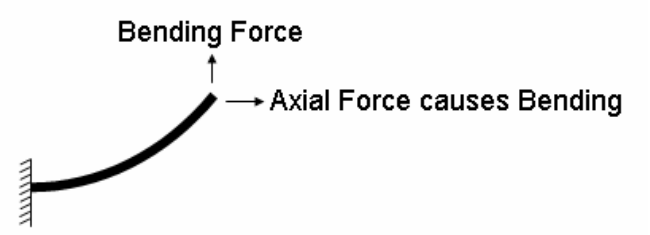


Figure 2.7: Beam subjected to axial and large transverse deflection

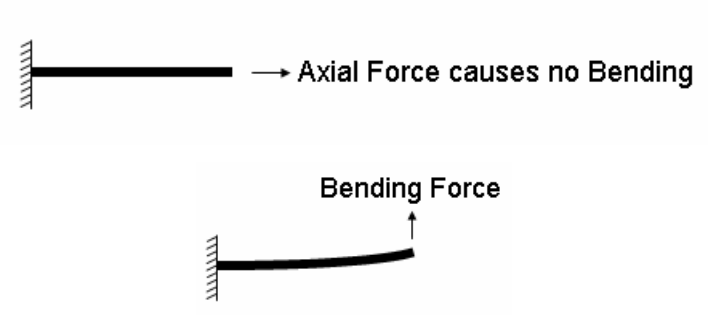


Figure 2.8: Beam components subjected to axial and bending forces

2.6 Recent Work combining the Chain Algorithm and the Pseudo-Rigid-Body Model

The idea of a pseudo-rigid-body model chain algorithm was introduced by Pauly. It relies on pseudo-rigid-body elements while implementing the chain algorithm in the same way as previously discussed. "Such a 'pseudo-rigid-body model chain' would possess dual advantages of expediency of modeling through the use of pseudo-rigid-body representations of compliant segments, and the inherent flexibility of the chain algorithm to geometry and load boundary conditions" [11] .

Two limitations to using pseudo-rigid-body modeling techniques are noted:

1. "In its current form, suitable modifications to existing conventions in the pseudo-rigid-body modeling technique are required in order to apply the parametric relationships developed…to determine the deformation kinematics of compliant segments for varied end force loading situations" [11].
2. "Presently, generalized pseudo-rigid-body models of reasonable accuracy are unavailable for compliant members with combined end force and moment load boundary conditions" [11].

The method of superposition was introduced by Pauly as a way to remedy these limitations [11]. The tip force and tip moment load pseudo-rigid-body models would be calculated separately, then their displacement results would be superposed on each other. This method provided reasonable approximations but one limitation remained: "The deformation kinematics of each element that comprises the planar continuum is determined by superposing the linear

displacements obtained from two separate pseudo-rigid-body models having different pivot locations along the undeformed compliant segment" [11]. In other words, error was introduced into the chain approximation because of incompatibilities in superimposing the pseudo-rigid-body models on each other.

Pauly, in a second paper, proposes using a "rudimentary equivalent pseudo-rigid-body model to represent compliant segments with combined load boundary conditions in the pseudo-rigid-body model chain algorithm" [12]. This method finds an equivalent pseudo-rigid-body model by adding the displacement of the two pseudo-rigid-body models as shown in equation 2.2:

$$\Theta_i^e = \Theta_i^f + \Theta_i^m \tag{2.2}$$

In equation 2.2, Θ_i^f represents the angular displacement in the force tip load pseudo-rigid-body model, Θ_i^m represents the angular displacement in the end-moment loaded pseudo-rigid-body model, and Θ_i^e represents the equivalent angular displacement. The subscript i denotes the equivalent model's location along the chain algorithm. The equivalent characteristic radius factor γ_i^e is set to equal the force model's characteristic radius factor γ_i^f with negligible error. The equivalent spring stiffness coefficients are found using equation 2.3:

$$\frac{1}{k^e} = \left(\frac{1}{k_1} + \frac{1}{k_2} \right) \tag{2.3}$$

The equivalent pseudo-rigid-body model remains incomplete because a suitable equivalent parametric angle coefficient has yet to be discovered. With current research, equivalent angle coefficients are found through trial and error. Once found, they are only applicable for special cases.

The method of superposition and use of an equivalent pseudo-rigid-body model would be useful in a chain algorithm to describe large deflections and the force-deflection relationship in individual segments. Research is currently being done in this area with the hope of making the theory of superposition compatible with the methodology of pseudo-rigid-body models [12].

Chapter 3

Large Deflection Analysis Methods

3.1 Introduction

This chapter presents two methods for analyzing large deflections of lateral torsional buckling problems. The first is useful for approximating deflections and beam-tip orientations for simple cases of lateral torsional buckling. The second method is to create a 3-D chain algorithm with pseudo-rigid-body elements. This method can also be applied to more general large deflection beam analysis with varying beam cross sections and cross-sectional shapes. These analysis methods will focus on prismatic cantilever beams with rectangular cross-sections where the aspect ratio of the cross section is large (see Figure 3.1). Changes in geometric and material properties will be considered.

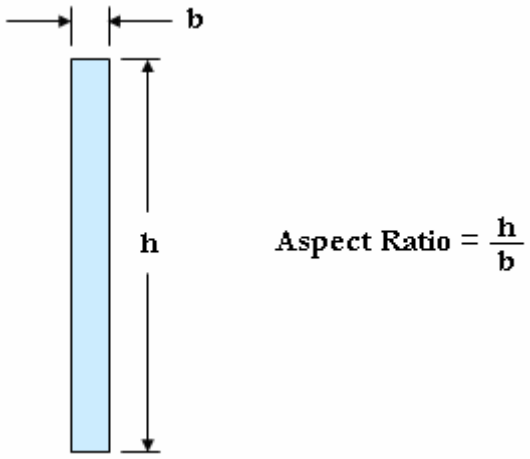


Figure 3.1: Aspect ratio for a rectangular cross-section

3.2 Displacement Analysis of Lateral Torsional Buckling

Using a finite element approach, cantilever beams with large aspect ratios were modeled and parameters were modified using the batch file technique. Table 3.1 shows the input variables. A force and displacement load were then placed at the beam tip, as shown in Figure 3.2.

Table 3.1: Input and output variables

Input Variables	Output Variables
Length	Tip Displacement in the X, Y, and Z Directions
Height	Tip Rotations around the X, Y, and Z Axes
Width	Reaction Forces in the X, Y, and Z Directions
Young's Modulus	Reaction Moments around the X, Y, and Z Axes
Poison's Ratio	
Displacement Load Steps	

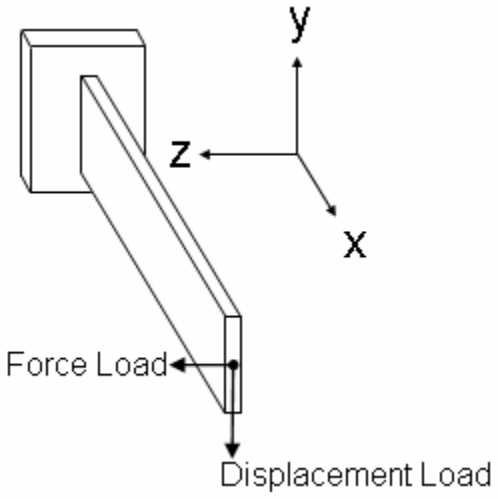


Figure 3.2: Initial beam loading

The force loads were first placed in the more compliant, or Z-direction, and the equilibrium position was determined. Then the displacement load was applied in small increments. The force load was held constant while the displacement load increased by small percents of the beam length until the beam's Y-deflection was 80% of its length, as shown in Figure 3.3.

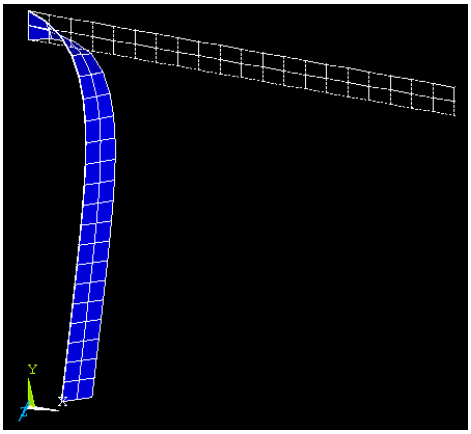


Figure 3.3: Beam buckled to 80% of its length along the Y-axis

The output variables shown in Table 3.1 were saved in a text file. The data was nondimensionalized to reduce the number of variables. The displacements were nondimensionalized with respect to length, and the absolute value of the displacements and rotations was taken. Dimensionless data was graphed so symmetries and patterns could be found. Predictable patterns were approximated with closed-form equations.

The above example used a single displacement load and a force load to illustrate the data collection process from a particular design space. However, this process began with a single vertical displacement load placed on the cantilever beam tip. A small Z-direction force was added only long enough to initiate buckling and then be removed. The iterative displacement load deflected up to 80% of the beam length. Once the important aspects of the single displacement load system had been analyzed, functions and graphs were formulated that approximated the beam displacement and rotation characteristics. Then a second dimension of forces was added and a new set of functions that modify the previous patterns and symmetries were created. This process could continue indefinitely as forces and moments are placed in the three dimensions of space at different locations along the beam. This method soon exhausted the

capability of computers as it became exponentially time consuming. Using this approach, an understanding of the mechanics of the lateral torsional buckling phenomenon was gained and an important deflection path discovery was made that will be discussed in Chapter 4.

It should also be noted that this method does not consider all the possible displacements that different orders of applied forces would cause. Order does play a part in deciding how a large-deflection beam will deflect. A simple 2-D example is shown in Figure 3.4, below. Three different forces are applied one after the other with the following result:

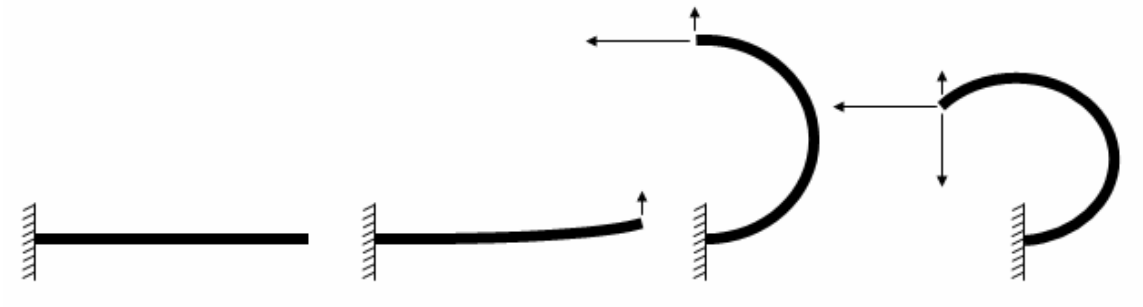


Figure 3.4: Three different forces applied in succession to a beam

Now the same three forces are placed at the beam tip in a different order. As can be seen in Figure 3.5, a completely different result is achieved.

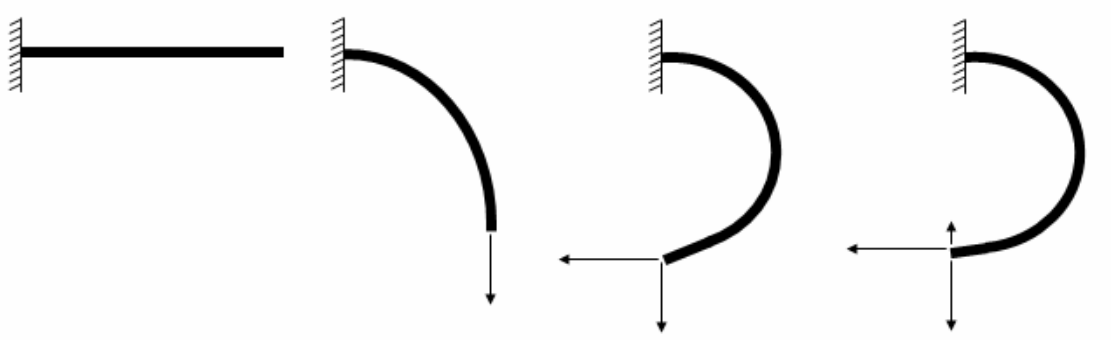


Figure 3.5: Three identical forces applied in a different order to a beam

Order is also important in large deflection, 3-D loading conditions. With a single force in the Y-direction and the same coordinate system, illustrated in Figure 3.2, a beam may buckle in either the positive or negative Z-direction. When a Z-direction force is added and it is not too great, placing forces in different orders on the beam causes different 3-D displacements. If an X-direction force were also added, then a complication similar to the 2-D example in Figures 3.4 and 3.5 could occur along with the combinations of deflections caused by the Y- and Z-direction forces. Moments may have a similar effect but these possibilities will not be explored in this thesis.

As far as this thesis is concerned, for a single Y-direction force, a perturbation force is applied and then removed after buckling is initiated. This causes the buckling phenomenon to occur in only one direction. When a force is applied in the Z-direction and Y-direction, the Z-direction force is always applied first. Combinations of more than two forces will be discussed in Chapter 4, but only combinations of two forces will be implemented using this displacement synthesis approach.

3.3 Chain Algorithm with Pseudo-Rigid-Body Model Elements

Rigid links and springs can be used to create a theoretical model of lateral torsional buckling. This model allows the user to input forces and moments along the length of the beam and then retrieve displacement and rotation outputs. Beam rotations can be output in a way that is more useful to designers than raw Euclidean Angle measurements in three dimensions. The model also approximates the location of the actual beam along its length so the designer has a good idea of the beam's location in space.

Included in this system of linkages is the theory behind chain algorithms. The lateral torsional buckling phenomenon is broken down into its principal components. These components will be modeled using pseudo-rigid-body models. These pseudo-rigid-body models will then be superimposed on each other, creating single elements to the chain algorithm. The elements of the chain algorithm, when lined up and implemented, will be able to accurately predict combinations of force and moment loads on rectangular cantilever beams.

3.4 Conclusion

The first method, displacement analysis, may be useful in solving simple large deflection buckling problems. This method highlights some interesting characteristics of lateral torsional buckling and could be a springboard for future research of large deflection buckling problems. However, as forces and moments are added to the beam, the amount of computing to describe a design space and functions required to describe displacements approaches infinity.

The second method, the chain algorithm with pseudo-rigid-body model elements, relies on modeling different types and different amounts of displacements along the length of a buckled beam. They are then modeled as pseudo-rigid-body models and solved individually in sequence using a chain algorithm. The chain algorithm has the ability to efficiently handle force and moment loads applied anywhere along the beam.

Chapter 4

Displacement Analysis

4.1 Introduction

Displacement analysis takes the complex lateral torsional buckling process and describes particular aspects of it in a simple way. When looking at a buckled beam, it may be a surprise to think that the displacements and rotations of the beam have simple patterns that make them easier to understand. The realization of these ideas and patterns can allow designers to make quick approximations and break down lateral torsional buckling into 2-D components.

4.2 Single Displacement Load

A single displacement force "F" was applied to the beam tip in the negative Y-direction as shown in Figure 4.1. A small perturbation force in the Z-direction was applied long enough to initiate buckling and then was removed. Beams of different aspect ratios, lengths, and material properties were deflected in the negative Y-direction 80% of the beam length using the finite element method.

From the beam deflection results, the following discovery was made. If a pane of glass were placed in front of the beam tip in the YZ plane and a second pane of glass were placed at the side of the beam in the XY plane, as the beam deflected, the tip would draw an arc on each pane (see Figure 4.2).

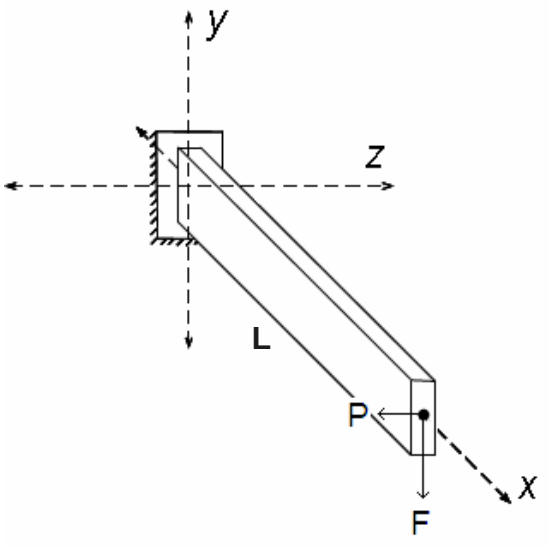


Figure 4.1: Single displacement force "F" applied to beam tip in negative Y-direction

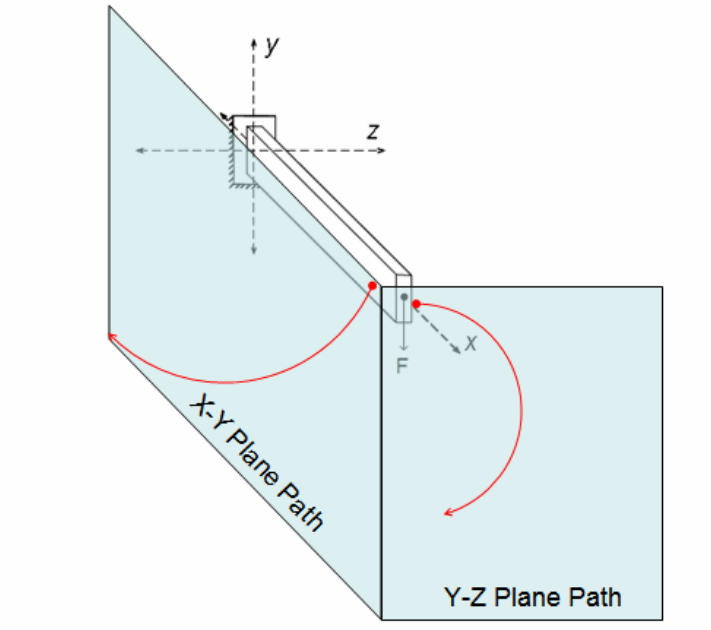


Figure 4.2: XY and YZ plane deflection paths

Since the deflection paths appeared circular in the XY and YZ planes, Excel solver was used to find a best-fit circle that matched these arcs. This was done by choosing a guess value for the circle center. The first guess was the beam root where X, Y, and Z were equal to zero. The

absolute values of the deflections were taken. Then the distance, or approximate radius, between each displacement point along the beam path in the XY plane and the circle center point was found. The average radius was also found and then the difference between the average radius and each approximate radius was computed. The discrepancies were then summed together. The Excel solver minimized this discrepancy by modifying the XY or YZ plane coordinates until the best-fit circle was found.

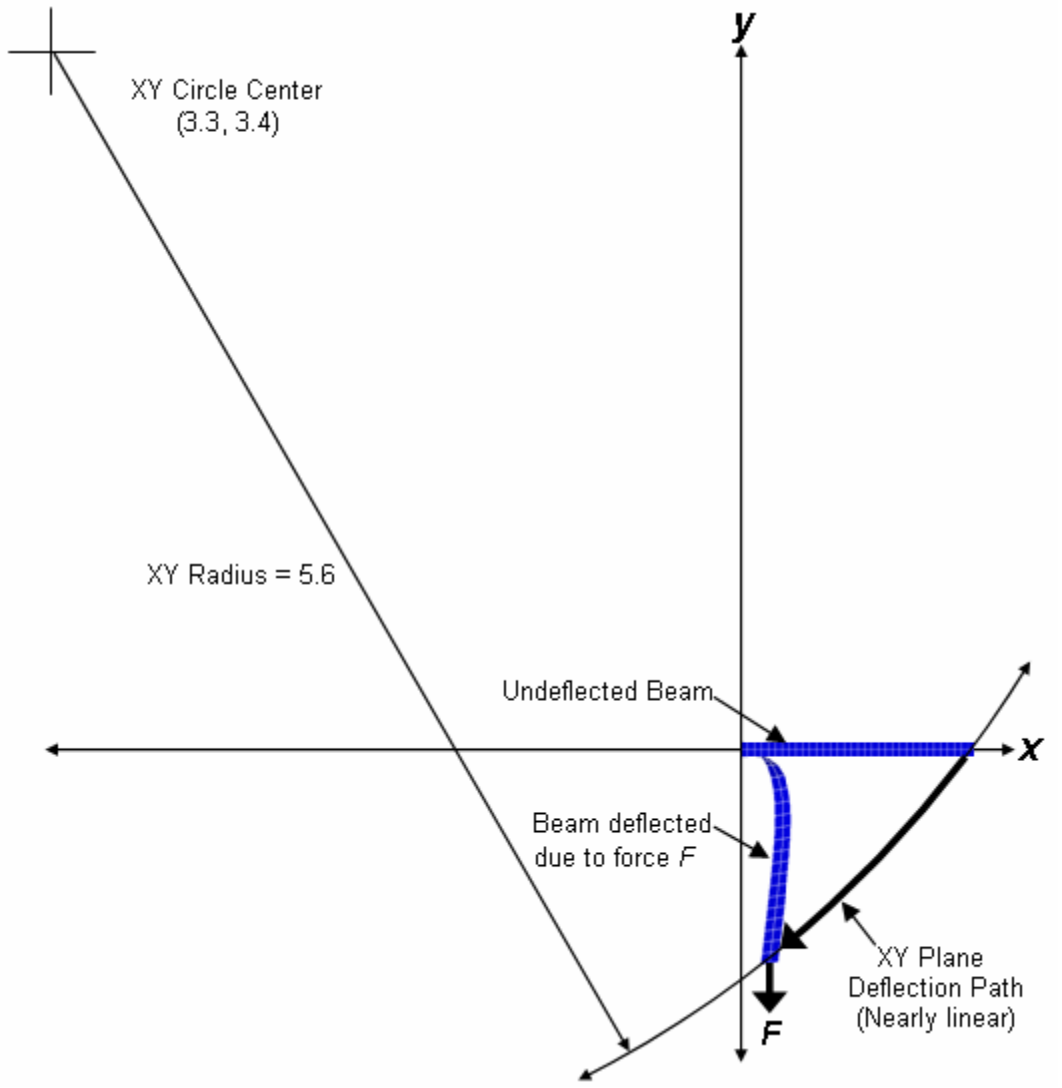
The radius of the circular curve in the XY plane is large in comparison to the beam length. The circle drawn in the YZ plane is smaller when compared to the beam length. Beams with different material and geometric properties were buckled. When the beam displacements were nondimensionalized with respect to length, it could be seen that the large circular arc remained nearly linear and the small circular arc was scaled directly to the beam length with very little variation. Figure 4.3 shows these circles, drawn to scale, for a polypropylene beam.

An undeflected and a deflected dimensionless beam were each placed in the XY and YZ planes to further clarify the deflection paths. In the XY plane the undeflected beam is viewed straight down its central axis so only the beam tip, or rectangular cross-sectional shape, can be seen.

To qualitatively validate these displacement findings from Ansys, a polypropylene Ansys beam model with an aspect ratio of 25 was compared to an actual steel beam with an aspect ratio of 45 (see Figure 4.4). Weights were applied to the end of the steel beam and the change in the XY and YZ planes was measured with a square and a ruler. Once the displacement data had been recorded, the XY and YZ displacement paths for the polypropylene and steel beams were nondimensionalized with respect to the beam lengths, and the absolute values of the plots are shown in Figures 4.5 and 4.6.

As beams displaced in the Y-direction, there was a small amount of deviation from a perfect circle in both the XY and YZ planes because the beam displacement did not follow a perfect circular path. Figures 4.7 and 4.8 represent the common deviation trend seen between a perfect circle and the nondimensionalized beam path calculated using Ansys. The error, or difference, was calculated using the following equation:

$$\text{Average Circle Radius} - \text{Approximate Circle Radius} = \text{Dimensionless Error} \qquad (4.1)$$



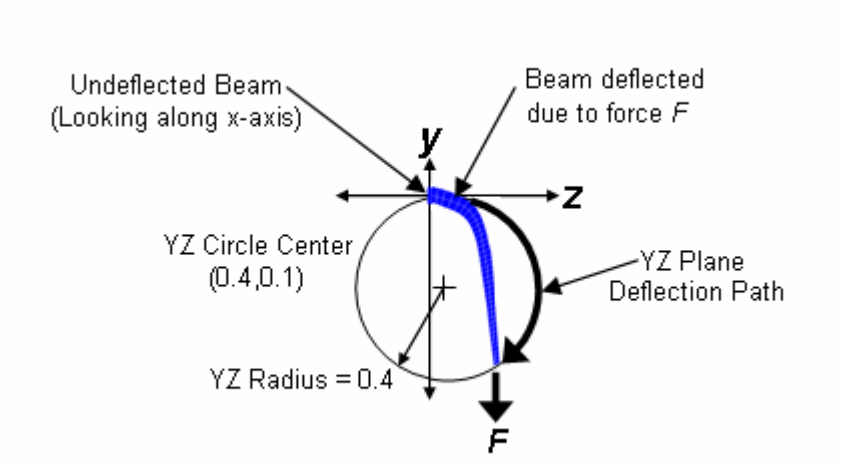


Figure 4.3: Dimensionless beam paths (Drawn to scale)

Figure 4.4: Steel beam

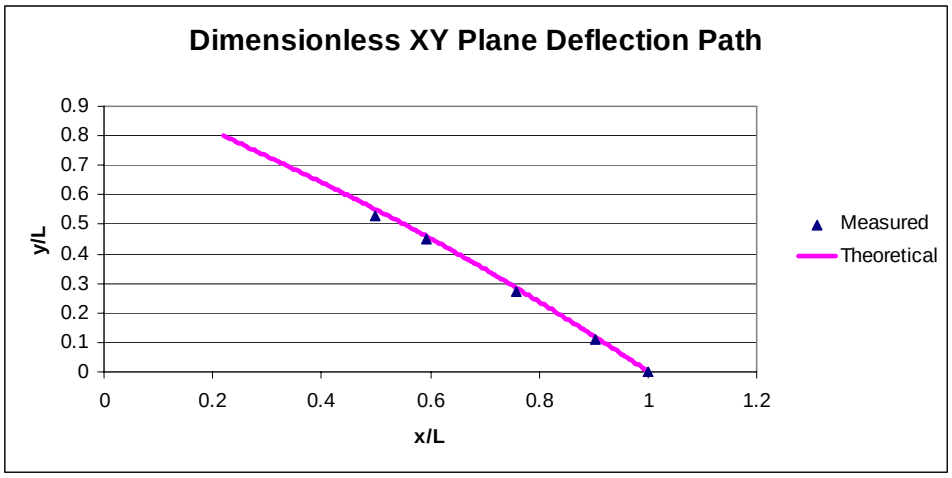


Figure 4.5: Dimensionless XY plane beam tip deflection path

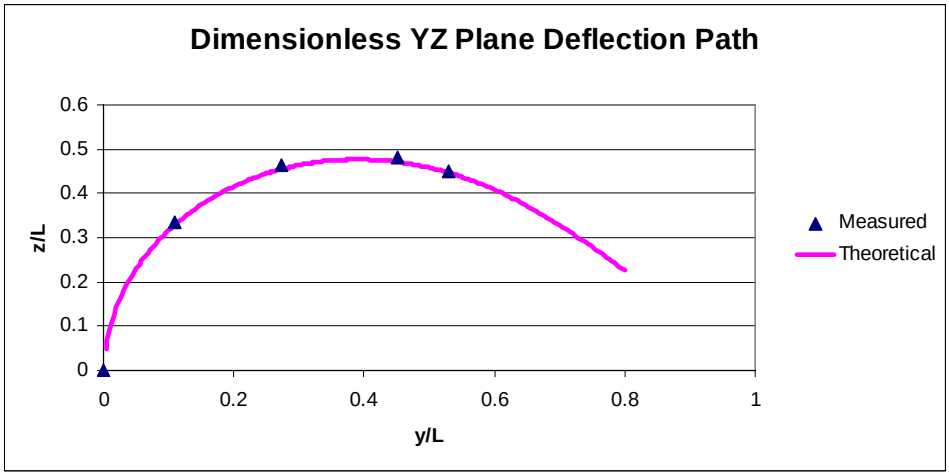


Figure 4.6: Dimensionless YZ plane beam tip deflection path

The two graphs below represent the dimensionless deviation from the best-fit circle for a finite element model of a 48-inch-long cantilever beam calculated by equation 4.1. The height of the cross-section was 2.5 inches while the width of the cross-section was 0.125 inches, making the aspect ratio 20. This model used polypropylene material properties with a Young's modulus of 200,000 psi and a Poisson's ratio of 0.4. All beam deflections were nondimensionalized with respect to the beam length. The percent error in both graphs below can be scaled with the beam length. In both graphs the error is small, but it is greater in the XY plane. The maximum error in both plots is larger at the beginning and the end. The large error at the beginning occurs while the beam is making its transition from bending in plane to buckling out of plane. The large error at the end is associated with the displacement force growing exponentially and thus forcing the beam to leave its circular path in the direction of the force. This example of the deviation error is similar for other rectangular cantilever beams where the aspect ratio is above 20.

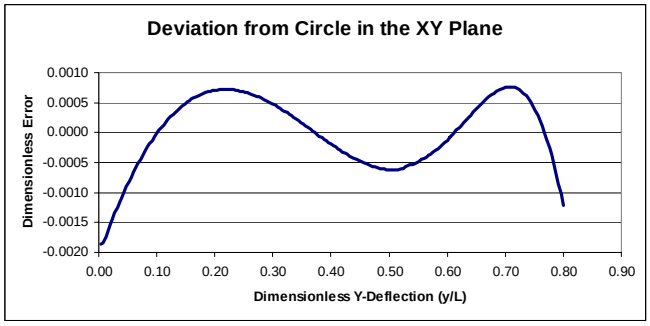


Figure 4.7: XY plane dimensionless error

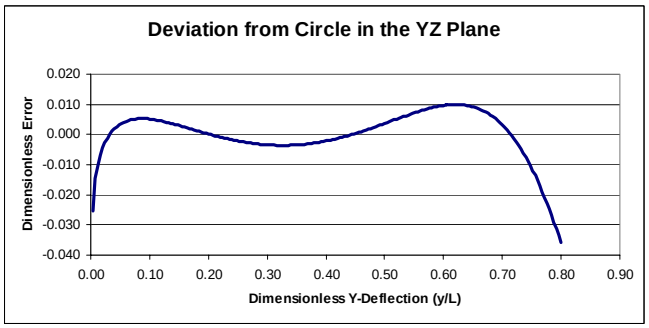


Figure 4.8: YZ plane dimensionless error

Beam tip rotations also remain constant for any dimensionless Y displacement. The following three graphs (see Figures 4.9, 4.10, and 4.11) represent rotation around the X, Y, and Z axes where the axes are located at the beam tip and follow a Euclidean vector rotation process. The key for each graph is coded as follows:

- "P" or "S" represents the material property. In this case "P" is for polypropylene (E = 200,000 psi, υ = 0.4) and "S" is for steel (E = 30,000,000 psi, υ = 0.28).
- "L" stands for length and is followed by the length of the beam.
- "AR" stands for aspect ratio and is followed by the aspect ratio.

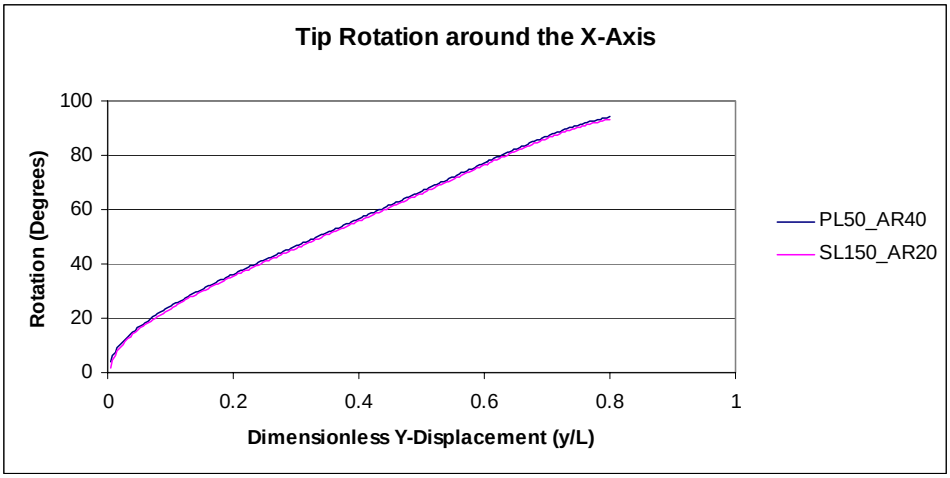


Figure 4.9: X-axis rotation around the beam tip

These graphs show results for two prismatic beams. One beam has the material properties of polypropylene. The beam is 50 inches long and has an aspect ratio of 40. The beam width is 0.125 inches. The second beam is made from steel. It is 150 inches long and has an aspect ratio of 20. The beam width is also 0.125 inches. In each graph the lines are nearly identical with small variation. This adds to the significance of the close displacements between buckling beams with different material and geometric properties. Not only are the displacements tightly matched, but the tip rotations are tightly matched as well. Further analysis shows that for rectangular beams with an aspect ratio larger than 20 and a cross section height at least 20 times smaller than the

length, for these particular boundary contentions, the dimensionless displacement paths and rotations are almost identical.

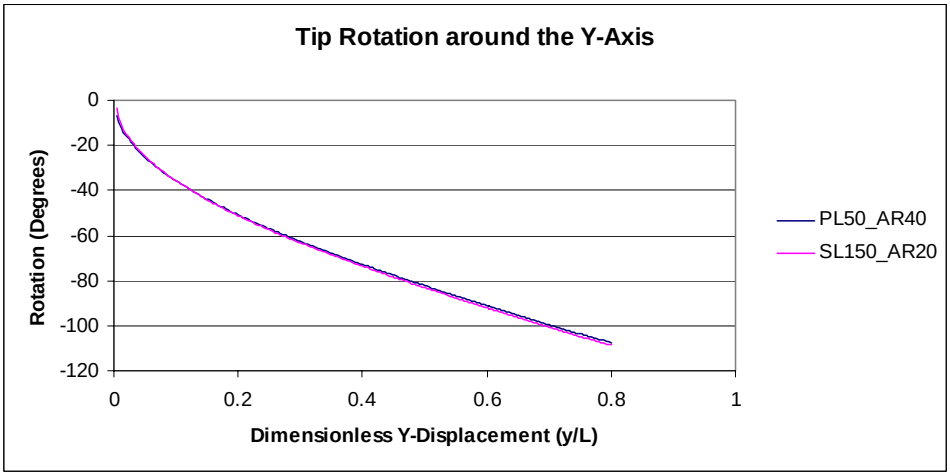


Figure 4.10: Y-axis rotation around the beam tip

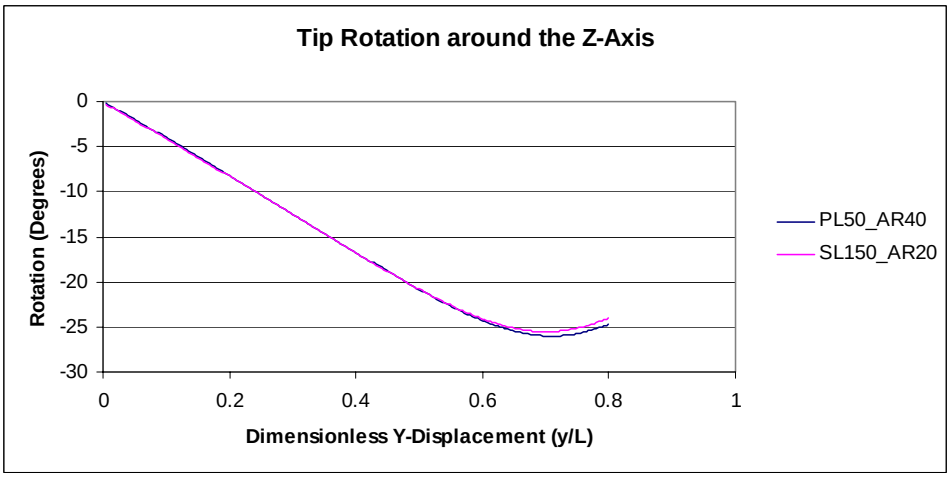


Figure 4.11: Z-axis rotation around the beam tip

Now, holding the material property constant and only changing the beam geometry, the next three graphs (see Figures 4.12, 4.13, and 4.14) show an even closer fit.

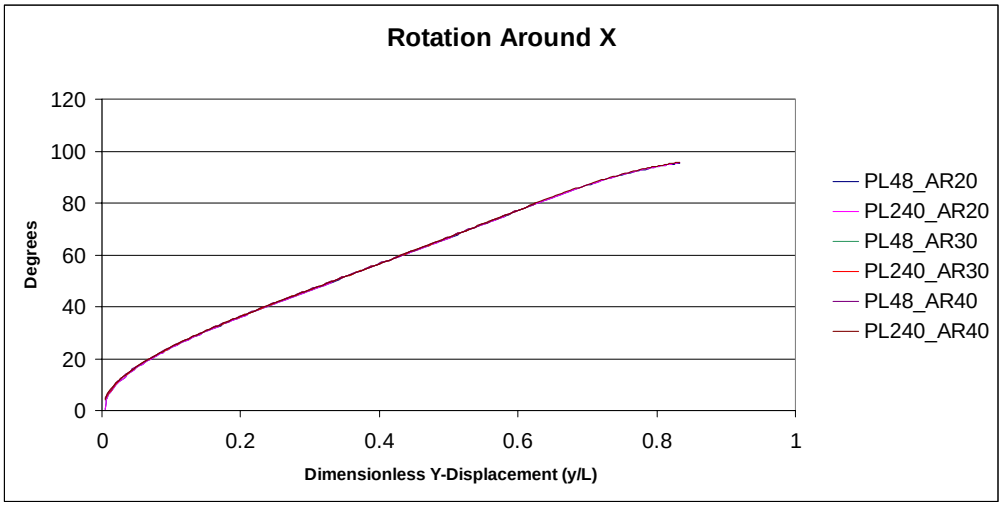


Figure 4.12: Rotation around the X-axis of the polypropylene beam tip

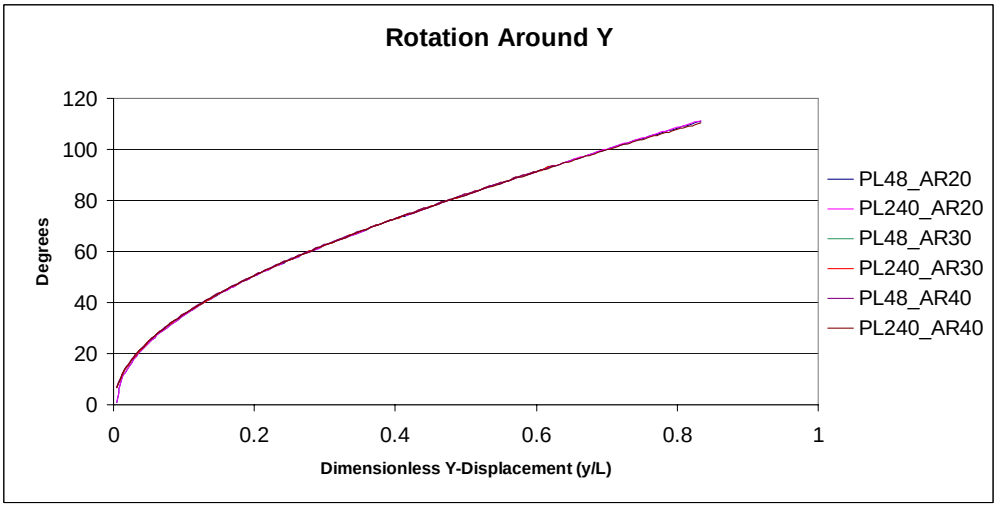


Figure 4.13: Rotation around the Y-axis of the polypropylene beam tip

These graphs represent the absolute value of the difference in rotation between the reference coordinate system at the beam root and the local coordinate system at the beam tip for six nondimensionalized polypropylene beams with aspect ratios between 20 and 40.

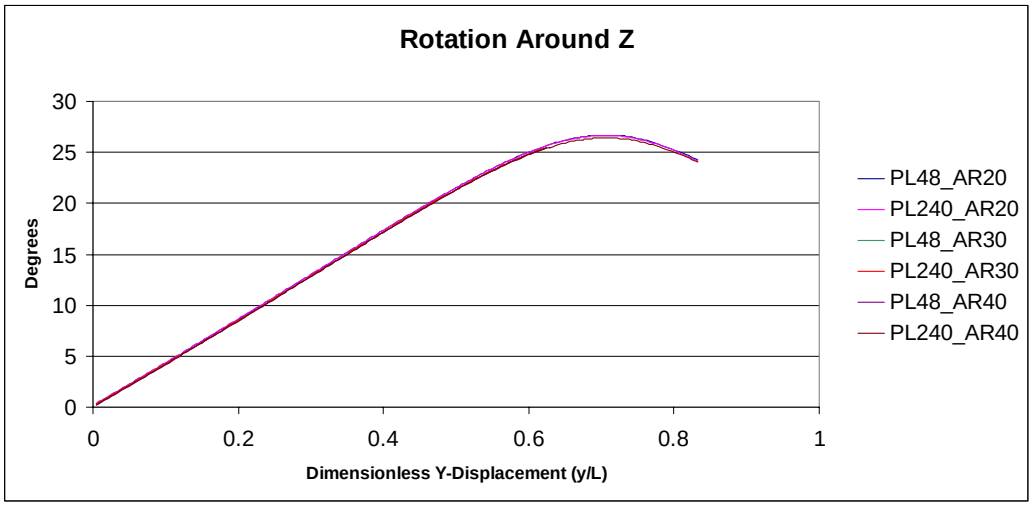


Figure 4.14: Rotation around the Z-axis of the polypropylene beam tip

Noticeable in both sets of X, Y, and Z rotation graphs is the close fit between the lines being compared. Holding the material property constant did tighten the tolerance between the lines. This was most noticeable with the Z-axis where the tail end of the plots remained close instead of deviating slightly like they did in Figure 4.11.

4.3 Displacement and Rotation Variation

Section 4.2 showed that beam displacements and rotations could be closely approximated to be a function of the beam length. Also noticeable was that there was a small difference between beam displacement paths and rotations. Two causes of this difference were a difference in Poisson's ratio and a difference in the aspect ratio. To understand how Poisson's ratio and the beam's aspect ratio cause this difference, it is helpful to break down the three types of deflections and moments a beam is subjected to when it undergoes lateral torsional buckling (see Figure 4.15).

First, it is subjected to a rotation along its changing center axis. The torsion causing this rotation comes about from imperfections in the beam as well as imperfections in the applied load. Second, the beam deflects laterally because of a small but growing fraction of the load pushing against the beam's slender direction. Third, the majority of the load deflects the stiff direction of the beam by a very small amount.

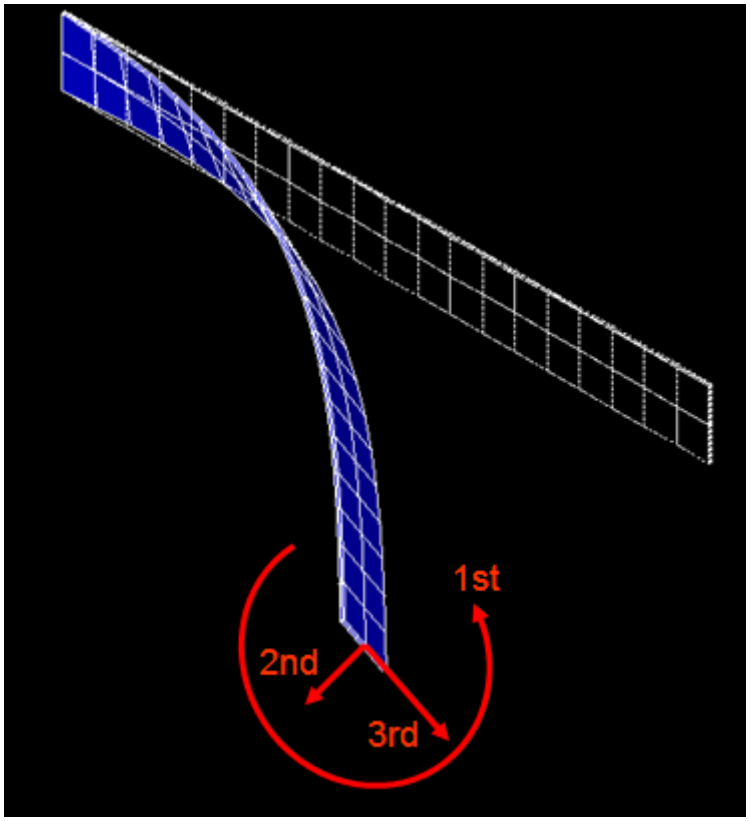


Figure 4.15: Three types of deflections

Along the length of the beam the magnitude of these three deflection components does not remain constant. This is because of the change in the 3-D moment from one end of the beam to the other. Because of these moments, any point along the beam not at the root rotates around its X, Y, and Z axis.

Instead of tabulating the rotation and deflection information as it changes from one end of the beam to the other and then comparing this data to data taken from other beams with different aspect ratios and variations in Poisson's ratio, a simpler relation can be formulated by looking at the three deflection components as pseudo-rigid-body models. The pseudo-rigid-body model with the stiffest spring correlates to bending the beam in its stiffest direction as a cantilever beam. The equation for this tip force-loaded pseudo-rigid-body model torsional spring constant is:

$$K_{stiff} \approx \pi\gamma^2 \frac{EI}{L} \quad \text{where } I = \frac{bh^3}{12} \text{ and } \gamma = 0.85 \quad [1] \tag{4.2}$$

The springs in the other two pseudo-rigid-body models are very weak compared to this spring. One of these weaker springs correlates to bending the beam in its most flexible direction as a cantilever beam. Its torsional spring constant can be defined by the same pseudo-rigid-body model as follows:

$$K_{slender} \approx \pi\gamma^2 \frac{EI}{L} \text{ where } I = \frac{hb^3}{12} \text{ and } \gamma = 0.85 \quad [1] \tag{4.3}$$

The final pseudo-rigid-body model describes pure torsion. Its torsional spring acts along the length of the beam at the center axis. Its torsional spring constant is:

$$K_{torsional} = \frac{GJ}{L} \text{ where } G = \frac{E}{2(1+\upsilon)} \text{ and } J = hb^3\left[\frac{1}{3} - 0.21\left(\frac{b}{h}\right)\left(1 - \frac{b^4}{12h^4}\right)\right] \tag{4.4}$$

Because the stiffest spring is so much stiffer than the other two, the assumption can be made that it is infinitely stiff. The other two springs will be related together as the ratio $K_{slender}$ /$K_{torsion}$. When simplified, this ratio becomes:

$$\frac{K_{slender}}{K_{torsion}} = \frac{\pi\gamma^2(1+\upsilon)}{6\left[\frac{1}{3} - 0.21\left(\frac{b}{h}\right)\left(1 - \frac{b^4}{12h^4}\right)\right]} \tag{4.5}$$

Young's Modulus drops out, leaving the ratio a function of Poisson's ratio and the aspect ratio. This ratio is then plotted against the beam's aspect ratio resulting in the following graph (see Figure 4.16). In this graph the geometry of both beams is the same. The width "b" is equal to 0.125 inches while the length is 20 inches.

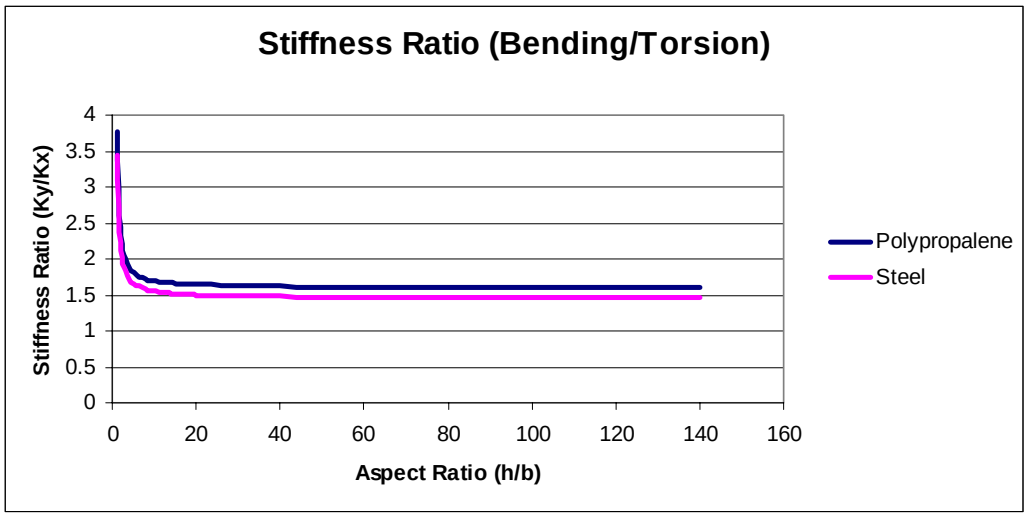


Figure 4.16: Stiffness ratio comparison with two values of Poisson's ratio

The polypropylene and steel beams follow each other closely. At the left end of the graph where the aspect ratio is small, the torsional spring representing the slender stiffness of the beam "$K_{slender}$" is dominant. The beam on this side of the graph is not very slender. Before an aspect ratio of 20, the spring ratio changes drastically. After an aspect ratio of about 20, the ratio has settled down to a near constant relationship between slender-stiffness and torsional-stiffness in the beam. For lateral torsional buckling, this means that above the threshold of an aspect ratio of 20, all beams will react with the same proportionality between bending and torsion. As the aspect ratio increases, a constant relationship can be assumed with increasing accuracy. For aspect ratios below 20, the relationship between the torsional-stiffness and slender-bending stiffness changes drastically. The slender-bending stiffness is dominant below the aspect ratio of 1. It is also dominant over the stiff-bending stiffness not shown in Figure 4.16.

The differences in the two plots are also very noticeable. Poisson's ratio used for polypropylene was 0.4 and Poisson's ratio for steel was assumed to be 0.28. Because the geometry of both beams was exactly the same, the only cause for variation in the two plots was Poisson's ratio. The following graph (see Figure 4.17) represents the most extreme cases for Poisson's ratio.

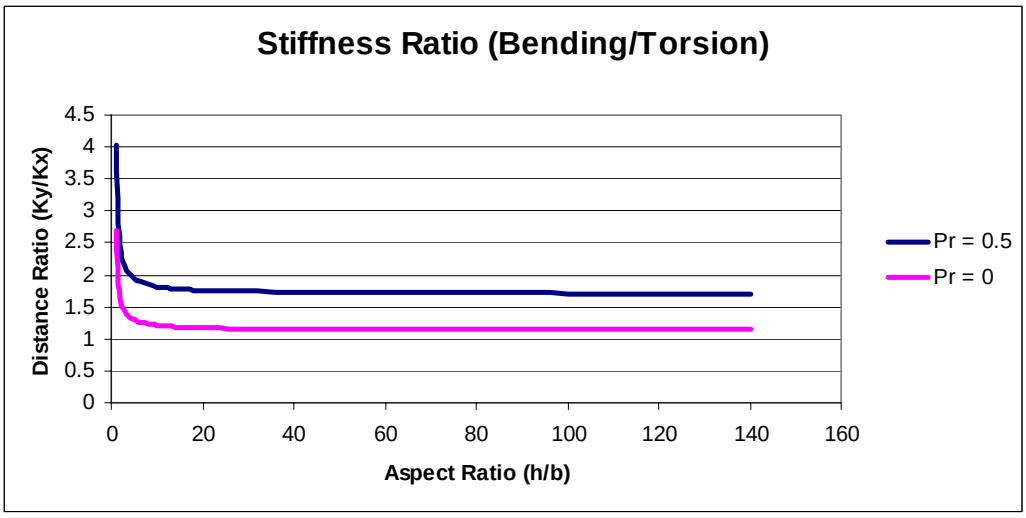


Figure 4.17: Stiffness ratio with minimum and maximum values of Poisson's ratio

The plot of stiffness ratio representing Poisson's ratio equal to 0.5 settles down at a constant value of approximately 1.710, while the stiffness ratio plot representing Poisson's ratio equal to zero settles down at a constant 1.140. This correlates to the following displacement plots (see Figures 4.18 and 4.19)

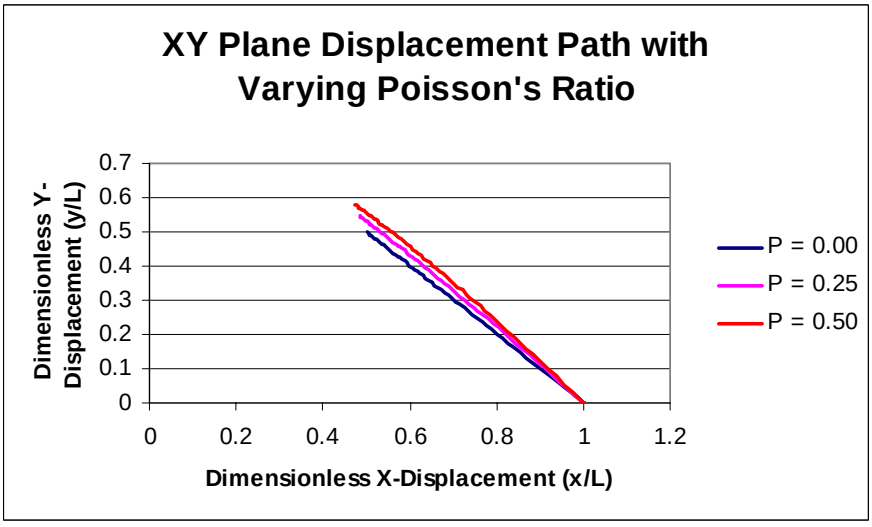


Figure 4.18: Change in XY displacement path caused by Poisson's ratio

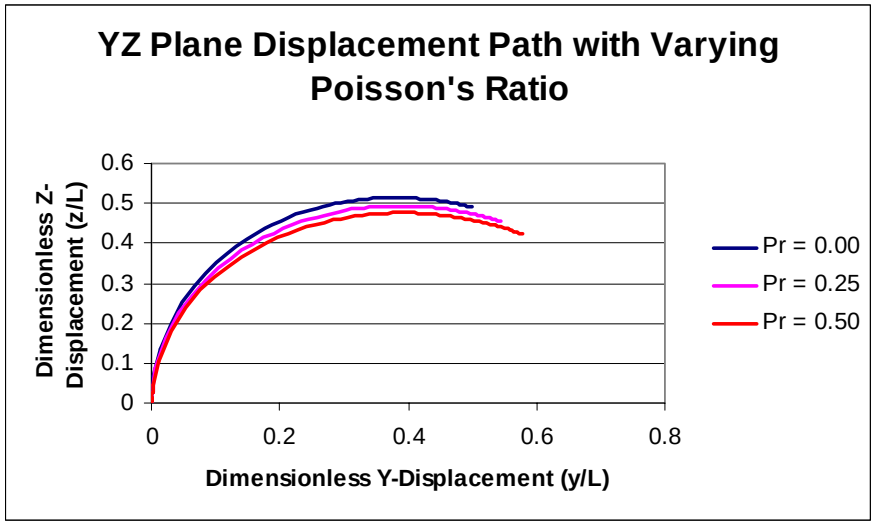


Figure 4.19: Change in YZ displacement path caused by Poisson's ratio

Because Ansys was not able to solve beam problems at extreme values of Poisson's ratio, the method that will be described in Chapter 5 was used to find these plots. As can be seen in these plots, Poisson's ratio has a small role in describing the deflection path of lateral torsional buckling beams. Though materials do not reach these extreme values of Poisson's ratio, these values do create a theoretical envelope and illustrate the magnitude of variation caused by Poisson's ratio. This effect also modified the best-fit circle sizes for these graphs, as can be seen in Tables 4.1 and 4.2.

Table 4.1: Change in circle properties for XY-plane

Pr	XY Circle Radius	X-Coord.	Y-Coord.	Max XY Error
0.00	32.4925	-22.2089	-22.7395	0.000357
0.25	8.9796	-5.7824	-5.8845	0.000637
0.50	5.5763	-3.3856	-3.4439	0.001018

Table 4.2: Change in circle properties for YZ-plane

Pr	YZ Circle Radius	Y-Coord.	Z-Coord.	Max YZ Error
0.00	0.4161	0.4082	0.0924	0.0228
0.25	0.4028	0.3963	0.0842	0.0220
0.50	0.3985	0.3941	0.0724	0.0187

Figure 4.3 in section 4.2 illustrates how to use this tabulated data. As can be seen in the first table, as Poisson's ratio approaches zero, the XY circle radius grows exponentially. The circular arc essentially becomes a straight line. The YZ circle radii growth has a smaller, more linear relationship than the XY circle growth, but the error associated with it is a magnitude higher than it is for the XY data.

Poisson's ratio, μ, and the cross-sectional aspect ratio, h/b, are the two main sources of variation found in this theoretical analysis. Poisson's ratio modifies the beam path and the rotation of the beam tip by a small amount. The aspect ratio plays a large roll in modifying the beam deflection path when the aspect ratio is small. For larger aspect ratios above about 20, there is little effect.

4.4 Buckling Beam Paths for More General End Loads

This section will explore how the beam deflection paths are modified for a cantilever beam with an X- and Y-force load at the tip. The variations described in section 4.3 will be neglected. If this technique were used to explore a design space with multiple forces and moments with reasonable accuracy using current technology, excessive and time-consuming computations would be required. However, if a narrow design space needed to be explored for functions that are fast and efficient approximations for describing changes in deflection paths, then this process would be useful.

A cantilever beam consisting of one-hundred beam elements is loaded at its tip and will undergo lateral torsional buckling. In this case an iterative force load is first added to the beam in the Z-direction as shown in Figure 4.20. This force load will start at nearly Zero and increase in forty uniform increments until the beam is deflected in the Z-direction 80% of its length. For each of these force iterations the beam will undergo forty uniform displacement loads in the Y-

direction. From this method, an accurate design space for a double-tip load displacement path will be gained.

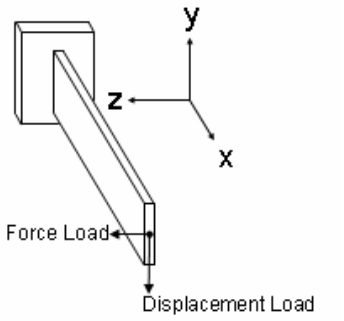


Figure 4.20: Beam with Y- and Z-direction forces at tip

The data received for each force load corresponding to forty displacement iterations in the Y-direction is nondimensionalized and best-fit circles are found for every case. The change in the dimensionless radii as a greater Z-axis force is applied to the beam is shown in Figures 4.21 and 4.22.

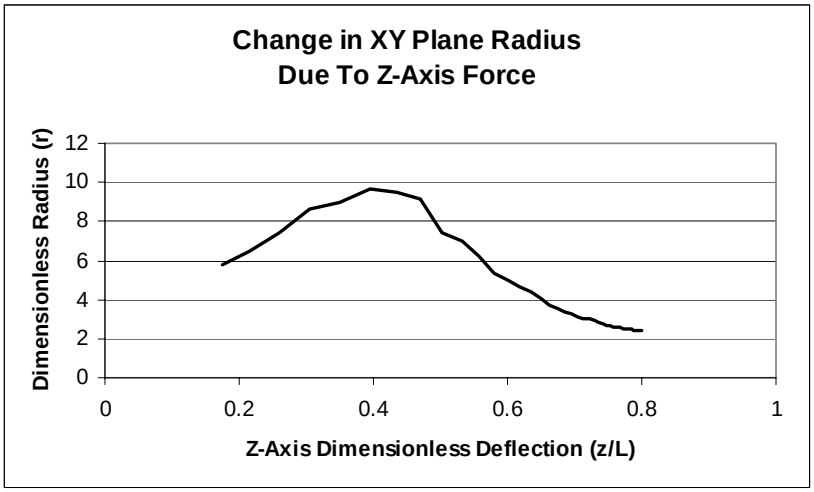


Figure 4.21: Change in dimensionless XY plane radius caused by Z-axis deflection

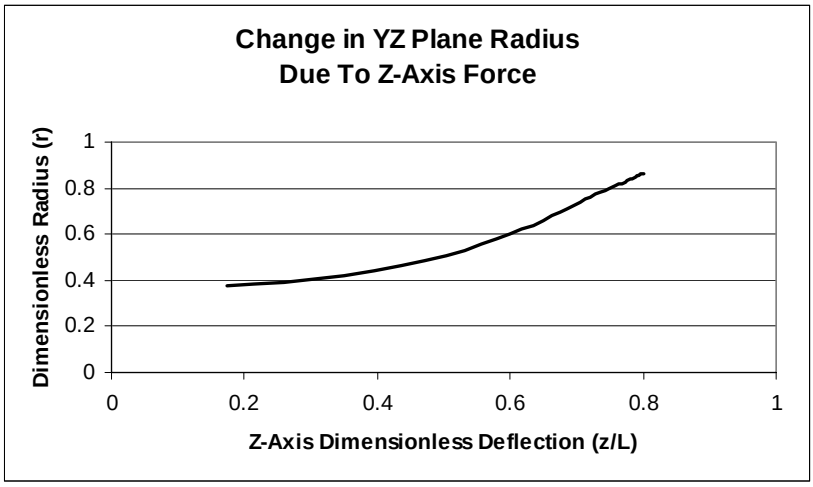


Figure 4.22: Change in dimensionless YZ plane radius caused by Z-axis deflection

The independent variable is the Z-axis displacement in both cases. As shown in Figures 4.21 and 4.22, above, deflecting the beam in the Z-direction before buckling it changes the size of the dimensionless radii described in section 4.2. Noise is also very noticeable in the first plot. This defect is caused by Ansys as it attempts to converge to the right solution. Better plots could be achieved through trial and error, but the process is time-consuming and these plots are acceptable for illustrating the buckling beam paths for this loading scenario.

As already noted, the XY and YZ deflection paths remain circular when the beam is deflected horizontally before the beam is buckled by a vertical force. The deviation from the best-fit circle associated with the change in these radii is described in Figure 4.23.

The error was once again found using equation 4.1, except that in this case the absolute value of the error was also taken. This is evident in the center bulge of the XY circle plot. The maximum error of the YZ plane circle tends to improve when the beam is deflected to the side before buckling. The error associated with the XY plane circle remains nearly the same.

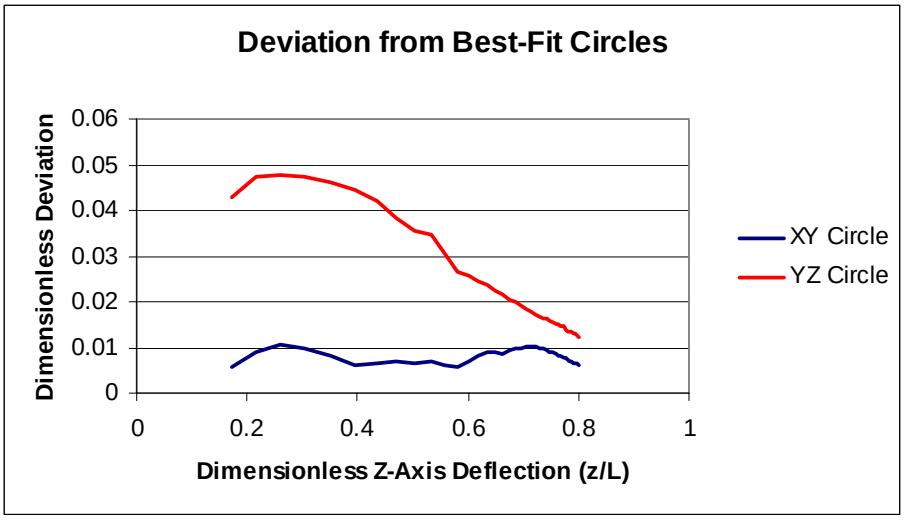


Figure 4.23: Error of dimensionless radii

The location of the circle's center undergoes a shift corresponding to the given Z-axis deflection. Figures 4.24 and 4.25 show this shift in the XY and YZ planes respectively.

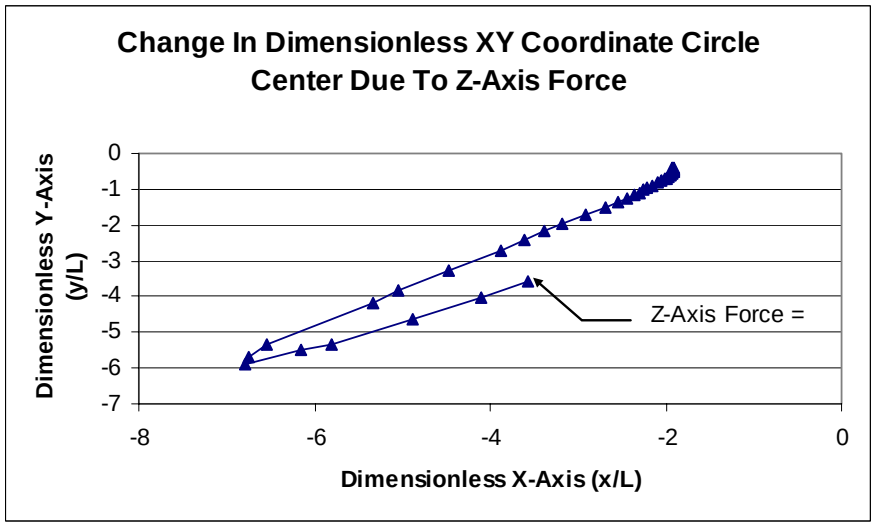


Figure 4.24: Shift in XY circle center

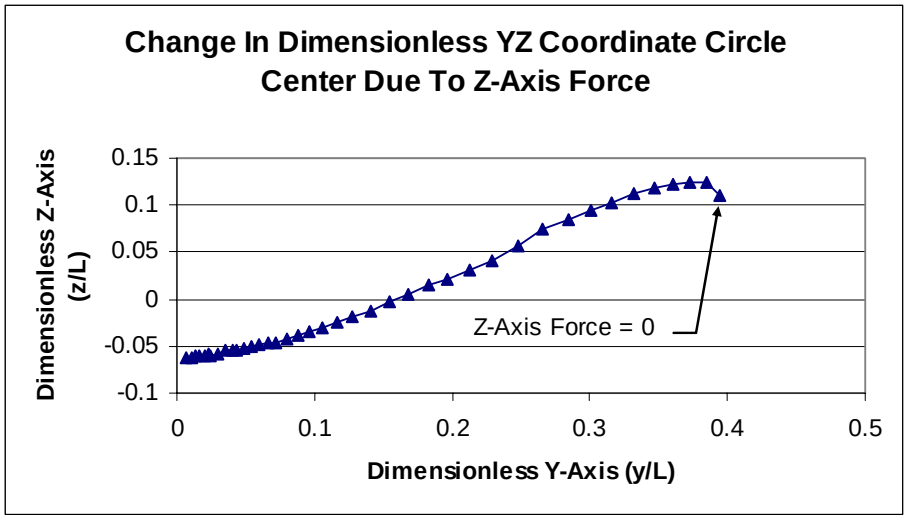


Figure 4.25: Shift in YZ circle center

In each graph an arrow points to the end of the graph where no Z-axis force is applied. The other end of the graph represents a Z-axis force causing a displacement in the Z-direction 80% of the beam length. The axes of these graphs are the nondimensional in-plane axes of the circles described in Figure 4.3.

From this data, polynomial functions can be formulated that describe the change in radii and circle center locations with respect to the Z-axis deflection. Once the radii are known, the problem can be solved as illustrated in section 4.5.

4.5 Case Study: Quarter Circle Mechanism

The mechanism shown in Figure 4.26 combines a parallel guiding mechanism with a lateral-torsional buckling cantilever beam. The function of this mechanism is to buckle the cantilever beam so the tip of the beam follows a completely circular path. The motion is terminated when the parallel guiding mechanism hits a stop. At this point the beam tip has drawn a quarter circle in the YZ plane and is at its maximum out-of-plane deflection.

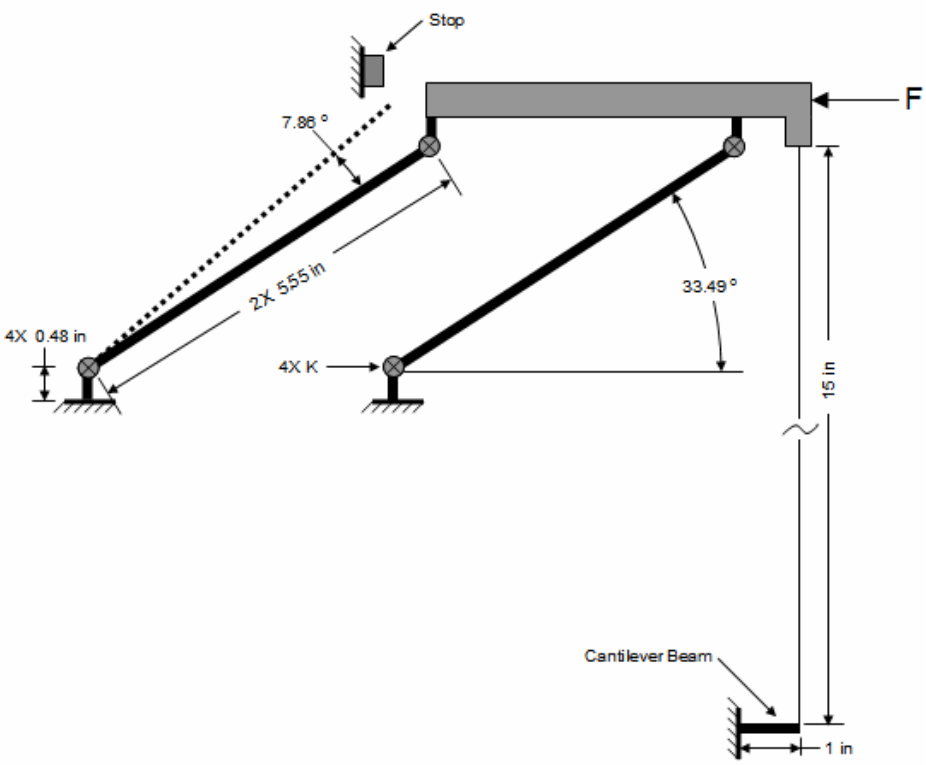


Figure 4.26: A parallel guiding mechanism combined with
a lateral torsional buckling cantilever beam

The parallel guiding mechanism is deflected 7.86 degrees by the force. The path of the parallel guiding mechanism is optimized to follow the lateral-torsional buckling beam path so the cantilever beam will not be subjected to the horizontal force. Because of the length of the very thin strip of material connecting the parallel guide to the cantilever beam, the cantilever beam will experience a nearly vertical force from the rising parallel guide. This vertical force causes the beam to buckle with the center of the beam tip drawing a perfect quarter circle in the YZ plane. There is a maximum force error of 1.53 degrees as the buckling beam deflects out of plane and the parallel guiding mechanism remains in plane. When the parallel guiding mechanism hits the stop, the cantilever beam is at its maximum out-of-plane deflection of 0.476 inches. The aspect ratio of the cantilever beam must be above 20. For this beam it is recommended to choose a cross-section of 0.09375 inches high by 0.003125 inches thick, which corresponds to an aspect ratio of 30. The beam geometry is small enough compared to the parallel guiding mechanism so that the force required to buckle the mechanism would be negligible.

MEMs are usually designed to operate in a 2-D plane. A mechanism similar to this one could be used as a switch to connect a lower layer to an upper layer. Or perhaps it could be used to assist a 2-D MEMs layout to pop up into a 3-D structure. One really useful aspect of this mechanism is that if the height and width measurements are off, the mechanism will still follow the same path.

4.6 Conclusion

Displacement analysis provides a way to better understand rotations and displacement paths of lateral torsionally buckled beams. For large aspect ratios, the change in beam-tip X, Y, and Z-axis orientations has very little variation due to different beam geometries and isotropic material properties. Similarly, the circular beam deflections in the XY and YZ planes vary only a little because of the same properties. These discoveries will lead to ideas and inventions. They may also fuel more research into large deflections of lateral torsional buckling.

Chapter 5

3-D Chain Algorithm with Pseudo-Rigid-Body Model Elements

5.1 Introduction

The various displacements caused by lateral torsional buckling do not lend themselves to a pseudo-rigid-body model with a single link. Angular deflections vary drastically along the length of the beam, as do the X, Y, and Z displacements. Every part of the beam reacts to applied moments and loads according to how it is oriented. The orientation at a particular location along the beam length depends on the orientation of the part of the beam immediately behind it or slightly closer to the beam root. If each infinitely small segment of a prismatic, isotropic beam were subjected to the same force and moment load, they would all displace exactly the same. Therefore, the deflection at any point along the length of the beam can be represented by the same beam segment.

5.2 Governing Phenomena

Three phenomena govern these infinitely small segments, shown in Figure 5.1. The first is twist along the beam length. The second is bending in the slender direction. The third is bending in the stiff direction.

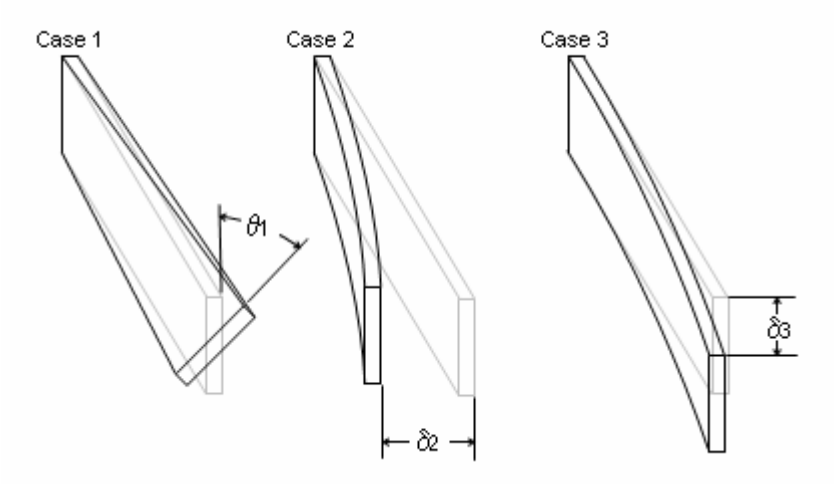


Figure 5.1: Three deflection components of lateral torsional buckling
Case 1: Twisting along the beam length
Case 2: Bending in the slender direction
Case 3: Bending in the stiff direction

The first two phenomena are responsible for nearly all of the displacement in the beam unless the cross-sectional aspect ratio is low. High aspect ratios would require an extremely large force to substantially deflect the beam in its stiff direction. Of course, before this deflection ever took place, the beam would buckle and twist. For this reason, deflection of the stiffer side of the beam can often be ignored with very small error. Hence, Case 3 can be neglected. At low aspect ratios these three phenomena are more evenly matched and all three must be taken into account.

If a rectangular beam subjected to lateral torsional buckling were cut through its cross-section at any point along its length, these three deflections would represent all possible displacements with regards to this buckling phenomenon. If a small enough beam segment were cut from the beam, its displacements could be very accurately predicted based on the forces and moments felt at the tip end of each segment.

5.3 Beam Segment Deflections

The three beam deflections described in section 5.2 can be modeled with simple two dimensional pseudo-rigid-body models. Consider Case 2 and Case 3 in Figure 5.1. The second is the beam bending in the slender direction and the third is the beam bending in

the stiff direction. These are the same phenomena with different stiffnesses. Now consider the classic two-dimensional cantilever beam, shown in Figure 5.2.

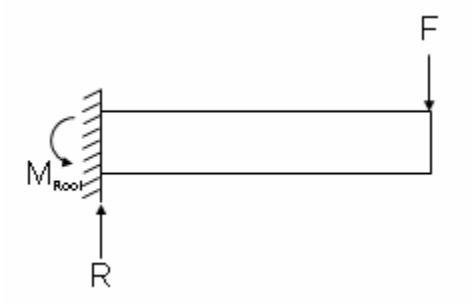


Figure 5.2: The classic cantilever beam

One end is fixed while the other end is free. The free end has a vertical force applied to it. The reaction caused by the vertical force F at the end of the beam is an opposing force and moment located at the beam root. The reaction force is felt as a constant shear force (V) through the beam length (see Figure 5.3).

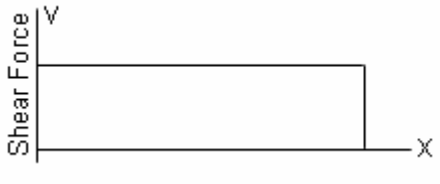


Figure 5.3: Shear force diagram

As is done in classical beam theory, the shear force is neglected because the beam deflection due to shear is very small compared to the beam deflection caused by the bending moment in the beam. The moment at the beam tip is zero and increases linearly along the beam length, reaching a maximum magnitude at the beam's root (see Figure 5.4).

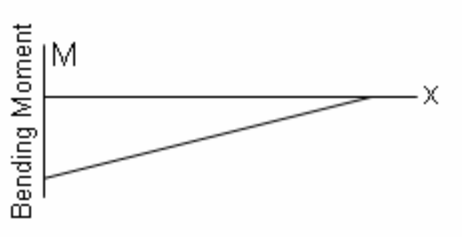
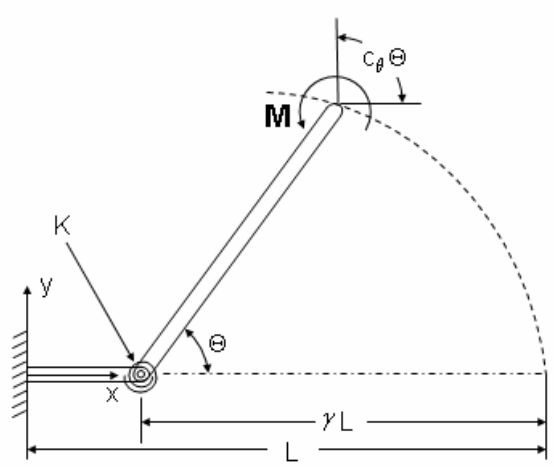


Figure 5.4: Moment load diagram

Now a small segment is cut from the beam and the forces and moments are analyzed. At the beam segment tip is the applied load and a moment representing the force applied at the beam tip multiplied by the distance to the beam tip. At the beam segment root is the reaction force along with a moment representing the force applied at the beam tip multiplied by the sum of the distance to the beam tip and the length of the beam segment.

If the beam segment is now shrunk in length to an infinitely small length, the force and the reaction force converge to the same vertical plane and an assumption can be made that they cancel out. The moments also converge together. A pseudo-rigid-body model for a cantilever beam with a moment at the free end is now applied to the infinitely short beam as shown in Figure 5.5.

The torsional spring in Figure 5.5 representing the deflection of the beam segment is placed near the beam segment root. Its distance from the beam segment tip is 0.7346 multiplied by the beam segment length. As the linkage deflects from the applied moment, the actual beam tip angle is represented by the linkage angle multiplied by 1.5164. The torsional spring constant is 1.5164 multiplied by $\frac{EI}{L}$. This pseudo-rigid-body model approximates a pure moment in a two-dimensional cantilever beam with less than 0.5% error over a range of 124.4 degrees from the undeflected position. [1]



Characteristic Radius Factor: $\gamma = 0.7346$

Parametric Angle Coefficient: $c_\theta = 1.5164$

Spring Constant: $K = \gamma K_\theta EI/L$

Stiffness Coefficient: $K_\theta = 2.0643$

Figure 5.5: Components of the end moment loaded pseudo-rigid-body model

The second beam deflection is twisted around the centroidal axis. The angular deflection for this element is calculated with the following equation [1],

$$\theta = \frac{TL}{KG} \tag{5.1}$$

where T is the torque, L is the length of the beam, K is the torsional spring constant and G is the shear modulus. The torsional spring constant, K, for this element is

$$K = bh^3\left[\frac{1}{3} - 0.21\frac{h}{b}\left(1 - \frac{h^4}{12b^4}\right)\right], \text{ where } b \geq h \qquad [1] \tag{5.2}$$

Here, b represents the height of the beam and h represents the width. This spring can be placed anywhere along the length of the beam segment. For convenience it will be

placed at the same location as the two torsional springs previously discussed. The error associated with the torsional spring constant *K* is no greater than 4% [10].

5.4 Superposition

Up to this point a cantilever beam undergoing lateral torsional buckling has been broken down into beam segments. The three governing phenomena that describe the displacements associated with this type of buckling have been explained. Each phenomena has been modeled as a pseudo-rigid-body model or two links connected with a torsional spring.

These three models can now be superimposed on each other (see Figure 5.6.). Each beam segment contains the three models with the three springs placed at exactly the same point. The three springs are positioned orthogonal to each other so that if a torque were applied in the plane of one spring, the other two springs would not feel it. The spring characterizing bending in the stiffer direction of the beam corresponds to the Y-direction. The spring characterizing bending in the slender direction corresponds to the Z-direction. The final spring describes axial torsion along the X-axis.

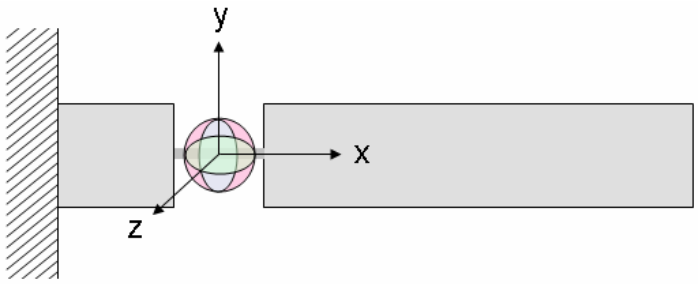


Figure 5.6: Three models superimposed on each other

5.5 3-D Chain Algorithm

Before combining the new 3-D pseudo-rigid-body model linkage with other like linkages, two differences between this element and 2-D chain elements should be pointed out.

First, the 3-D pseudo-rigid-body linkage deflects in 3-D. Displacement positions of each node will have an X, Y, and Z component. Also, where a two-dimensional chain of elements would require a single axis of rotation along with one spring, lateral torsional buckling requires three springs with three axes. In two-dimensional algorithms, sine and cosine operators can be used. The 3-D linkage must be represented with vectors and the spring axes are represented as a group by a local coordinate system. When the element rotates around one of the three spring axes characterized by a unit vector, the other two spring vectors must rotate around the first vector in the same direction and with the same amount. This can be accomplished with the following formula [13], which rotates a vector around another vector or axis.

$$\vec{v}' = \vec{v}\cos\varphi + (1-\cos\varphi)(\vec{v}\cdot\hat{w})\hat{w} - \vec{v}\times\hat{w}\sin\varphi \tag{5.3}$$

With this equation the vector v is rotated counterclockwise around the axis w by an angle φ, as shown and derived in Appendix D. The length of the vector is preserved.

Second, the 3-D chain cantilever is indeterminate. In a 2-D chain, each segment would be attached to a node with one axis of rotation. As the segment rotates along that axis, a circle is drawn with a radius equal to the length of the segment. In the 3-D chain cantilever only two of the three degrees of freedom are required for the free segment to touch any point on a sphere with a radius equal to the length of the free segment.

Now the 3-D chain cantilevers can be placed end to end to form a chain (see Figure 5.7). Every node will be placed in 3-D space with respect to the global X-Y-Z coordinate system. Each node will have its own local coordinate system starting with X1-Y1-Z1 and ending with Xn-Yn-Zn where n represents the last node in the final 3-D chain cantilever.

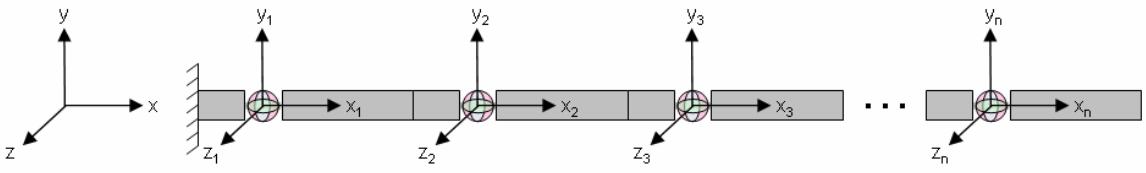


Figure 5.7: 3-D chain of elements

A force in the negative Y-direction is now placed at the tip of the segmented beam for convenience in explaining how the 3-D chain algorithm is analyzed (see Figure 5.8).

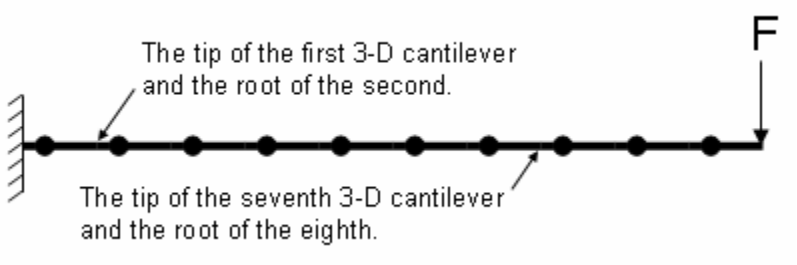


Figure 5.8: 3-D chain algorithm

Before the moments can be found, the location of each 3-D element tip and the location of the acting force must be calculated. At the beginning the initial orientation of each element will be defined and then each element will be aligned root to tip down the chain. The equations used to find the location of each beam tip are described as follows:

$$\vec{P}_i = \begin{bmatrix} P_{x_i} \\ P_{y_i} \\ P_{z_i} \end{bmatrix} = \begin{bmatrix} P_{x_{i-1}} + ux_{x_{i-1}} S\_L(1-0.7346) + ux_{x_i} S\_L(0.7346) \\ P_{y_{i-1}} + ux_{y_{i-1}} S\_L(1-0.7346) + ux_{y_i} S\_L(0.7346) \\ P_{z_{i-1}} + ux_{z_{i-1}} S\_L(1-0.7346) + ux_{z_i} S\_L(0.7346) \end{bmatrix} \tag{5.4}$$

$$\vec{u}x_i = \begin{bmatrix} ux_{x_i} \\ ux_{y_i} \\ ux_{z_i} \end{bmatrix} \tag{5.5}$$

$\vec{P}_i$ represents the current location of each beam tip, and $\vec{u}x_i$ is a unit vector and part of the local coordinate system for each node that points in the axial direction of each 3-D cantilever. It should be noted that $\vec{u}x_0$ represents a unit vector in the global coordinate

system and is positioned at

$$\vec{P}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.6}$$

This unit vector of the previous 3-D cantilever is scaled by the segment length (S_L), which for this model is constant. S_L represents the sum of the length of the beam on the left side of the node in Figure 5.9 and the length on the right side. By finding the beam tips of each 3-D cantilever, the position of the force is also known. It is located on the tip of the 3-D cantilever $\vec{P}_n$.

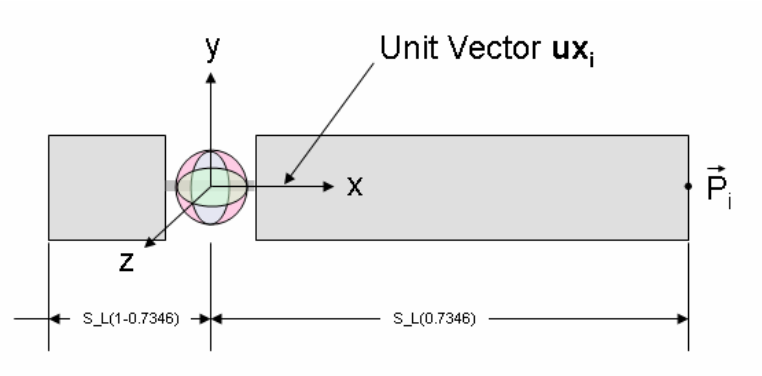


Figure 5.9: 3-D element coordinate system

The moment can now be calculated for node 1 using the moment equation:

$$\vec{M} = \vec{P}_1 \times \vec{F} \tag{5.7}$$

The moment $\vec{M}$ is now broken down into three components with respect to the local coordinate system of node 1.

$$\begin{aligned} \vec{M}_{x_1} &= \vec{M} \bullet \vec{u}x \\ \vec{M}_{y_1} &= \vec{M} \bullet \vec{u}y \\ \vec{M}_{z_1} &= \vec{M} \bullet \vec{u}z \end{aligned} \tag{5.8}$$

Next the rotation of the spring whose axis lies on the local coordinate system unit vector $\vec{u}x$ is found.

$$\varphi_{x_1} = \frac{M_{x_1} L}{K_t G} - \varphi_{x_1 Last} \tag{5.9}$$

In this equation K_t represents the torsional spring constant calculated in section 5.3. The variable $\varphi_{x_1 Last}$ represents the deflection of that same spring for its previous iteration. If this was the first time this calculation was performed then $\varphi_{x_1 Last} = 0$.

The rotation of the X-axis spring of the local coordinate system for node 1 requires that the Y and Z axes for this local coordinate system also rotate around the X-axis the same amount and in the same direction. This is done with the vector rotation formula introduced at the beginning of section 5.5. In this case, the unit vector $\vec{u}x_1$ is the axis of rotation. The vectors $\vec{u}y_1$ and $\vec{u}z_1$ are rotated around $\vec{u}x_1$ by an angle φ_{x_1}. Now all unit vectors in local coordinate systems at each node with a subscript higher than that of the current node must rotate by the angle φ_{x_1} around the axis $\vec{u}x_1$.

Next, the rotation of the spring associated with the local unit vector $\vec{u}y_1$ is calculated.

$$\theta 1_{y_1} = \frac{M_{y_1}}{K_y} - \theta 1_{y_1 Last} \tag{5.10}$$

Similar to what was done earlier, the unit vectors $\vec{u}x_1$ and $\vec{u}z_1$ at that same node are then rotated around axis $\vec{u}y_1$ by an angle $\theta 1_{y_1}$. However, all unit vectors in local coordinate systems at each node with a subscript higher than that of the current node do not rotate by the angle $\theta 1_{y_1}$ although they do rotate around $\vec{u}y_1$. If the end angle of the actual end moment loaded beam were compared to the end angle of the end moment loaded pseudo-rigid-body model, it would be evident that the difference between these angles grows as the beam displacement increases. This comparison can easily be seen in Figure 5.10.

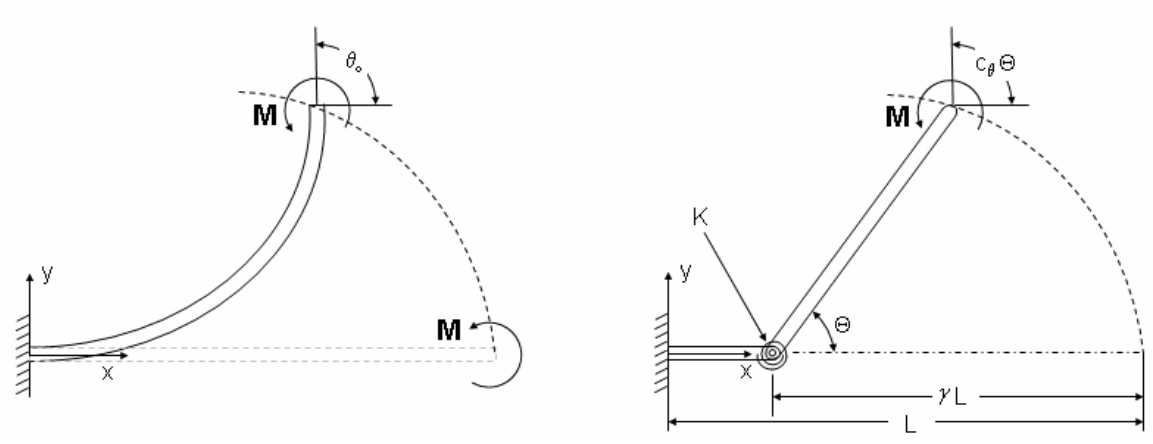


Figure 5.10: Actual beam tip angle compared to pseudo-rigid-body model tip angle

As noted in section 5.3, the correction factor or parametric angle coefficient is 1.5164. Before the unit vectors at nodes with a subscript higher than the current subscript can be rotated, this parametric angle coefficient must be multiplied by $\theta 1_{y_1}$.

Finally, the last spring can be analyzed in the same way as the $\vec{u}y_1$ spring.

$$\theta 2_{z_1} = \frac{M_{z_1}}{K_z} - \theta 2_{z_1 Last} \tag{5.11}$$

The angle $\theta 2_{z_1}$ is found, followed by the rotation of the $\vec{u}x_1$ and $\vec{u}y_1$ around the axis $\vec{u}z_1$. The same parametric angle coefficient is then multiplied by the angle $\theta 2_{z_1}$ and all unit vectors for nodes with a subscript higher than the current subscript are rotated around $\vec{u}z_1$ by an angle $1.5164\theta 2_{z_1}$. This process continues by moving from node 2 to the final node n. All nodes with a number lower than the current node number being analyzed are held rigid and before any node can be analyzed, the positions of each node that has been modified must be found.

This iterative process is repeated over and over. After each run of analyzing the first node to the last node, the distance between a particular node's old position and its new position should converge to zero. When enough decimal places of accuracy have been achieved the process can be terminated and the approximation can be considered good enough.

If a substantial force or moment is to be placed on the beam, it may be important to begin small and add small increments until the desired load has been reached. A force or moment that is too large will cause the algorithm to go unstable and not converge to a solution.

In the simple tip-loaded cantilever case used to describe the chain algorithm process, it should be noted that the moment on the last 3-D cantilever would be zero because the distance from the beam tip to the applied force is zero. This illustrates how the 3-D chain algorithm approximates a continuous process by digitizing the moment as shown in Figure 5.11.

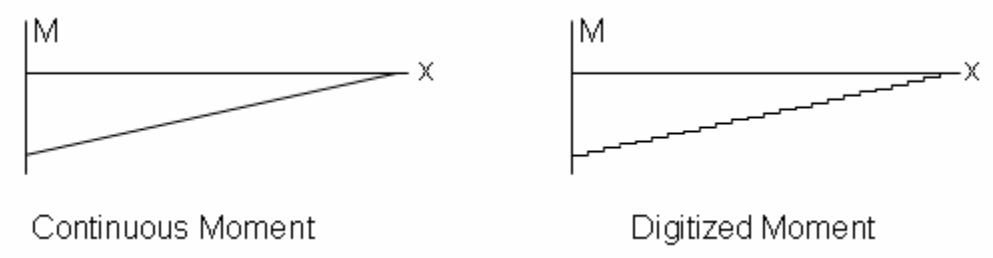


Figure 5.11: Digitization of moment load

This 3-D chain algorithm is useful in applications where applied forces and moments are known. It is robust and can converge to solutions with fairly large load increments. For smoother plots smaller load increments should be used. Better approximations of real phenomena can be achieved by using more segments. Multiple forces and moments can be placed at any node along the chain to simulate different or very complex displacements. After loads have been applied, the chain should be allowed to settle within a specified number of digits of precision. During the buckling process, this may require more iterations while at other times fewer iterations are needed. Displacement loads could be achieved by adding iterative processes that included optimization techniques. However, these types of processes could greatly increase the time to achieve a suitable answer, making a displacement load chain algorithm very inefficient compared to other methods.

5.6 Ansys Compared with 3-D Chain Algorithm

In this comparison, Ansys and the 3-D chain algorithm were used to analyze the lateral torsional buckling behavior of the same beam. The 3-D deflection paths from this bucking process were previously modeled in Ansys and validated in section 4.2. The beam is 100 inches long, 5 inches high, and 0.125 inches wide. This gave it an aspect ratio of 40. It was isotropic and has material properties approximating polypropylene. Young's modulus was 200,000 psi and Poisson's ratio was 0.34. Both Ansys and the 3-D chain Algorithm used 200 elements. A small perturbation force was applied in the Z-direction with a magnitude of 0.000738 lbs. A Y-direction force was applied in two-hundred increments. After 200 increments the beam deflected in the Y-direction 80% of the beam length. Ansys used equal displacement loads while the chain algorithm added equal force load increments, and they both finished with the same 3-D displacement. The deflection path comparisons between Ansys and the 3-D chain algorithm are shown in Figures 5.12 and 5.13. Figure 5.14 shows the force-deflection comparison.

In the XY plane, there was a 0.24% difference between Ansys and the chain algorithm with respect to the Y-deflection found after the beams had displaced 40% of their length. In the YZ plane there was a 0.37% difference with respect to the Y-deflection after deflecting 40% of the length.

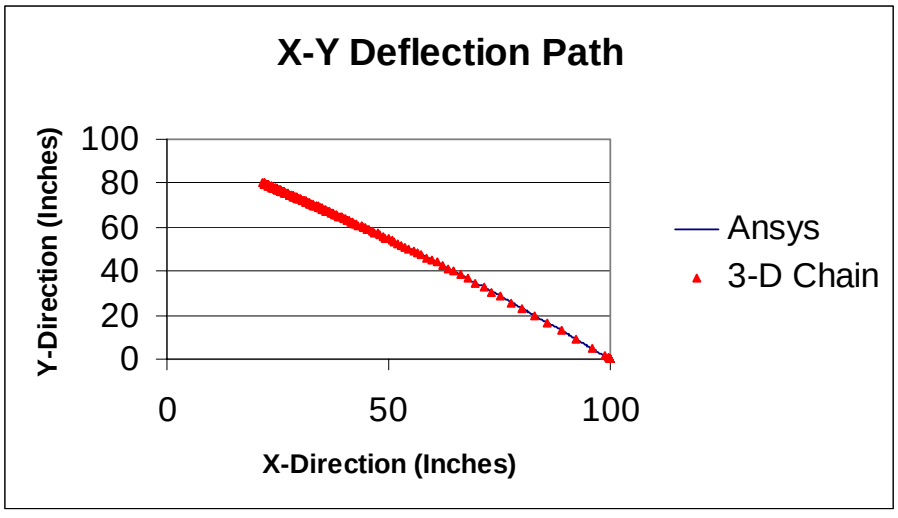


Figure 5.12: XY plane beam tip path

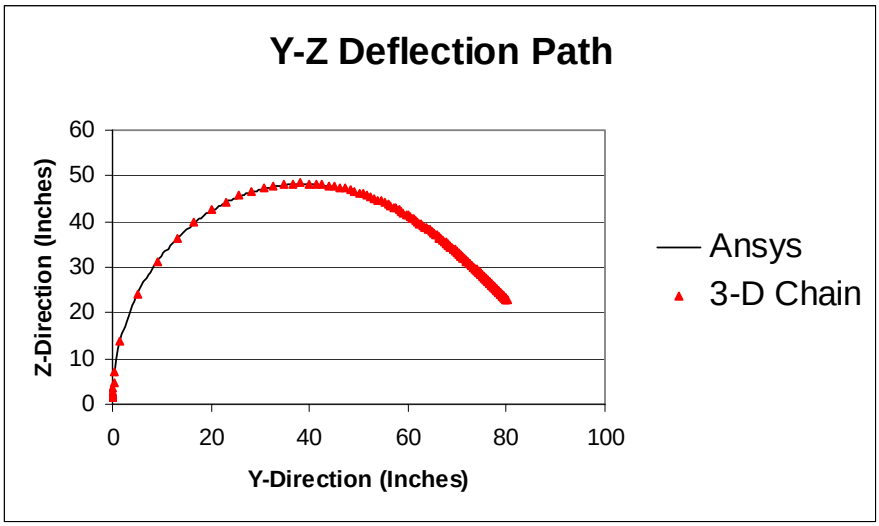


Figure 5.13: YZ plane beam tip path

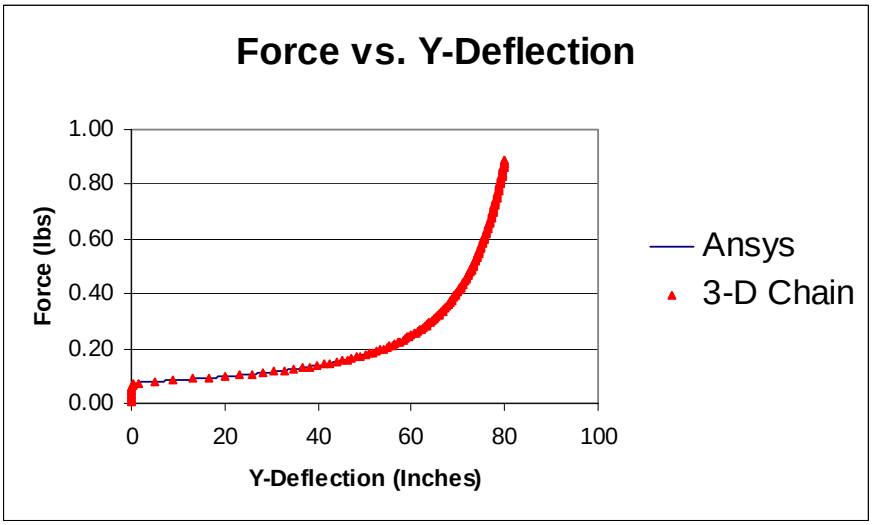


Figure 5.14: Relationship between force and Y-deflection

The force was found to have a 0.0008% difference with respect to the Y-deflection after the beam had deflected 40% of its length in the Y-direction.

These differences were calculated with the following equations:

$$\frac{error}{\delta_y} = \frac{\left|x_{Ansys} - x_{Chain}\right|}{\delta_y} \tag{5.12}$$

$$\frac{error}{\delta_y} = \frac{\left|z_{Ansys} - z_{Chain}\right|}{\delta_y} \tag{5.13}$$

$$\frac{error}{\delta_y} = \frac{\left|F_{Ansys} - F_{Chain}\right|}{\delta_y} \tag{5.14}$$

Also, it should be noted that Ansys required about 75 minutes on the BYU supercomputer while the chain algorithm required only 18 minutes on a laptop to complete the same number of steps. This illustrates the accuracy and efficiency of the 3-D chain algorithm.

5.7 Method for Understanding Beam Rotations

The prevailing technique for understanding rotations consists of using a rotation matrix in one of two forms [14].

$$M(\alpha,\beta,\gamma) = \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma + \cos\alpha\sin\gamma & -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ -\cos\beta\sin\gamma & -\sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma + \sin\alpha\cos\gamma \\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix} \tag{5.15}$$

or,

$$M(\hat{\mathbf{v}},\theta) = \begin{bmatrix} \cos\theta + (1-\cos\theta)x^2 & (1-\cos\theta)xy - (\sin\theta)z & (1-\cos\theta)xz + (\sin\theta)y \\ (1-\cos\theta)yx + (\sin\theta)z & \cos\theta + (1-\cos\theta)y^2 & (1-\cos\theta)yz - (\sin\theta)x \\ (1-\cos\theta)zx - (\sin\theta)y & (1-\cos\theta)zy + (\sin\theta)x & \cos\theta + (1-\cos\theta)z^2 \end{bmatrix} \tag{5.16}$$

The first form uses Euler angles alpha, beta, and gamma. The second form uses a single direction vector $\hat{v}$ or axis and one angle of rotation. The 3-D chain algorithm uses the second form in a vector format. This technique is useful for computers to describe orientations. However it can be much more difficult for a designer to visualize the different orientations of a twisted surface and design things with respect to it. For this reason engineers often prefer to stay in the simpler realm of 2-D designs.

An alternative approach that may be easier to visualize requires a reference plane with a direction vector on that plane. The object to be rotated is also given a plane with a direction vector. The object is then positioned so its plane is parallel to the reference

plane and its direction vector is pointing in the same direction as the reference direction, as shown in Figure 5.15.

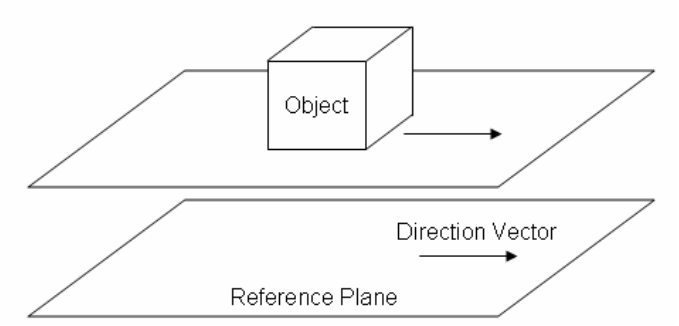


Figure 5.15: Parallel planes with direction vectors pointing in the same direction

The object then undergoes its 3-D rotation by means of a rotation matrix. Finally, two angles are given that wholly describe the rotation of the object with respect to the reference plane and reference vector.

As shown in Figure 5.16, the first angle represents the angle between the two planes. The second angle represents the angle between the projection of the object's direction vector on the reference plane and the reference plane's direction vector. The second angle also positions the first angle so it lies in a plane that is orthogonal to the reference plane and the object's plane. The first angle is equal to the angle found between the normal vectors of the planes, as shown in Figure 5.17.

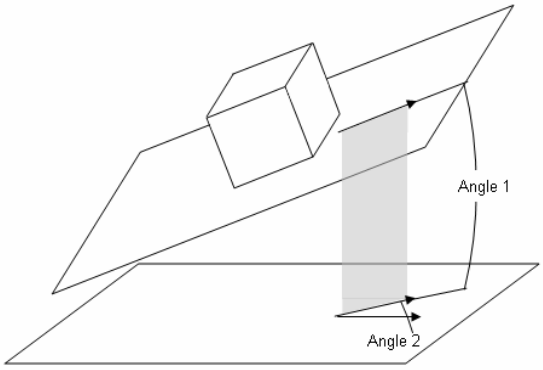


Figure 5.16: Orientation described by two angles

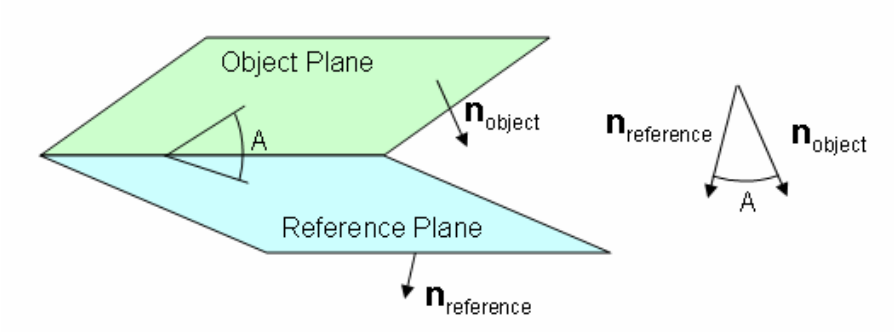


Figure 5.17: Normal vectors used to find the angle between two planes

This lends itself well to the 3-D chain algorithm because local coordinate systems are known at each node. For example, these coordinate systems can be easily viewed as an XY plane and a normal vector Z or any other combination. In the case of the XY plane, the X or the Y axis could be used as the direction vector. A reference plane and a reference normal vector are also easy to establish based on the chain algorithm's global coordinate system. To find the angle between the two normal vectors, slide the tails of the two vectors together, as shown in Figure 5.17, and utilize the following dot product equation:

$$\vec{n}_{ref} \bullet \vec{n}_{obj} = \left|\vec{n}_{ref}\right|\left|\vec{n}_{obj}\right|\cos A \tag{5.17}$$

Because the normal vectors are chosen from coordinate systems consisting of unit vectors the norms of each vector equal one as shown in equation 5.18.

$$\left|\vec{n}_{ref}\right| = \left|\vec{n}_{obj}\right| = 1 \tag{5.18}$$

The equation is simplified as follows:

$$\vec{n}_{ref} \bullet \vec{n}_{obj} = \cos A \tag{5.19}$$

This can be rewritten as

$$\arccos\left(\vec{n}_{ref} \bullet \vec{n}_{obj}\right) = A \tag{5.20}$$

And the first angle is found.

As was previously explained, the second angle represents the projection of the object's direction vector onto the reference plane. In other words, the object's 3-D direction vector not planar to the reference plane is thrown out. Then, on the 2-D plane, the tails of the projected vector and the reference direction vector are placed end to end. The angle between them is found as follows. Assuming the direction vector on the reference plane points in the X-direction and the in-plane unit vector orthogonal to the direction vector is the Y-axis, then the projection can be illustrated as shown in Figure 5.18.

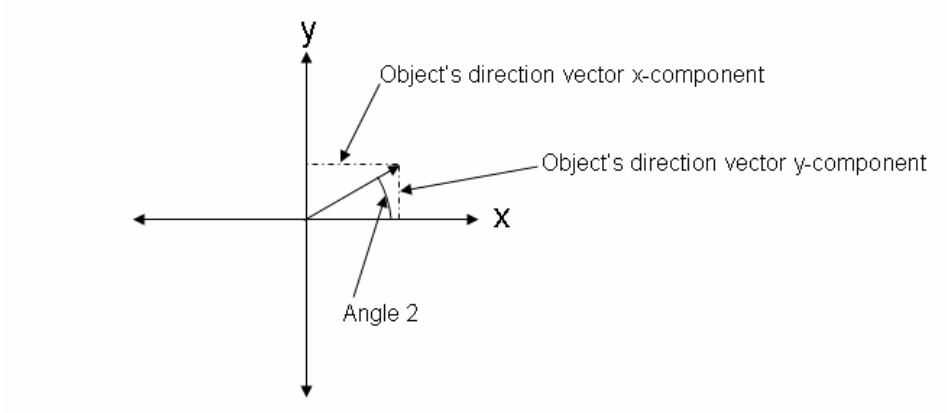


Figure 5.18: Projection of object's direction vector on reference plane

The second angle is solved by the following equation:

$$Angle\_2 = \arctan\left(\frac{n_{obj_y}}{n_{obj_x}}\right) \tag{5.21}$$

Understanding the rotations of a node along a beam can be understood more simply by breaking down the 3-D rotations into two 2-D angles referenced to a known plane and direction. This will be illustrated in a case study following this section.

5.8 Stress Analysis of the 3-D Chain Algorithm

Although this thesis's focus is on force and deflection, a summary of how maximum stresses can be found along the beam would be useful and simple. For a 2-D pseudo-rigid-body model of an end-moment loaded cantilever beam, the maximum stress is found at the beam root by the following equation:

$$\sigma_{\max} = \frac{M_0 c}{I} \tag{5.22}$$

Here, M_0 is the moment felt at the beam root and c is the distance from the neutral axis to the outer surface of the beam [1].

For a rectangular beam in pure torsion the following equation [1] can be applied to find the maximum shear stress:

$$\tau_{\max} = \frac{3T}{bh^2}\left[1 + 0.6095\frac{h}{b} + 0.8865\left(\frac{h}{b}\right)^2 - 1.8023\left(\frac{h}{b}\right)^3 + 0.9100\left(\frac{h}{b}\right)^4\right] \tag{5.23}$$

where $b \geq h$

Using combined loading techniques, two forms of the end-moment loaded cantilever beam and the pure torsion model can be combined to find the locations of maximum stress along the beam.

5.9 Case Study: Comparing a Torsional Micromirror with a Lateral Torsional Buckling Micromirror Mechanism [1]

One possible application of lateral torsional buckling would be in creating a new type of micromirror that could be used in projectors. Current projectors use a torsional micromirror as shown in Figure 5.19.

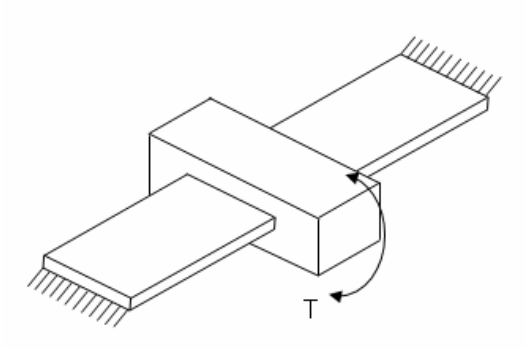


Figure 5.19: Torsional micromirror

In Figure 5.19 a mirror is placed on top of the middle platform. The mirror oscillates, reflecting light from a color wheel through lenses onto a white screen where the reflected color becomes a single pixel. The oscillation is caused by a magnetic field that causes the torsional beam, which is positioned under the mirror and fixed at both ends, to twist along its axis.

Though much design work and research would need to take place to come up with a working lateral torsional buckling micromirror, a simple schematic with the required components is shown in Figure 5.20.

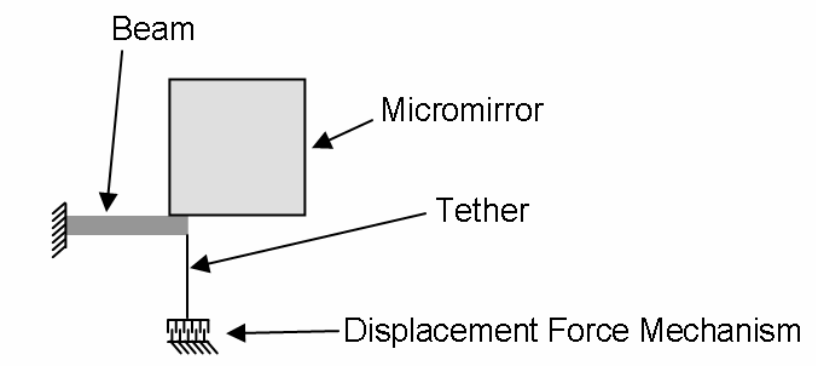


Figure 5.20: Lateral torsional buckling micromirror

In this lateral torsional buckling micromirror arrangement, the displacement force mechanism causes the beam to deflect. The beam's deflection in its stiffest direction causes it to buckle. The buckling action causes the mirror to rotate with respect to the

wafer surface. In this case Figure 5.20 shows the mirror lying flat against the wafer surface. This mirror would then be placed in an array of mirrors as shown in Figure 5.21, below.

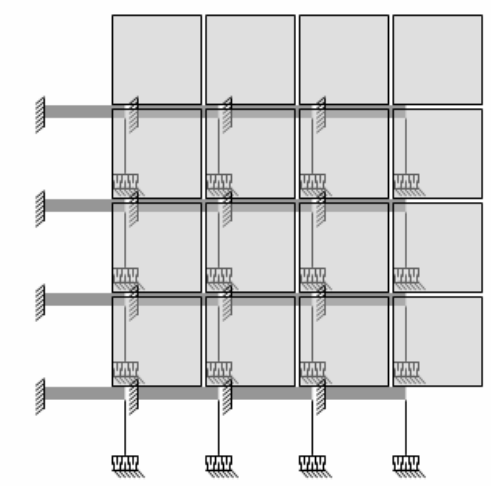


Figure 5.21: Array of lateral torsional buckling micromirrors

This design can be viewed as a first step. It also brings out challenges in creating a lateral torsional buckling micromirror. For example, if the mirror were to curl slightly, how could it be attached to the beam so the beam didn't curl as well? Would the weight of the mirror cause an unwanted moment on the beam? How could the components be attached to each other so the mirrors could be placed close together with no obstructions? Since these mirrors oscillate many times each second, what are the dynamic effects on this mechanism? These problems will be neglected in this study.

For a first look at this mechanism the force applied to the beam will be assumed to be slow, and the potential energy required to rotate the mirror 10 degrees will be found for both the torsional micromirror and the lateral torsional buckling micromirror. After a comparison is made with the traditional approach, conclusions will be drawn. In addition to this, the proposed method for understanding beam rotations will be utilized.

(A) Torsional Micromirror

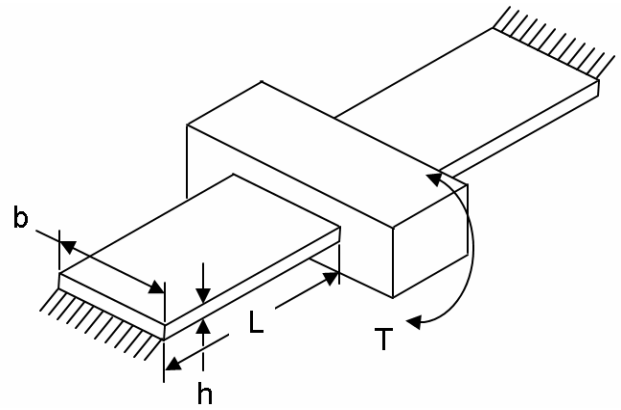


Figure 5.22: Torsional micromirror dimensions

For a torsional micromirror (shown in Figure 5.22), "the torsional hinges have a rectangular cross section because of fabrication constraints at the micro level. Suppose that the cross-sectional area is 0.2 by 1.2 μm, and that each hinge is 4 μm long and is made of sputtered aluminum. Calculate the torque required to deflect the mirror 10°" [1]. Next, calculate the potential energy stored in the beam at the 10° deflection.

Solution:

The torque can be found with the following equation:

$$T = \frac{KG}{L}\theta \tag{5.24}$$

For a rectangular cross-section,

$$K = bh^3\left[\frac{1}{3} - 0.21\frac{h}{b}\left(1 - \frac{h^4}{12b^4}\right)\right] \quad \text{where } b \geq h \tag{5.25}$$

For sputtered aluminum the following material properties will be used.

- E = 71.7 GPa

- v = 0.34
- G = 26.8 GPa (Calculated from E and v)

For the given geometry, K = 0.0029 μm^4 and the torque is calculated as follows:

$$T = \frac{\left(0.0029\mu m^4\right)\left(26.8GPa\right)}{4\mu m}\left(0.175rad\right) = 3.4\times10^{-12}\ N\cdot m \tag{5.26}$$

Because the torque increases linearly with the angle θ, the following equation can be used to approximate the torque as a linear spring with a torsional spring constant k_t:

$$k_t = \frac{T}{\theta} = \frac{3.4\times10^{-12}\ N\cdot m}{0.175rad} = 1.94\times10^{-11}\ \frac{N\cdot m}{rad} \tag{5.27}$$

Finally the potential energy is found with the following equation:

$$PE = \frac{1}{2}k_t\theta^2 = \frac{1}{2}\left(1.94\times10^{-11}\ \frac{N\cdot m}{rad}\right)\left(0.175rad\right) = 0.296\times10^{-12}\ J \tag{5.28}$$

The potential energy must then be multiplied by 2 because the torque described by Equation 5.26 analyzes only one side of the fixed-fixed beam. Hence, the total torque to rotate the beam 10 degrees is,

$$2\cdot0.296\times10^{-12}\ J = 0.592\times10 = ^{-12}\ J \tag{5.29}$$

(B) Lateral Torsional Buckling Micromirror

The same material properties and dimensions will now be used for a lateral torsional buckling micromirror, except that the length is 8 μm. In this case the spring constant does not remain constant. As the beam buckles the spring constant grows. The chain algorithm will start with a positive 0.25 μN force in the Y-direction and a 1.0 nN perturbation force

in the positive Z-direction. The $0.05\,\mu N$ force is an iterative value. After the chain algorithm has converged to a solution within ten decimal places of accuracy, $0.05\,\mu N$ will be added to the Y-direction force and the chain algorithm will again converge to the solution. The program will iterate one hundred times. Each segment of the beam is $0.08\,\mu m$ long and there are one hundred segments.

The buckling beam is modeled as a linear spring whose spring constant is continually changing with each added force iteration. The spring constant at each load step is calculated with the following equation:

$$k = \frac{F}{\Delta x} \tag{5.30}$$

The potential energy added to the beam with each load increment is expressed by the following equation:

$$PE = \frac{1}{2} k(\Delta x)^2 \tag{5.31}$$

The potential energy from each load increment is then summed up to find the total potential energy in the beam. The method described in section 5.7 is used to calculate when the beam tip has rotated 10 degrees. The summation of the potential energy is described as follows:

$$PE = \frac{1}{2} \sum_{i=1}^{100} k_i (\Delta x)_i = 0.092 \times 10^{-12}\, J \tag{5.32}$$

As shown from this case study lateral torsional buckling requires about five times less energy to deflect 10 degrees than the fixed-fixed torsional beam. In a dynamic setting, on a silicone wafer, this implies that a lateral torsional buckling micromirror array would not heat up as much as a torsional micromirror array.

In actual fact, micromirrors rotate only a fraction of a degree. The following graph compares the energy requirement for both beam configurations from 0 to 10 degrees.

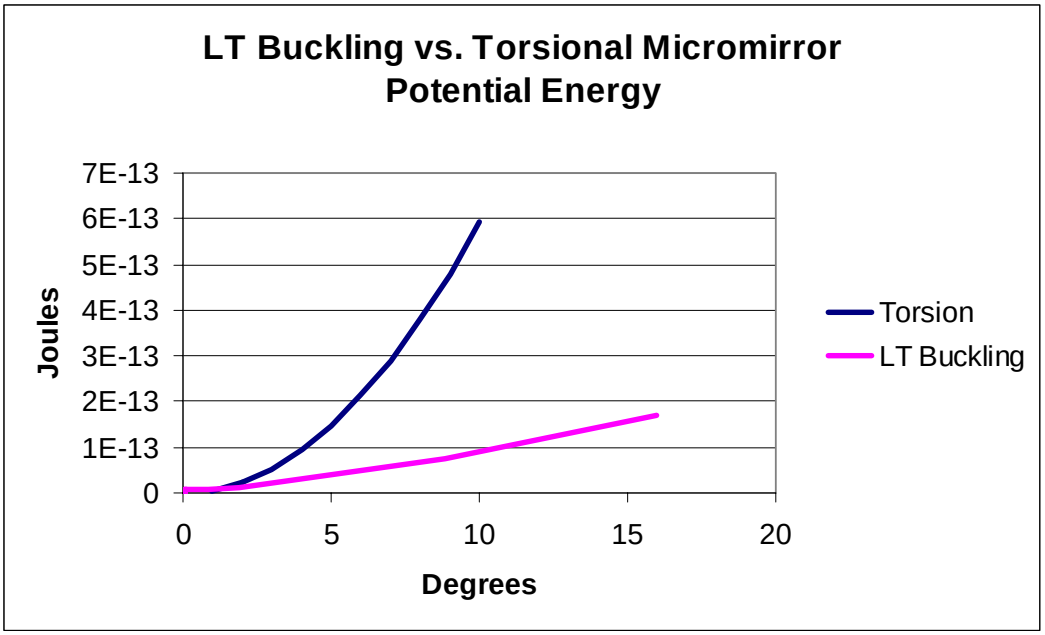


Figure 5.23: Potential energy comparison

As shown in the graph, lateral torsional buckling is more efficient above approximately 1 degree. The buckling beam is less efficient below 1 degree because the force at the beam tip must reach a required preload before the buckling phenomenon occurs.

This case study used beam geometry optimized for the torsional micromirror mechanism. A lateral torsional buckling beam optimized for maximum tip rotation with minimal force and displacement would consist of a stubby beam with an aspect ratio much higher than that used in this case study. The thickness of the beam would be minimized. Figure 5.24 illustrates how the plot would change below 1 degree if the thickness of the beam being analyzed for both the torsional micromirror and the lateral torsional buckling micromirror was divided by 4 making the aspect ratio jump from 6 to 24.

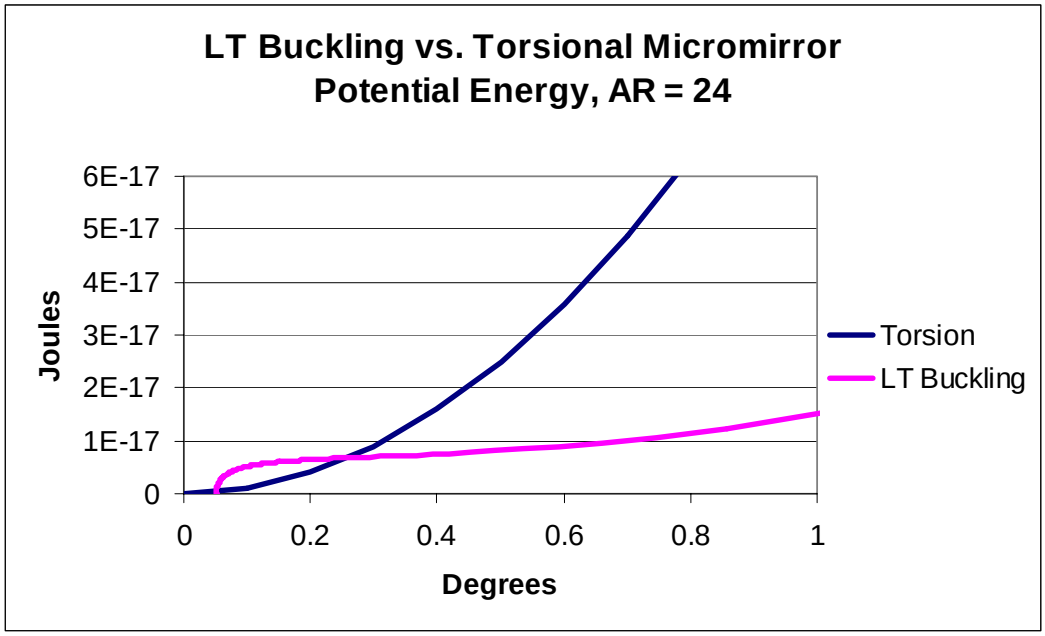


Figure 5.24: Potential energy comparison at a higher aspect ratio

As can be seen from the above plot, changing the design in favor of lateral torsional buckling makes lateral torsional buckling more efficient from a little over 0.2 degrees up to 1 degree. Changing the design parameters changes which micromirror beam configuration has the upper hand on efficiency.

5.10 Conclusion

The 3-D chain algorithm described in this chapter has great potential because it models large 3-D beam displacements accurately and in many applications faster than the finite element method. The rigid links and springs that make up the elements of the chain make the force deflection relationship and displacements at each node more understandable. The use of pseudo-rigid-body model elements greatly simplifies the 3-D chain algorithm, allowing it to be programmed in a few pages of code (see Appendix E). Since the three pseudo-rigid-body models act orthogonally to each other, no interactions between models need to be addressed. Each model brings to the element excellent accuracy and in the case of the end moment loaded pseudo-rigid-body models, large deflection potential.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions and Contributions

The purpose of this thesis has been to define the special deflection properties of lateral torsional buckling and to create a 3-D chain algorithm with pseudo-rigid-body model elements that accurately predicts 3-D force and large deflections of beams. The displacement analysis approach used best-fit circles to match displacements described in the XY and YZ planes. These best-fit circles were strongly scaled to the beam length as long as the cross-sectional aspect ratio was over 20. To do this, the deflection path was first broken down into the XY and YZ plane components. Using optimization, a circle was shown to be a close fit in both of these planes. Because these deflection paths were scaled with the beam length, beam deflections could be easily approximated. Beam tip rotations during the buckling process proved to remain nearly constant for beams with cross-sectional aspect ratios over 20. The deflection path and rotation consistencies described in Chapter 4 can facilitate creative ideas and assist designers in visualizing and understanding beams undergoing lateral torsional buckling.

A 3-D chain algorithm was designed that relied on three pseudo-rigid-body models to represent chain elements. This 3-D chain algorithm satisfies force and moment boundary conditions along the length of the beam and also calculates an accurate force-deflection relationship. It allowed for multiple forces and moments to be placed at any node along the beam length with force-deflection relationships given at any node. The 3-D chain algorithm described in Chapter 5 is a general beam design tool that can describe beam deflections and force deflection relationships with good accuracy much faster than finite element methods.

6.2 Recommendations

(A) Displacement Loads

The chain algorithm works well when all loads applied along the beam are force loads. Applying displacement loads, however, is often more convenient than using force loads, particularly for mechanism design purposes. However, when using the chain algorithm, these displacement loads can only be achieved by making a guess and then using an optimization approach as described in Chapter 2. If the guess is not close enough to the right answer, then the chain calculation will not converge to the right solution. A different method that requires only applying the boundary conditions and no guess value would be more convenient. This section outlines a 2-D method to do just that and finishes with a description of the challenges required to follow a similar procedure in 3-D.

To illustrate this 4-bar linkage/chain algorithm approach to solving displacement loads, both ends of the beam will have fixed positions and rotations. First, a vector is drawn from the first link to the last link as shown in Figure 6.1.

Figure 6.1: A vector connects the first link to the last link

Then the vector is divided into the number of segments specified by the user. These small vectors are then lined up tail to tip as shown in Figure 6.2.

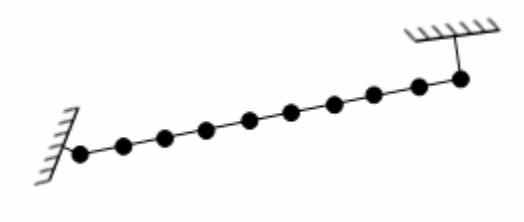


Figure 6.2: Small vectors lined up

Next, the first four-bar linkage in the chain is analyzed. The fifth spring is held constant, as shown in Figure 6.3.

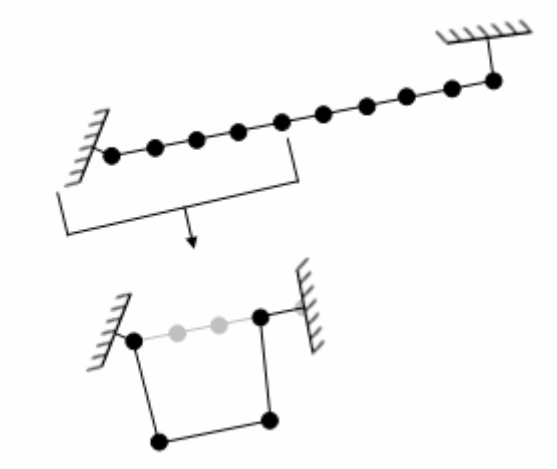


Figure 6.3: Analyzing the first four-bar linkage in the chain

The segments are then given their true length and the springs are allowed to relax. The correct relaxed position means the spring at each node in the four-bar linkage is at its lowest possible energy state with the stipulation that each of the four nodes has the same energy level.

Once this relaxed position is found, the next four-bar linkage is analyzed. The next four-bar linkage includes the second, third, and fourth node of the first four-bar linkage. The new fourth node in the linkage is the fifth node in the chain, which is now no longer held rigid. Instead, the first node and the sixth node in the chain are held rigid. The new link is given its true length and the four-bar system is moved to its relaxed position. The shifting and relaxing process continues until the chain of elements has settled.

The four-bar linkage approximates beam curvature. Force and moment loads can be placed anywhere along the beam and the reaction moments caused by the loads further down the chain can be placed at the fourth node in the four-bar linkage being analyzed, after which the four-bar linkage system assumes its relaxed position.

If not all degrees of freedom are to be constrained at the tip end of the beam, then optimization techniques described in Chapter 2 can be used on the last few elements of the chain. Because there are only a few elements in this final chain, this optimization process should be very fast and require only small deflections. If optimization is used on the last few links, when the chain settles, the accompanying end loads are already known. If no optimization is required on the last few links, the tip forces may be found by using a system of equations that finds the loads based on spring deflections.

If this process were tested and found to work, the next step would be to modify it for the 3-D chain algorithm described in Chapter 5. There are three main challenges that must be overcome to make this work. First, a four-bar system using the 3-D chain elements is not 2-D. Therefore, it is not a four-bar linkage but a four-segment spatial mechanism. Instead of a single segment rotating in a circle around a node as is done with a four-bar linkage, the 3-D chain elements are able to move within a sphere. Second, a four-bar linkage has a single degree of freedom. A four-segment spatial mechanism made of 3-D elements has six degrees of freedom. Third, because there are three springs at each node, there is more than one way for a segment to rotate and twist around its node to get to a point in a specific direction. This makes each element indeterminate.

There are several ideas that could assist in overcoming these challenges. Perhaps the 3-D spatial mechanism could be broken down into three 2-D four-bar linkages that represent the 3-D spatial mechanism projected onto three ortho-normal planes. The three linkage systems would not be independent. As one linkage rotates, all three linkages would experience changes in segment length and rotate by different amounts. A new challenge would arise in describing the three spring components for a given node projected onto each of the three planes.

Another idea that could be useful is to realize what is already known. Three of the four segments are the same length and all nodes are at equal energy levels and collectively at their lowest energy level. Perhaps an equation that describes the full motion of this spatial mechanism could be simplified by these consistencies.

The indeterminacy of each element means there are an infinite number of rotational combinations of the three springs that would allow the segment tip to rest in the same XYZ

position. However, if a coordinate system were attached to that segment and the X-direction pointed along the length of the segment, then perhaps a useful one-to-one relationship could be found. Different combinations of rotations that allow the segment tip to rest at a particular position would probably correspond to the Y- and Z-axes rotated around the X-axis by different amounts. Another possibility that could assist this indeterminacy issue may be the semitangential assumption used in the 3-D chain algorithm described in Chapter 2.

In any case, this problem has yet to be solved. The 3-D chain algorithm has been proven to work with great accuracy, and therefore it can be assumed that a similar 2-D chain algorithm that implements the ideas in Chapter 5 would be just as accurate. However, using the chain algorithm with the four-bar linkage displacement load technique has not been tested. If it were found to work, it would greatly increase the usefulness of the 2-D chain algorithm and add much more meaning to solve the challenges with the 3-D version.

(B) Invention of new mechanisms

The increase in understanding of the lateral torsional buckling phenomena will spark new ideas and inventions. The circular deflection paths and constant tip angle rotation plots described in Chapter 4 play an important part in that. This is because they take a complex buckling process and describe aspects of it very simply. Because they are simple, they can facilitate visualization of lateral torsional buckling in the mind, which is were improved ideas and useful inventions are born.

Many of these ideas could be implemented efficiently using the 3-D chain algorithm. Because the 3-D chain algorithm can be much faster than finite element methods, it could play a roll in optimizing buckling, and large twisted deflections of beams that would perform specific functions. The exciting new looks of twisted, curling mechanisms could give consumers interesting stylistic options for products that boxy 2-D mechanism are not capable of. Mechanisms using different variations of lateral torsional buckling have a bright future in our world.

(C) The accuracy of the 3-D chain algorithm at low aspect ratios

Ansys, using beam element 189, was not able to compute the deflection paths and force deflection relationship at low aspect ratios; however, the chain algorithm could. Figures 6.4 and 6.5 show the XY and YZ plane buckling paths at very low aspect ratios.

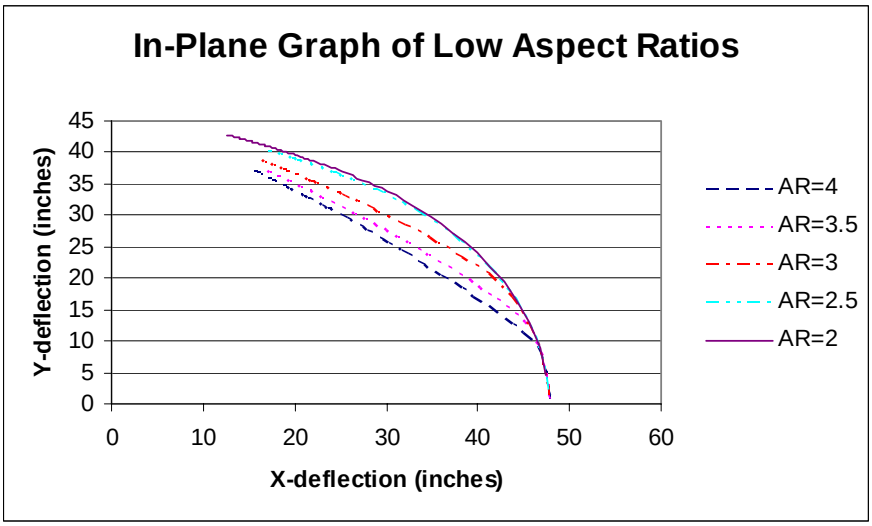


Figure 6.4: XY plane deflection path of beams with low aspect ratios

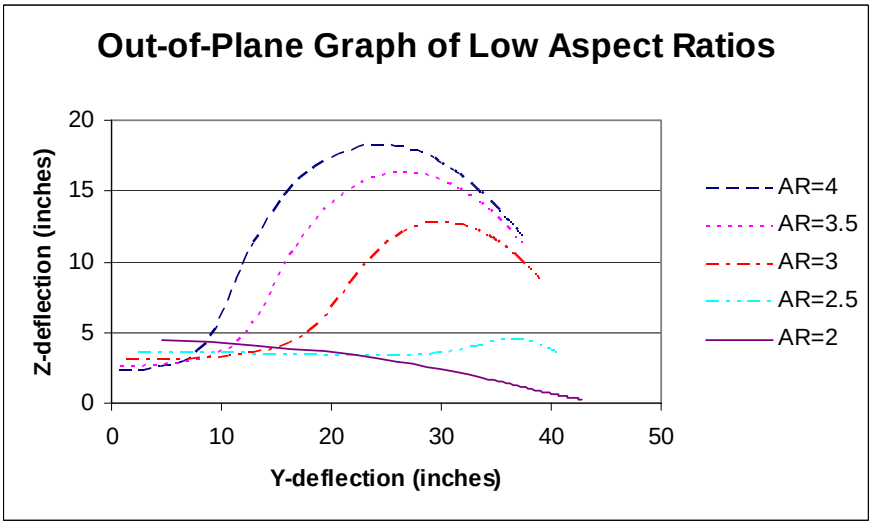


Figure 6.5: YZ plane deflection path of beams with low aspect ratios

As can be seen in these graphs, as the cross-sectional aspect ratio drops, the buckling process is impaired. The YZ buckling bump moves back and gets smaller until it completely disappears. Finally, as the aspect ratio shrinks to two, the beam's Z-direction deflection is only caused by the side force used to initiate buckling. However, this data could not be tested against Ansys because Ansys had difficulties computing buckling deflections when aspect ratios dropped below twenty.

As the cross-sectional aspect ratio drops below 20, a new component to the buckling phenomena becomes important. That component is the warping of the cross-section in the lengthwise or X-direction. At high aspect ratios the cross-section can be thought of as a straight line. Because the chain algorithm only analyzes beams, the cross-section now thought of as a line may rotate or change position because of bending and torsion, but it remains a line with negligible lengthwise warping.

When torsion is applied to a beam with a square cross-section, it will warp into the beam's lengthwise direction. The formula used in the 3-D chain algorithm to describe the torsional spring assumes pure torsion. This means the cross-section is able to warp at both ends of the beam. Warping probably plays an important role at low aspect ratios and can not be neglected. If the 3-D chain algorithm were used to analyze a beam with a low aspect ratio, and torsion were applied to that beam along its axis, then the theoretical rotation around the beam axis based on the 3-D chain algorithm would be weaker than the same case with a physical beam. This is because the real beam would be constrained from warping at its beam root. Constraining the cross-sectional warping increases the resistance to torsion. The 3-D chain algorithm should be compared to another method that can accurately analyze lateral torsional buckling at low aspect ratios when the beam root is constrained.

(D) Other areas for future chain algorithm research

Future work in using the chain algorithm in a dynamic setting could also prove to be useful. In such cases, each linkage would have a mass that would resist changes in velocity. Spring energy and time would also need to be taken into account. Such a model, if efficient when compared to finite element methods, could be used to analyze and optimize a dynamic mechanism such as the lateral torsional bucking mircomirror described in Chapter 5.

Another useful tool where the 3-D chain algorithm could be expanded would be analyzing beams where products of inertia are not equal to zero. An example of this is a beam with a Z-

shaped cross section. If a force is applied at the beam tip, the reaction moment acting along the length of the beam deflects the beam to the side. Figures 6.6 and 6.7 show XY and YZ plane deflection paths for beams with the same length but different cross-sections.

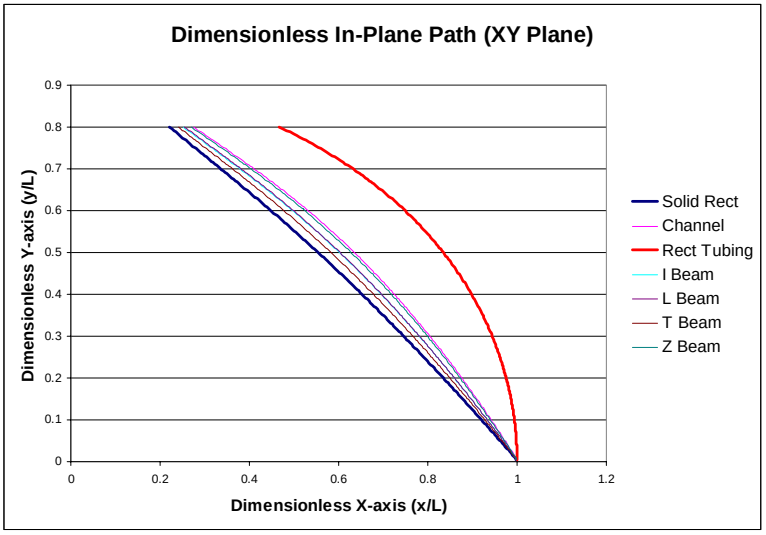


Figure 6.6: XY plane deflection path for beams with same length but different cross-sections

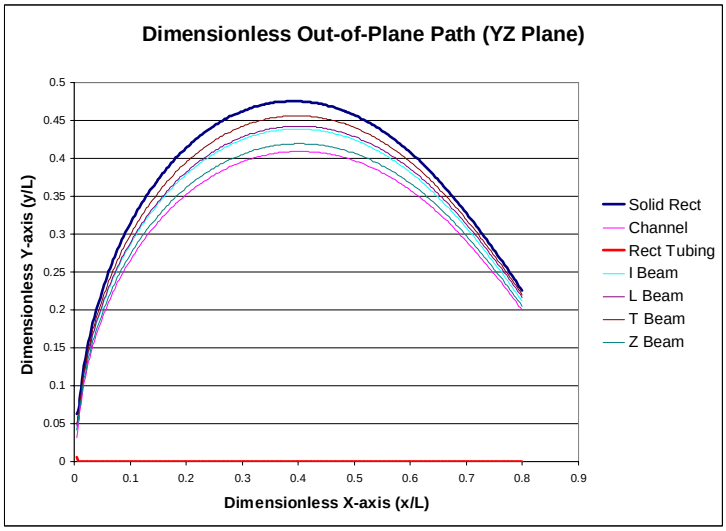


Figure 6.7: YZ plane deflection path for beams with same length but different cross-sections

In both graphs the bounds are set by the solid rectangular cross-sectioned beam with a high aspect ratio and the rectangular, thin-walled tubing that is a boundary because it represents a beam that resists out-of-plane deflection. Some of these beams can be simulated by the chain algorithm because their product of inertia is zero. Others, like the channel and the beam with the Z-shaped cross-section, do not. Future chain algorithm work could consist of modifying the ways springs are modeled. For example, beams with products of inertia not equal to zero could be modeled by a combination of a spring constant and a function, or the spring constants could be modified to describe a composite material. This would add a new dimension to the capability of chain algorithms.

References

[1] Howell, L.L., 2001. *Compliant Mechanisms.* John Wiley & Sons, Inc., New York, p. 261, 266, 166, 192, 191.

[2] Gere, J. M., 2001. *Mechanics of Materials,* 5th ed. Brooks/Cole, Pacific Grove, CA, p. 632.

[3] Rasmussen, N.O., and J.W. Wittwer, R.H. Todd, L.L. Howell, and S.P. Magleby, 2006. *A 3D Pseudo-Rigid-Body Model for Large Spatial Deflections of Rectangular Cantilever Beams.* In Proceedings of IDETC/CIE 2006 ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, September 10–13, 2006, Philadelphia, PA.

[4] Villaggio, P., 1997. *Mathematical Models for Elastic Structures.* Cambridge University Press, Cambridge, UK, p. 219.

[5] Love, A.E.H., 1944. *The Mathematical Theory of Elasticity.* New York Dover Publications, New York, p. 419.

[6] Rasmussen, N.O., and R.H. Todd, L.L. Howell, and S.P. Magleby, 2006. *Investigation of Compliant Ortho-planar Springs for Rotational Applications.* In Proceedings of IDETC/CIE 2006 ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, September 10–13, 2006, Philadelphia, PA.

[7] Coulter, B.A., and R.E. Miller, 1988. "Numerical Analysis of a Generalized Plane 'Elastica' with Non-linear Material Behaviour." *International Journal for Numerical Methods in Engineering* 26 (1988): 617–630, p. 619.

[8] Hill, T.C., and A. Midha, 1990. "A Graphical, User-Driven Newton-Raphson Technique for Use in the Analysis and Design of Compliant Mechanisms." *Journal of Mechanical Design* 112 (March 1990): 123–130, p. 124, 123.

[9] Nahvi, H, 1991. "Static and Dynamic Analysis of Compliant Mechanisms containing Highly Flexible Members." Ph.D. dissertation, Purdue University, 1991, p. 42, 88, 89.

[10] Young, W.C., and R.G. Budynas, 2002. *Roark's Formulas for Stress and Strain.* McGraw-Hill, New York, p. 64, 383.

[11] Pauly, J., and A. Midha, 2006. *Pseudo-Rigid-Body Model Chain Algorithm, Part 1: Introduction and Concept Development.* In Proceedings of IDETC/CIE 2006 ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, September 10–13, 2006, Philadelphia, PA, p. 1, 2, 8.

[12] Pauly, J., and A. Midha, 2006. *Pseudo-Rigid-Body Model Chain Algorithm, Part 2: Equivalent Representations for Combined Load Boundary Conditions.* In Proceedings of IDETC/CIE 2006 ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, September 10–13, 2006, Philadelphia, PA, p. 2.

[13] Murison, M.A., 1998. *How to Rotate a Vector.* Accessed 21 October 2006; available from http://www.alpheratz.net/murison/papers/Notes/RotateVector/RotateVector.pdf#search=%22vectors%2C%20vector%20analysis%20euclidean%20geometry%20murison%22; internet.

[14] "Rotation Matrix." In *Wikipedia: The Free Encyclopedia.* Accessed 14 September 2006, available from http://en.wikipedia.org/wiki/Rotation_matrix; internet.

Appendix A

Ansys Batch File

```
FINISH
/CLEAR
!This batch file calculates maxiumum reaction forces due to a .8L Z-displacement of the
beam tip


!Geometry
L=150
h=1.25
/INPUT,ANSYS_tlb_inputs,txt
b=.0625

!Material Properties
Modulus=30000000
PR=.3

!Deflection
steps=200
!Pz=1.0E-3
L_def=.8

*dim,x_def,ARRAY,steps,1,1,                !x displacement
*dim,y_def,ARRAY,steps,1,1,                !y displacement
*dim,z_def,ARRAY,steps,1,1,                !z displacement
*dim,x_rot,ARRAY,steps,1,1,                !x rotation
*dim,y_rot,ARRAY,steps,1,1,                !y rotation
*dim,z_rot,ARRAY,steps,1,1,                !z rotation
*dim,x_F,ARRAY,steps,1,1,          !x Force
*dim,y_F,ARRAY,steps,1,1,          !y Force
*dim,z_F,ARRAY,steps,1,1,          !z Force
*dim,x_M,ARRAY,steps,1,1,          !x Moment
```

```
*dim,y_M,ARRAY,steps,1,1,          !y Moment
*dim,z_M,ARRAY,steps,1,1,          !z Moment

/PREP7
ET,1,BEAM189
SECTYPE,1,BEAM,RECT    !RECTANGULAR CROSS SECTION
SECDATA,h,b,20,5       !BASE, HEIGHT OF CROSS SECTION
!SECPLOT,1
MP,EX,1,Modulus        !MODULUS OF ELASITICITY
MP,NUXY,1,PR           !POISONS RATIO
K,1
K,2,L
L,1,2
ALLSEL
LESIZE,1,,,200

SECNUM,1
LMESH,ALL
DK,1,ALL


!ASEL,ALL

NLGEOM,ON          !TURN ON NONLINEAR OPTION
Fk,2,FZ,Pz         !APPLY PERTURBATION FORCE

Finish

/solu



*DO,I,1,steps,1

Dk,2,UY,-I/200*L_def*L          !ITERATIVE DISPLACEMENT
!Dk,2,UZ,.8*L
NSUBST,10
            !10 SUBSTEPS PER STEP DISPLACEMENT
SOLVE
/OUT

!*IF,I,EQ,1,Then
!       FKDELE,2,FZ        !DELETES SIDE LOAD AFTER FIRST INTERATION
!*ENDIF

*ENDDO
```

```
KSEL,S,KP,,1
NSLK,S
*GET,node1,NODE,0,NUM,MAX

KSEL,S,KP,,2
NSLK,S
*GET,node2,NODE,0,NUM,MAX

ASEL,ALL
FINISH

/POST1


*DO,n,1,steps,1
        Set,n
        *get,x_def(n),NODE,node2,U,X
        *get,y_def(n),NODE,node2,U,Y
        *get,z_def(n),NODE,node2,U,Z
        *get,x_rot(n),NODE,node2,rot,X
        *get,y_rot(n),NODE,node2,rot,Y
        *get,z_rot(n),NODE,node2,rot,Z
        *get,x_F(n),NODE,node1,RF,FX
        *get,y_F(n),NODE,node1,RF,FY
        *get,z_F(n),NODE,node1,RF,FZ
        *get,x_M(n),NODE,node1,RF,MX
        *get,y_M(n),NODE,node1,RF,MY
        *get,z_M(n),NODE,node1,RF,MZ
*ENDDO

/output,Data_SL150_AR20,txt,,APPEND                    !Outputs the couple curve
points for Excel
!*vwrite
!       X                   Y                   Z                   RX
        RY                  RZ                  FX                  FY
        FZ                  MX                  MY                  MZ
*vwrite,x_def(1),y_def(1),z_def(1),x_rot(1),y_rot(1),z_rot(1),x_F(1),y_F(1),z_F(1),x_M(
1),y_M(1),z_M(1)
(E16.8,2x,E16.8,2x,E16.8,2x,E16.8,2x,E16.8,2x,E16.8,2x,E16.8,2x,E16.8,2x,E16.8,2x,E
16.8,2x,E16.8,2x,E16.8)
/output
Finish
/eof
```

Appendix B

Matlab File used to Iterate Ansys Batch Files

```
%File name: Eval_A_TLB.m
clear all

for J = 1:200
% ========================================
%       Define INPUT Parameters
% ========================================
Pz=0.30776264E+00*1/200*J;
% ========================================
%       Define FILE NAMES
% ========================================

strWinBatchFile = 'run_ansys_tlb.bat';
% ANSYS Job Name - must be the same as in the .bat file
strAnsysJobName = 'ANSYSJOB_tlb';
% Define the ANSYS Working Directory, where all temporary
% files will be created (and deleted).
% Must ensure that ANSYS is started in this directory
% (see the .bat file)
strAnsysDir = '';
% Choose the name for the temporary INPUT PARAMETER text file
strInputFile = sprintf('%s%s',strAnsysDir,'ANSYS_tlb_inputs.txt');
% Choose the name for the temporary OUTPUT RESULTS text file
strResultsFile = sprintf('%s%s',strAnsysDir,'ANSYS_tlb_results.txt');

% Other ansys files that are to be deleted before each run
% (to prevent the files from becoming to large for example)
strErrorFile = sprintf('%s%s%s',strAnsysDir,strAnsysJobName,'.err');
% The status file is the file created in the .bat as the "output" using -o
strStatusFile = sprintf('%s%s%s',strAnsysDir,strAnsysJobName,'.out');

% Set params.display to default if it is not set
%try params.display; catch params.display=1; end
```

```
%try params.errmode; catch params.errmode=0; end
%try params.plot; catch params.plot=0; end

% ========================================
%   Create INPUT PARAMETER file
% ========================================
% When debugging, check to make sure input file is created correctly.

% OPEN the input file for writing
fid = fopen(strInputFile,'w');
if fid == -1
  error(sprintf('Could not open file (check path): %s',strInputFile));
end

fprintf(fid,'\r\n');
fprintf(fid,'%s = %g\r\n','Pz',Pz);

% CLOSE the input file
status=fclose(fid);
if status~=0
  error(sprintf('Could not close file: %s',strInputFile));
end

% ======================================================
%   RUN ANSYS
% ======================================================

cput=cputime;   % Start timer

%errCode=system(strWinBatchFile);
!ansys <ansys_tlb.txt> out.txt

cput2=cputime;
modelTime=cput2-cput;

end
%end
```

Appendix C

Excel Optimization Macro

```
Sub DIMENSION()
'<***Variables***>
L = 100: b = 0.125: E = 200000: Pr = 0.4: Steps = 40: Iter1 = 40
H_F = 10: Iter2 = 5
'<***Shift Cells Down 2***>
    ActiveCell.Rows("1:1").EntireRow.Select
    Selection.Insert Shift:=xlDown
    ActiveCell.Rows("1:1").EntireRow.Select
    Selection.Insert Shift:=xlDown
    ActiveCell.Rows("1:1").EntireRow.Select
    Selection.Insert Shift:=xlDown
    ActiveCell.Rows("1:1").EntireRow.Select
    Selection.Insert Shift:=xlDown
'<***Define Constants***>
    ActiveCell.Select
    ActiveCell.FormulaR1C1 = "Constants"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "L"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = L
    ActiveCell.Offset(-1, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "b"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = b
    ActiveCell.Offset(-1, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "E"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = E
    ActiveCell.Offset(-1, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Pr"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = Pr
    ActiveCell.Offset(-1, 1).Range("A1").Select
```

```
    ActiveCell.FormulaR1C1 = "Steps"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = Steps
    ActiveCell.Offset(-1, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Iter1"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = Iter1
    ActiveCell.Offset(-1, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "H_Factor"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = H_F

    ActiveCell.Offset(0, -6).Range("A1").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
'<***Label Components***>
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "X Disp."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Y Disp."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Z Disp."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "X Rot."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Y Rot."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Z Rot."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "X Force"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Y Force"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Z Force"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "X Mom."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Y Mom."
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Z Mom."
'<***Steps***>
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Steps"

    For i% = 1 To Steps
       ActiveCell.Offset(1, 0).Range("A1").Select
```

```
        ActiveCell.FormulaR1C1 = i%
    Next i%
'<***Dimensionless Displacement Component***>
    ActiveCell.Offset(-Steps, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "DIM X"

    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        ActiveCell.FormulaR1C1 = "=1-ABS(RC1/R3C1)"
    Next i%

    ActiveCell.Offset(-Steps, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "DIM Y"

    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        ActiveCell.FormulaR1C1 = "=ABS(RC2/R3C1)"
    Next i%

    ActiveCell.Offset(-Steps, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "DIM Z"

    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        ActiveCell.FormulaR1C1 = "=ABS(RC3/R3C1)"
    Next i%
'<***Dimensionless Force Component***>
    ActiveCell.Offset(-Steps, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "DIM FX"

    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        ActiveCell.FormulaR1C1 = "=ABS(RC7*R3C1^2/(R3C2*(R3C2*R3C7*" & q% &
")^3/12*R3C3))"
    Next i%

    ActiveCell.Offset(-Steps, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "DIM FY"

    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        ActiveCell.FormulaR1C1 = "=ABS(RC8*R3C1^2/(R3C2*(R3C2*R3C7*" & q% &
")^3/12*R3C3))"
    Next i%

    ActiveCell.Offset(-Steps, 1).Range("A1").Select
```

```
    ActiveCell.FormulaR1C1 = "DIM FZ"

    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        ActiveCell.FormulaR1C1 = "=ABS(RC9*R3C1^2/(R3C2*(R3C2*R3C7*" & q% &
")^3/12*R3C3))"
    Next i%
'<***Prepare Path Optimization***>
    ActiveCell.Offset(-Steps, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Distance"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "ABS(AVG-DIST)"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "X-Coord."
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = 0
    ActiveCell.Offset(-1, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Y-Coord."
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = 0
'<***Prepare Path Optimization Tables***>
    ActiveCell.Offset(-1, -3).Range("A1").Select
    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        Cell_Value = "=SQRT((R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% -
1) & "C22-RC[-6])^2+(R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C23-RC[-5])^2)"
        ActiveCell.FormulaR1C1 = Cell_Value
    Next i%

    ActiveCell.Offset(-Steps + 2, 2).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Average"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=Average(R[-2]C[-2]:R[" & Steps - 3 & "]C[-2])"
    ActiveCell.Offset(-3, -1).Range("A1").Select
    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        Cell_Value = "=ABS(RC[-1]-R" & 7 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 *
(q% - 1) & "C22)"
        ActiveCell.FormulaR1C1 = Cell_Value
    Next i%
    ActiveCell.Offset(-Steps + 4, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Sum"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C[-1]:R[" & Steps - 5 & "]C[-1])"
    ActiveCell.Offset(1, 0).Range("A1").Select
```

```
    ActiveCell.FormulaR1C1 = "Max Error"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=MAX(R[-6]C[-1]:R[" & Steps - 7 & "]C[-1])"
'<***Solve***>
    Cell_Value1 = "$V$" & 9 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1)
    Cell_Value2 = "$V$" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
":$W$" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1)
    SolverOk SetCell:=Cell_Value1, MaxMinVal:=2, ValueOf:="0",
ByChange:=Cell_Value2
    SolverSolve UserFinish:=True
'<***Prepare Path Optimization***>
    ActiveCell.Offset(-7, 2).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Distance"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "ABS(AVG-DIST)"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Y-Coord."
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = 0
    ActiveCell.Offset(-1, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Z-Coord."
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = 0
'<***Prepare Path Optimization Tables***>
    ActiveCell.Offset(-1, -3).Range("A1").Select
    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        Cell_Value = "=SQRT((R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% -
1) & "C26-RC[-9])^2+(R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C27-RC[-8])^2)"
        ActiveCell.FormulaR1C1 = Cell_Value
    Next i%

    ActiveCell.Offset(-Steps + 2, 2).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Average"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=Average(R[-2]C[-2]:R[" & Steps - 3 & "]C[-2])"
    ActiveCell.Offset(-3, -1).Range("A1").Select
    For i% = 1 To Steps
        ActiveCell.Offset(1, 0).Range("A1").Select
        Cell_Value = "=ABS(RC[-1]-R" & 7 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 *
(q% - 1) & "C26)"
        ActiveCell.FormulaR1C1 = Cell_Value
    Next i%
    ActiveCell.Offset(-Steps + 4, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Sum"
```

```
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C[-1]:R[" & Steps - 5 & "]C[-1])"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Max Error"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=MAX(R[-6]C[-1]:R[" & Steps - 7 & "]C[-1])"
'<***Solve***>
    Cell_Value1 = "$Z$" & 9 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1)
    Cell_Value2 = "$Z$" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
":$AA$" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1)
    SolverOk SetCell:=Cell_Value1, MaxMinVal:=2, ValueOf:="0",
ByChange:=Cell_Value2
    SolverSolve UserFinish:=True
'<***Prepare for next iteration***>
ActiveCell.Offset(-7, -25).Range("A1").Select
ActiveCell.Offset((Steps + 1), 0).Range("A1").Select
ActiveCell.Rows("1:1").EntireRow.Select
Selection.Insert Shift:=xlDown
ActiveCell.Offset(-1, 0).Range("A1").Select
Next j%
Next q%

'<***Results***>
Range("AC1").Select
    ActiveCell.FormulaR1C1 = "Results"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "AR"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Zo-Def"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "XY-X"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "XY-Y"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "XY-Radius"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Max Error"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "YZ-X"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "YZ-Y"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "YZ-Radius"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Max Error"
    Range("AC2").Select
```

```
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = q% * H_F
    Next j%
    Next q%
    Range("AD2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C16"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AE2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C22"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AF2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C23"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AG2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 7 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C22"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AH2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
```

```
    Cell_Value = "=R" & 11 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C22"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AI2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C26"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AJ2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 5 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C27"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AK2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 7 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C26"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
    Range("AL2").Select
    For q% = 1 To Iter2
    For j% = 1 To Iter1
    ActiveCell.Offset(1, 0).Range("A1").Select
    Cell_Value = "=R" & 11 + (Steps + 1) * (j% - 1) + (Steps + 1) * Iter1 * (q% - 1) &
"C26"
    ActiveCell.FormulaR1C1 = Cell_Value
    Next j%
    Next q%
End Sub
```

Appendix D

How to Rotate a Vector [13]

HOW TO ROTATE A VECTOR

MARC A. MURISON
Astronomical Applications Deptartment, U. S. Naval Observatory, Washington, DC
murison@aa.usno.navy.mil

November 23, 1998

ABSTRACT
Rotation of a vector around a direction in space is shown.

Key words: vectors — vector analysis — Euclidean geometry

1. ROTATION OF A VECTOR.

We wish to rotate a vector $\vec{v}$ counterclockwise around an axis $\vec{w}$ by an angle φ. The geometric picture is illustrated in Figure 1. The components of $\vec{v}$ and $\vec{v}'$ perpendicular to $\vec{w}$ are $\vec{u}$ and $\vec{u}'$. We have

$$\vec{u} = \vec{v} - (\vec{v}\cdot\hat{w})\hat{w} \tag{1}$$

and

$$\vec{u}' = \vec{v}' - (\vec{v}'\cdot\hat{w})\hat{w} \tag{2}$$

since it is apparent that $\vec{v}\cdot\hat{w} = \vec{v}'\cdot\hat{w}$.

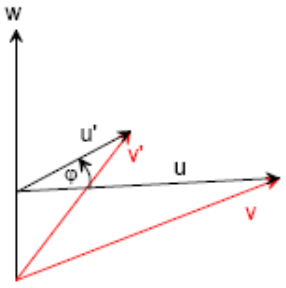


Figure 1

The trick to determining $\vec{v}'$ is to notice that the rotation of the vector is just a coordinate rotation in the $\vec{u}\vec{u}'$ plane. The unit vector perpendicular to $\vec{u}$ is $\frac{\hat{w}\times\vec{u}}{|\hat{w}\times\vec{u}|}$. Hence we can write $\vec{u}'$ in terms of its components parallel and perpendicular to $\vec{u}$,

$$\vec{u}' = u\left(\hat{u}\cos\varphi + \frac{\hat{w}\times\vec{u}}{|\hat{w}\times\vec{u}|}\sin\varphi\right) \tag{3}$$

But $\|\hat{w}\times\vec{u}\| = \sqrt{u^2 - (\vec{u}\cdot\hat{w})^2} = u$, so

$$\vec{u}' = \vec{u}\cos\varphi - \vec{u}\times\hat{w}\sin\varphi \tag{4}$$

Now, from (2) we have

$$\vec{v}' = \vec{u}' + (\vec{v}'\cdot\hat{w})\hat{w} \tag{5}$$

Using (4) for $\vec{u}'$ and (1) for $\vec{u}$, this becomes

$$\begin{aligned}\vec{v}' &= \vec{u}\cos\varphi - \vec{u}\times\hat{w}\sin\varphi + (\vec{v}'\cdot\hat{w})\hat{w} \\ &= [\vec{v} - (\vec{v}\cdot\hat{w})\hat{w}]\cos\varphi \\ &\quad - [\vec{v} - (\vec{v}\cdot\hat{w})\hat{w}]\times\hat{w}\sin\varphi \\ &\quad + (\vec{v}'\cdot\hat{w})\hat{w}\end{aligned} \tag{6}$$

Hence we have the result

$$\boxed{\vec{v}' = \vec{v}\cos\varphi + (1-\cos\varphi)(\vec{v}\cdot\hat{w})\hat{w} - \vec{v}\times\hat{w}\sin\varphi} \tag{7}$$

2. ROTATION OF A POSITION VECTOR AROUND A POINT IN SPACE.

Now consider a position vector $\vec{r}$ relative to a coordinate origin O. We wish to rotate this position vector counterclockwise by and angle φ around an axis $\vec{w}$ with anchor point $\vec{p}$. See Figure 2.

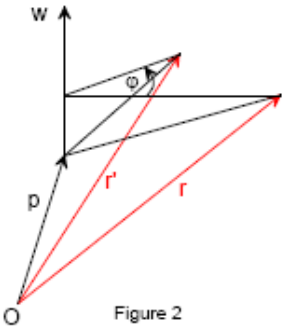


Figure 2

Notice that eq. (7) can be viewed as a linear operator, $\vec{R}_{\hat{w},\varphi}(\vec{v})$, acting on the argument $\vec{v}$. That is, let

$$\vec{R}_{\hat{w},\varphi}(\vec{v}) = \vec{v}\cos\varphi + (1-\cos\varphi)(\vec{v}\cdot\hat{w})\hat{w} - \vec{v}\times\hat{w}\sin\varphi \tag{8}$$

Then it is apparent from Figure 2 that the rotated position vector can be written

$$\boxed{\vec{r}' = \vec{p} + \vec{R}_{\hat{w},\varphi}(\vec{r}-\vec{p})} \tag{9}$$

Appendix E

3-D Chain VB Code

```
Private Sub Command1_Click()
'Dimension Variables
Dim s_L As Double, seg As Integer, F(3) As Double
Dim dfy As Double, df(3) As Double, Iter_total As Integer, Iter As Integer
Dim Tip_last(3) As Double, Tip(3) As Double
Dim b As Double, h As Double, E As Double, mu As Double, G As Double
Dim Kt As Double, Kx As Double, Ky As Double, II As Double, Kz As Double
Dim ux(1000, 3) As Double, Phi0 As Double, D(1000, 3) As Double
Dim uy(1000, 3) As Double, uz(1000, 3) As Double, MM(1000, 3) As Double
Dim Phi(1000) As Double, Phi_ref(1000) As Double, Digits As Integer
Dim v(3) As Double, w(3) As Double, Theta(1000) As Double, Theta_ref(1000) As
Double
Dim Theta2(1000) As Double, Theta2_ref(1000) As Double, s(100) As String * 25

'Link Geometry
s_L = Text4.Text: seg = Text3.Text: F(1) = Text5.Text: F(2) = Text6.Text: F(3) =
Text7.Text

'Initialize Iteration and Data Collection Variables
dfy = Text9.Text: df(1) = 0: df(2) = dfy: df(3) = 0
Iter_total = Text11.Text: Iter = -1: Digits = Text16.Text
Tip_last(1) = 0: Tip_last(2) = 0: Tip_last(3) = 0
Tip(1) = 0: Tip(2) = 0: Tip(3) = 0

'Material and Spring Properties
b = Text12.Text: h = Text13.Text: E = Text14.Text: mu = Text15.Text:
G = E / (2 * (1 + mu)): II = b * h ^ 3 / 12: II2 = h * b ^ 3 / 12
Kta = b * h ^ 3: Ktb = (1 / 3 - 0.21 * h / b * (1 - h ^ 4 / (12 * b ^ 4)))
Kt = Kta * Ktb:
Kx = Kt * G / s_L: Ky = 1.5164 * E * II / s_L: Kz = 1.5164 * E * II2 / s_L

'Create and Initialize Unit Vectors
For i% = 1 To seg
```

```
   ux(i%, 1) = 1: ux(i%, 2) = 0: ux(i%, 3) = 0
   uy(i%, 1) = 0: uy(i%, 2) = 1: uy(i%, 3) = 0
   uz(i%, 1) = 0: uz(i%, 2) = 0: uz(i%, 3) = 1
Next i%

'Create and Initialize Angle Variables
Phi0 = 0: v(1) = 0: v(2) = 0: v(3) = 0: w(1) = 0: w(2) = 0: w(3) = 0

'Find Distance to each Segment Tip

For n% = 1 To seg
   For m% = 1 To seg
      If m% > n Then
         D(n%, 1) = D(n%, 1) + ux(m% - 1, 1) * s_L * (1 - 0.7346)
         D(n%, 2) = D(n%, 2) + ux(m% - 1, 2) * s_L * (1 - 0.7346)
         D(n%, 3) = D(n%, 3) + ux(m% - 1, 3) * s_L * (1 - 0.7346)
         D(n%, 1) = D(n%, 1) + ux(m%, 1) * s_L * 0.7346
         D(n%, 2) = D(n%, 2) + ux(m%, 2) * s_L * 0.7346
         D(n%, 3) = D(n%, 3) + ux(m%, 3) * s_L * 0.7346
      End If
   Next m%
Next n%

'Main Loop
While Iter < Iter_total
DoEvents
   'Iterate Force if Beam has settled to specified digits of precision
   If Left(Tip(1), Digits) = Left(Tip_last(1), Digits) And Left(Tip(2), Digits) =
Left(Tip_last(2), Digits) And Left(Tip(3), Digits) = Left(Tip_last(3), Digits) Then
      Iter = Iter + 1
      Label1.Caption = Iter
      If Iter > 0 Then
         s(1) = Tip(1): s(2) = Tip(2): s(3) = Tip(3): s(4) = F(1): s(5) = F(2): s(6) = F(3):
s(7) = ACos(dot(0, 0, 1, uz(seg, 1), uz(seg, 2), uz(seg, 3))): s(8) = Atn(uz(seg, 2) / uz(seg,
1)): s(9) = Chr(13) & Chr(10)
         Text1.Text = Text1.Text & s(1) & s(2) & s(3) & s(4) & s(5) & s(6) & s(7) & s(8)
& Chr(13) & Chr(10)
         If CommonDialog1.FileName <> "" Then
            Open CommonDialog1.FileName For Append As #1
               Print #1, s(1) & s(2) & s(3) & s(4) & s(5) & s(6) & s(7) & s(8) '& s(9) &
s(10) & s(11) & s(12) & s(13) & s(14) & s(15) & s(16) & s(17) & s(18) & s(19) & s(20)
            Close #1
         End If
      End If
      F(1) = add_vectors(F(1), F(2), F(3), df(1), df(2), df(3), 1)
      F(2) = add_vectors(F(1), F(2), F(3), df(1), df(2), df(3), 2)
```

```
      F(3) = add_vectors(F(1), F(2), F(3), df(1), df(2), df(3), 3)
   End If
   Tip_last(1) = Tip(1): Tip_last(2) = Tip(2): Tip_last(3) = Tip(3)

'Single pass on each segment
For n% = 1 To seg
DoEvents

   'Find Moment
   MM(n%, 1) = cross(D(n%, 1), D(n%, 2), D(n%, 3), F(1), F(2), F(3), 1)
   MM(n%, 2) = cross(D(n%, 1), D(n%, 2), D(n%, 3), F(1), F(2), F(3), 2)
   MM(n%, 3) = cross(D(n%, 1), D(n%, 2), D(n%, 3), F(1), F(2), F(3), 3)

   'Take Moment Component in X-Direction and find Phi (Torsional Pivot Phi)
   Phi(n%) = dot(MM(n%, 1), MM(n%, 2), MM(n%, 3), ux(n%, 1), ux(n%, 2), ux(n%,
3)) / Kx - Phi_ref(n%)
   Phi_ref(n%) = dot(MM(n%, 1), MM(n%, 2), MM(n%, 3), ux(n%, 1), ux(n%, 2),
ux(n%, 3)) / Kx

   'Pivot Y and Z unit vectors around X unit vector
   v(1) = uy(n%, 1): v(2) = uy(n%, 2): v(3) = uy(n%, 3)
   w(1) = ux(n%, 1): w(2) = ux(n%, 2): w(3) = ux(n%, 3)
   Phi0 = Phi(n%)
   uy(n%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
   uy(n%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
   uy(n%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
   v(1) = uz(n%, 1): v(2) = uz(n%, 2): v(3) = uz(n%, 3)
   uz(n%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2), w(3))
* w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
   uz(n%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2), w(3))
* w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
   uz(n%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2), w(3))
* w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)

   'Pivot all segments after current segment with respect to Phi
   If n% < seg Then
      For m% = n% + 1 To seg
         v(1) = ux(m%, 1): v(2) = ux(m%, 2): v(3) = ux(m%, 3)
         ux(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
         ux(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
```

```
        ux(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
        v(1) = uy(m%, 1): v(2) = uy(m%, 2): v(3) = uy(m%, 3)
        uy(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
        uy(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
        uy(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
        v(1) = uz(m%, 1): v(2) = uz(m%, 2): v(3) = uz(m%, 3)
        uz(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
        uz(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
        uz(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
      Next m%
   End If

   'Take Moment Component in Z-Direction and find Theta (Bending Pivot Theta)
   Theta(n%) = dot(MM(n%, 1), MM(n%, 2), MM(n%, 3), uy(n%, 1), uy(n%, 2), uy(n%,
3)) / Ky - Theta_ref(n%)
   Theta_ref(n%) = dot(MM(n%, 1), MM(n%, 2), MM(n%, 3), uy(n%, 1), uy(n%, 2),
uy(n%, 3)) / Ky

   'Pivot X and Z unit vectors around Y unit vector
   v(1) = ux(n%, 1): v(2) = ux(n%, 2): v(3) = ux(n%, 3)
   w(1) = uy(n%, 1): w(2) = uy(n%, 2): w(3) = uy(n%, 3)
   Phi0 = Theta(n%)
   ux(n%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
   ux(n%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
   ux(n%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
   v(1) = uz(n%, 1): v(2) = uz(n%, 2): v(3) = uz(n%, 3)
   uz(n%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2), w(3))
* w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
   uz(n%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2), w(3))
* w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
   uz(n%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2), w(3))
* w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)

   'Pivot all segments after current segment with respect to Theta
   Phi0 = Theta(n%) * 1.5164
   If n% < seg Then
```

```
      For m% = n% + 1 To seg
         v(1) = ux(m%, 1): v(2) = ux(m%, 2): v(3) = ux(m%, 3)
         ux(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
         ux(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
         ux(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
         v(1) = uy(m%, 1): v(2) = uy(m%, 2): v(3) = uy(m%, 3)
         uy(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
         uy(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
         uy(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
         v(1) = uz(m%, 1): v(2) = uz(m%, 2): v(3) = uz(m%, 3)
         uz(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
         uz(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
         uz(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
      Next m%
   End If

   'Take Moment Component in Y-Direction and find Theta2 (Bending Pivot Theta2)
   Theta2(n%) = dot(MM(n%, 1), MM(n%, 2), MM(n%, 3), uz(n%, 1), uz(n%, 2),
uz(n%, 3)) / Kz - Theta2_ref(n%)
   Theta2_ref(n%) = dot(MM(n%, 1), MM(n%, 2), MM(n%, 3), uz(n%, 1), uz(n%, 2),
uz(n%, 3)) / Kz

   'Pivot X and Z unit vectors around Y unit vector
   v(1) = ux(n%, 1): v(2) = ux(n%, 2): v(3) = ux(n%, 3)
   w(1) = uz(n%, 1): w(2) = uz(n%, 2): w(3) = uz(n%, 3)
   Phi0 = Theta2(n%)
   ux(n%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
   ux(n%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
   ux(n%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
   v(1) = uy(n%, 1): v(2) = uy(n%, 2): v(3) = uy(n%, 3)
   uy(n%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
   uy(n%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
```

```
   uy(n%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)

   'Pivot all segments after current segment with respect to Theta2
   Phi0 = Theta2(n%) * 1.5164
   If n% < seg Then
      For m% = n% + 1 To seg
         v(1) = ux(m%, 1): v(2) = ux(m%, 2): v(3) = ux(m%, 3)
         ux(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
         ux(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
         ux(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
         v(1) = uy(m%, 1): v(2) = uy(m%, 2): v(3) = uy(m%, 3)
         uy(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
         uy(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
         uy(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
         v(1) = uz(m%, 1): v(2) = uz(m%, 2): v(3) = uz(m%, 3)
         uz(m%, 1) = v(1) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(1) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 1) * Sin(Phi0)
         uz(m%, 2) = v(2) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(2) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 2) * Sin(Phi0)
         uz(m%, 3) = v(3) * Cos(Phi0) + (1 - Cos(Phi0)) * dot(v(1), v(2), v(3), w(1), w(2),
w(3)) * w(3) - cross(v(1), v(2), v(3), w(1), w(2), w(3), 3) * Sin(Phi0)
      Next m%
   End If

   'Update Distance to each Segment Tip
   For i% = 1 To seg
      For j% = 1 To 3
         D(i%, j%) = 0
      Next j%
   Next i%
   For i% = 1 To seg
      For j% = 1 To seg
         If j% > i% Then
            D(i%, 1) = D(i%, 1) + ux(j% - 1, 1) * s_L * (1 - 0.7346)
            D(i%, 2) = D(i%, 2) + ux(j% - 1, 2) * s_L * (1 - 0.7346)
            D(i%, 3) = D(i%, 3) + ux(j% - 1, 3) * s_L * (1 - 0.7346)
            D(i%, 1) = D(i%, 1) + ux(j%, 1) * s_L * 0.7346
            D(i%, 2) = D(i%, 2) + ux(j%, 2) * s_L * 0.7346
            D(i%, 3) = D(i%, 3) + ux(j%, 3) * s_L * 0.7346
```

```
            End If
        Next j%
    Next i%
Next n%
Tip(1) = D(1, 1) + s_L * (1 - 0.7346) + ux(1, 1) * s_L * 0.7346
Tip(2) = D(1, 2) + ux(1, 2) * s_L * 0.7346
Tip(3) = D(1, 3) + ux(1, 3) * s_L * 0.7346
Wend
End Sub
```