

# Determination of Loading on Bird Feather Shafts

Senior Project – Mechanical Engineering – 2008

Alex Weintraub

Advisor: Prof. William Keat Ph.D.

Co-Advisor: Prof. Andrew Rapoff Ph.D.

## Table of Contents

1	Introduction.....	1
1.1	Purpose.....	1
1.2	Motivation.....	1
1.3	Wing and Feather Morphology.....	1
1.4	Related Work .....	4
1.5	Project Objectives .....	5
2	Development of a 3D Beam Model .....	6
2.1	Description of the Beam Element .....	6
2.1.1	Material Properties and Geometry .....	6
2.1.2	Degrees-of-Freedom .....	6
2.1.3	Global and Local Element Coordinate Systems .....	6
2.2	Element Stiffness Equations in Local Coordinates.....	7
2.3	Element Rotation Matrix.....	8
2.4	Governing Stiffness Equations in Global Coordinates .....	8
2.4.1	Transformation of the Element Stiffness Equations to Global Coordinates.....	8
2.4.2	Assembly of Results to form a Global Stiffness Matrix and Finding the Nodal Displacements .....	9
2.5	Calculation of Stresses.....	10
2.5.1	Calculation of Element Loads.....	10
2.5.2	Calculation of the Critical States of Stress .....	10
2.5.3	Evaluation of Principal Stresses.....	11
2.6	Results.....	11
2.6.1	Comparison with COSMOS .....	11
2.6.2	Plotting Results .....	12
2.7	Summary .....	13
3	Determination of Loading on a 3D Beam by Optimization.....	14
3.1	Constant Stress and Constant Stiffness Hypotheses .....	14
3.2	Function Being Minimized .....	14
3.3	Parameterization of Loads .....	14
3.4	Minimization Methods.....	16
3.4.1	Minimization by the Simplex Method .....	16
3.4.2	Minimization by the Univariate Method.....	16
3.5	Comparison of Minimization Methods.....	17
4	Characterization of Feather Shaft Geometry .....	19
4.1	CT Scanning.....	19
4.1.1	Equipment.....	19
4.1.2	Calibrations .....	19
4.1.3	Scanning Process .....	20
4.2	Cross Sectional Property Calculations through MATLAB .....	22
4.3	Preliminary Results.....	24
4.4	Conclusions.....	25
5	Conclusions.....	26
5.1	Summary .....	26
5.2	Future Work .....	26
6	References.....	28

7	Appendix – MATLAB Scripts.....	A1
7.1	3D Beam Code Function.....	A1
7.2	Univariate Optimization Code .....	A8

# 1 Introduction

## 1.1 Purpose

The purpose of this project is to understand the mechanical properties of bird feather shafts and to determine why a feather shaft is shaped the way that it is.

## 1.2 Motivation

We are interested in learning about the mechanical properties of bird feather shafts because understanding this biological system will allow for us to mimic what occurs in nature in our own aerial designs. Originally we planned to look at hummingbird feather shafts to improve upon micro aerial vehicles (MAVs) because the wings of MAVs mimic that of the hummingbird.

## 1.3 Wing and Feather Morphology

In the following section we will specifically look at the ruby-throated hummingbird seen below in Figure 1 [1].



Figure 1 – The ruby-throated hummingbird in some phase of hovering flight [1].

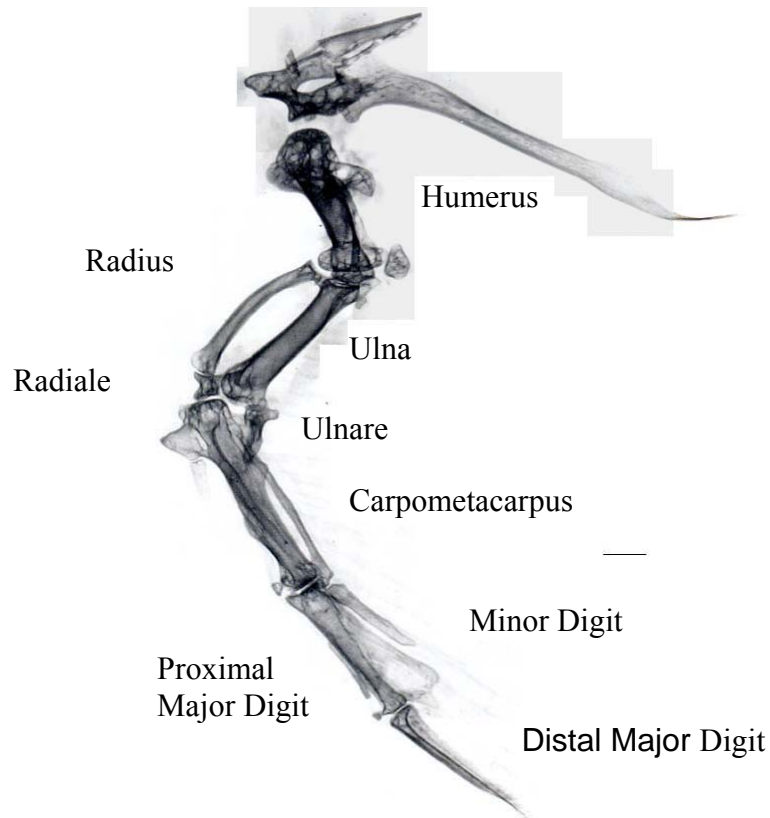
The wing, shown in Figure 2, is comprised of feathers, skin, bones, and musculoskeletal tissue.



**Figure 2 – A ruby-throated hummingbird wing with several of the feathers labeled [1].**

The wing has a span of 40mm, a root chord of 12mm, and an aspect ratio of 6.6. There are 16 flight feathers consisting of 10 primary feathers and 6 secondary feathers. The primary feathers emanate from locations near the phalanges and carpometacarpus bones of the manus (hand), and are numbered in ascending order nearest the body and proceeding outward. The secondary feathers emanate from locations near the ulna and radius bones of the forearm, and are numbered in descending order from nearest the body and proceeding outward. It is important to note that the flight feathers form the lifting surface of the wings, by having the leading edge of each feather (except for P10) lie on top of the trailing edge of the feather closer to the leading edge.

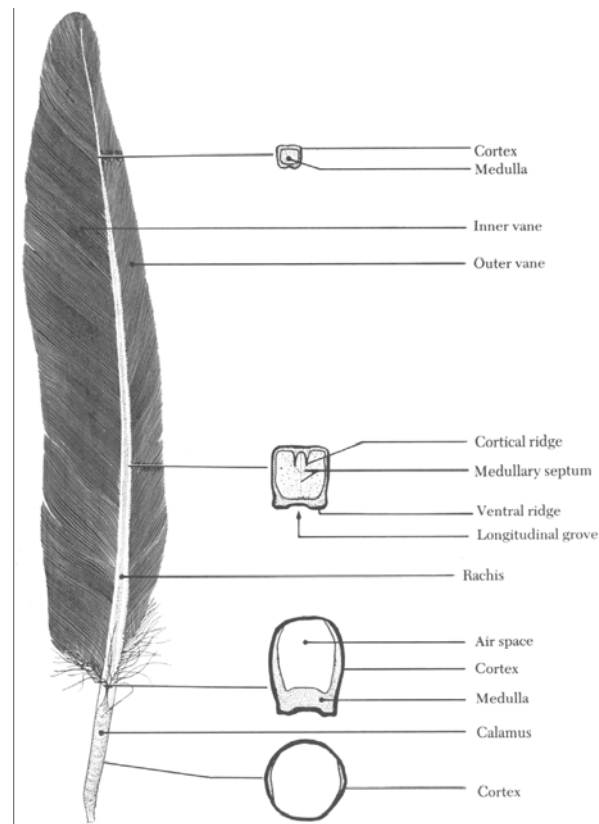
The bone structure of the wing is similar to that of a human arm, as shown in Figure 3.



**Figure 3 – The bone structure of a ruby-throated hummingbird wing [1].**

The load bearing bones of the wing are the humerus, ulna, radius, carpometacarpus, and the major and minor digits. The secondary bones are the ulnare, radiale, and the alular digit. It also should be noted that the bone structure of the wing, Figure 3, is only the upper left portion of the whole wing, Figure 2, with the rest of the wing being made up of the feathers.

Now we will look at one flight feather, Figure 4. It is important to note that the external appearance of a flight feather of the hummingbird is similar to that of larger birds only much smaller in size. This is important because we will be looking at the feathers of several different birds and not only hummingbirds.



**Figure 4 – A typical structure of a flight feather [1].**

The feather shaft is made up of keratin and divided into two sections, the calamus and the rachis. The calamus is located in the skin and has a consistent circular cross section comprised of cortex. The rachis is located outside of the skin and supports the vanes. The rachis is comprised of cortex, medulla, and air space, but is of specific interest because of how the cross section changes along the length of the rachis.

## **1.4 Related Work**

There has not been a substantial amount of research done on the structural properties of feathers. Although the amount of research in this field is limited, the research that has been done will be very influential in the work that we will be doing for this project.

Some of the earliest research on the structural properties of feathers shafts was done by Pennycuik and Lock in 1976 [2]. They conducted mechanical tests of whole pigeon feather shafts and 10 to 20mm sections from the shaft. The results that they

received are not very relevant to most of the current work on feathers, except for the fact that they determined that the shaft keratin is fairly linear elastic.

Andrew Rapoff, of Union College, recently worked to answer the following question of feather morphology: why is the rachis not axisymmetric in cross section [1]? To do this he developed stiffness and strength ratios for both square and circular cross sections that were subjected to bending, torsion, and direct shear loading. From this analysis he was able to determine that the shape of the cross section varies along the hummingbird feather shaft to create the optimal structure for the given loading.

## **1.5 Project Objectives**

To fully understand the mechanical properties of a bird feather shaft we plan to first fully characterize the feather shaft geometry in the SolidWorks program through the use of the CT scanner and several associated programs. With this model in SolidWorks we will then create a 3D beam model in the MATLAB program using a code that we have developed. Next this beam model will be run through an optimization code in MATLAB that will allow us to determine the loading on the feather shaft. With the loading determined we will then be able to analyze the results and hopefully compare the results from different birds. Hopefully we will be able to use these results to improve upon MAVs or another real world application.

## 2 Development of a 3D Beam Model

### 2.1 Description of the Beam Element

To model a bird feather shaft we will create a 3D beam model of the feather shaft using the MATLAB program. The model will be made up of beam elements that will characterize the feather shaft.

#### 2.1.1 Material Properties and Geometry

The material properties and the geometry of the feather shaft will need to be initially defined. Values for the elastic modulus, Poisson's ratio, the geometry and area of the cross section, the length of the shaft, the polar moment of inertia, and the moment of inertia with respect to the y and z-axes will all need to be defined. The shear modulus will be determined from the elastic modulus and Poisson's ratio by Equation 1.

$$G = \frac{E}{2(1 + \nu)} \quad (1)$$

It is important to note that the values for the material properties and geometry will vary along the length of the shaft, which must be accounted for.

#### 2.1.2 Degrees-of-Freedom

Each node of a beam element can have up to 6 degrees-of-freedom (DOF). This allows for forces and moments in the x, y, and z-directions to be present. For our model the 1<sup>st</sup> node at the end of the feather shaft will be fixed with respect to all 6 DOF, to represent that the end of the shaft that would be in the bone of the wing. Every other node will be free with respect to all 6 DOF.

#### 2.1.3 Global and Local Element Coordinate Systems

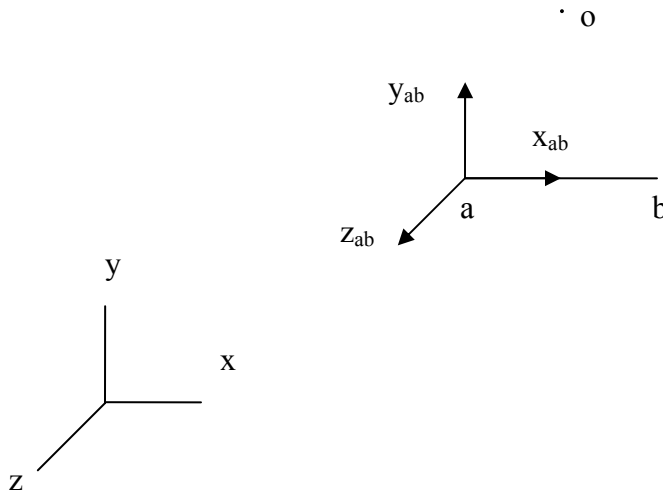
Each beam element is referenced with respect to a global and local coordinate system. The global coordinate system is one fixed system to which all of the local coordinate systems are referenced and is relative to the entire 3D beam model. The local coordinate systems are relative to each element and aligned with the element. The x-

direction of the local coordinate system will be defined by the two nodes of the element, while the y and z-directions will be defined by an arbitrary orienting point. With the known values of the unit vectors for the x-direction,  $i_1$ , and the orienting point,  $o_1$ , I can now determine the unit vectors for the y and z-directions with Equation 2 and Equation 3 respectively.

$$k_l = i_l \times o_l \quad (2)$$

$$j_l = k_l \times i_l \quad (3)$$

By determining these values a local coordinate system can now be established for each element, as seen in Figure 5 , which will be essential to our analysis.



**Figure 5 – An arbitrary element with both the global and local coordinate systems shown.**

## 2.2 Element Stiffness Equations in Local Coordinates

The local stiffness matrix will need to be determined for each element from the given material properties and geometry of the feather shaft. This matrix is of interest because by knowing either the loading or the displacements for an element we can use Equation 4 to determine the other.

$$[K]_l \{u\}_l = \{s\}_l \quad (4)$$

In this equation the local stiffness matrix is denoted by  $[K]_l$  and is of size  $12 \times 12$ , the local nodal displacements are denoted by  $\{u\}_l$ , which is of size  $12 \times 1$ , and the local nodal forces are denoted by  $\{s\}_l$ , which is of size  $12 \times 1$  as well.

## 2.3 Element Rotation Matrix

The element rotation matrix transforms values either from global coordinates to local coordinates or from local coordinates to global coordinates. The matrix is constructed by first determining the transformation matrix.

$$[T] = \begin{bmatrix} i \cdot i_l & j \cdot i_l & k \cdot i_l \\ i \cdot j_l & j \cdot j_l & k \cdot j_l \\ i \cdot k_l & j \cdot k_l & k \cdot k_l \end{bmatrix} \quad (5)$$

The transformation matrix,  $T$ , is determined from the relation between the global and local coordinate systems. Note that this matrix is  $3 \times 3$  and can currently only transfer a triplet of forces or moments that are acting on one node. To apply this to elements the transformation matrix can now be used to construct the rotation matrix,  $[R]$ .

$$[R] = \begin{bmatrix} T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{bmatrix} \quad (6)$$

Note that the element rotation matrix,  $[R]$ , is a  $12 \times 12$  matrix. The element rotation matrix is now capable of transferring the coordinates of an element between the global and local coordinate systems. Specifically we will use the rotation matrix to transfer the stiffness matrix, nodal displacements, and nodal forces from global coordinates to local coordinates.

## 2.4 Governing Stiffness Equations in Global Coordinates

### 2.4.1 Transformation of the Element Stiffness Equations to Global Coordinates

To use the  $[K]$  we would have to transform the values in the matrix from local to global coordinates. To do so we first would have to determine the stiffness matrix of each element in global coordinates by using the following set of equations.

$$\{u\}_l = [R]\{u\}_g \quad (7)$$

$$\{s\}_l = [R]\{s\}_g \quad (8)$$

By substituting Equations 7 and 8 into Equation 4 and pre-multiplying both sides by  $[R]^T$  we then can determine Equation 9, as seen below.

$$\{s\}_g = [R]^T [K]_l [R] \{u\}_g \quad (9)$$

Lastly we can either divide both sides by the  $\{u\}_g$  matrix to determine Equation 10, or simplify  $[R]^T [K]_l [R]$  to determine Equation 11.

$$[K]_g = [R]^T [K]_l [R] \quad (10)$$

$$[K]_g \{u\}_g = \{s\}_g \quad (11)$$

Note that the size of  $[K]_l$  is not altered in this matrix manipulation in Equation 10, so  $[K]_g$  is still of size  $12 \times 12$ .

#### 2.4.2 Assembly of Results to form a Global Stiffness Matrix and Finding the Nodal Displacements

After calculating each element stiffness equations are determined these results can now be compiled into a larger global stiffness matrix which contains values for the entire 3D beam model. The matrix is constructed as seen in Equation 12.

$$[K]_G = \begin{bmatrix} K_{11} & K_{12} & K_{13} & \dots \\ K_{21} & K_{22} & K_{23} & \dots \\ K_{31} & K_{32} & K_{33} & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \quad (12)$$

There are several important aspects about this matrix that need to be addressed. The first is that the size is based on the number of elements in the 3D beam model. So for any given model with  $e$  elements the  $[K]_G$  matrix will be of size  $6e \times 6e$ . Additionally each sub-matrix  $K_{ab}$ , where  $a$  and  $b$  are two node numbers, is of size  $6 \times 6$ , and is a quarter of an element stiffness matrix. Lastly if a sub-matrix is present in more than one elements stiffness matrix then the values are added together when constructing the global stiffness matrix.

Having the global stiffness matrix setup finding the nodal displacements was rather straightforward. To do so we used Equation 11 and solved for the  $\{u\}_g$  matrix.

Note that both the  $\{s\}_g$  and  $\{u\}_g$  matrices were of size  $e \times 1$ . Solving for the nodal displacements would now give us enough information to calculate the stresses.

## 2.5 Calculation of Stresses

### 2.5.1 Calculation of Element Loads

To calculate the loads we would first need to transform the global nodal displacements into local nodal displacements through the use of the rotation matrix. Then we would use Equation 4 again to determine the local nodal forces by using the known element stiffness matrices and the local nodal displacements that we just calculated.

### 2.5.2 Calculation of the Critical States of Stress

Knowing the loads that act on each element we could now determine the minimum and maximum states of stress at each of the nodes. To do so we would use Equation 13 and Equation 14.

$$\sigma_x = \pm \frac{Mc}{I} - \frac{P}{A} \quad (13)$$

$$\tau = \frac{Tr}{J} \quad (14)$$

The state of stress at each node is represented in Figure 6.

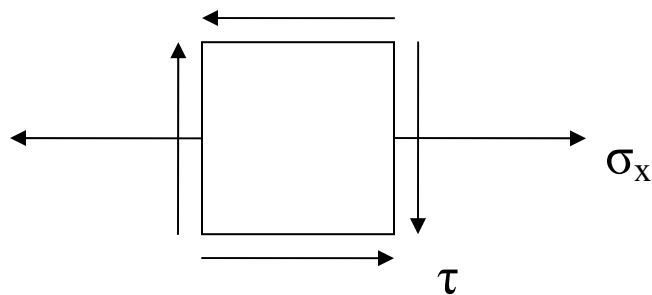


Figure 6 – A representation of the state of stress at a node.

With the state of stress determined we could now setup the stress tensor at each node. The stress tensor would only have values for  $\sigma_x$  and  $\tau_{xy}$ , with the rest of the values being zeros.

### 2.5.3 Evaluation of Principal Stresses

To evaluate the principal stresses at each node we would find the eigenvalues for each of the stress tensors at the nodes. Determining the principal stresses by this method would be the same as doing a Mohr's Circle analysis, but the method we will be using takes less steps and will be simpler and easier to carry out.

## 2.6 Results

### 2.6.1 Comparison with COSMOS

To confirm that the 3D beam model that was created was done correctly we would compare my finite element code to COSMOS, an existing finite element code. To do this we took an arbitrary geometry as seen in Figure 7.

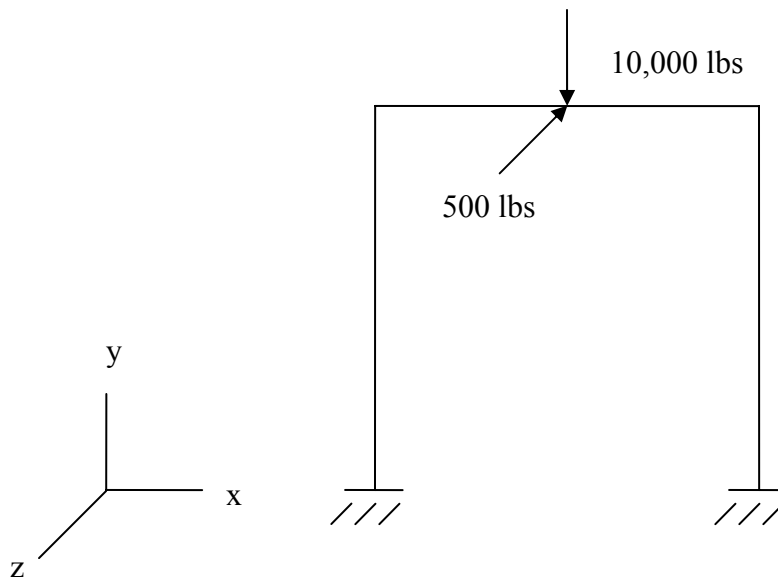


Figure 7 – The geometry used to compare my finite element code to the existing COSMOS code.

This arbitrary geometry was run through both the finite element code and COSMOS code, and both yielded the same displacements and stresses, which confirmed that the code could correctly analyze an arbitrary geometry.

### 2.6.2 Plotting Results

We additionally created another script in the MATLAB program that would allow for the results to be plotted, as seen in Figure 8. The areas of higher stress are noted by the denominations seen in the colorbar, with the red areas being the maximum stress. The significance of this is not the actual values seen here because an arbitrary geometry was used, but rather the significance is that this script works. It was important to confirm that this script worked because we will be using it later on in the project when we have to analyze a feather shaft.

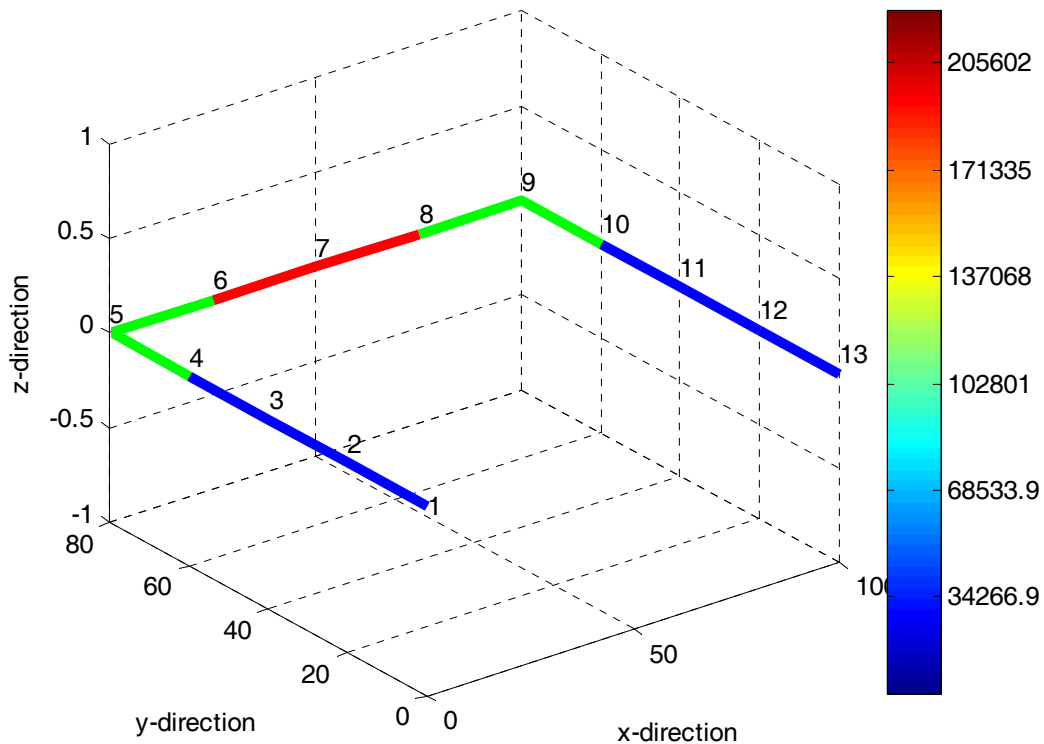


Figure 8 – Results from my finite element code for an arbitrary geometry.

## **2.7 Summary**

We have developed a 3D beam model that is able to handle any arbitrary geometry and determine and plot the stresses for a given pressure distribution. The code takes the material properties and geometries of the model, as well as the pressure distribution acting on the model as the inputs. With these inputs the code creates a model and meshes it with the desired number of elements. Through a series of stiffness equations the stresses for each element is determined, and these results are outputted into a plot as seen in Figure 8. This 3D beam code will be used inside an optimization code to determine the loading on a feather shaft.

## 3 Determination of Loading on a 3D Beam by Optimization

### 3.1 Constant Stress and Constant Stiffness Hypotheses

In our optimization we will be making an assumption based off of the constant stress and constant stiffness hypotheses. This assumption allows us to assume that the entire length of the feather shaft will fail together, instead of having the feather shaft fail at one point first. It is reasonable to make this assumption because we are basing it off the fact that evolution should produce an optimized structure. Later we will be able to validate these assumptions as long as we find a realistic loading for the feather shaft.

### 3.2 Function Being Minimized

In our optimization we will be minimizing the sum of the squares of the difference between the ultimate stress and maximum stress, Equation 15.

$$\sum_{i=1}^N (\sigma_u - \sigma_{1_i})^2 \quad (15)$$

Where  $\sigma_u$  is the ultimate stress of the feather shaft,  $\sigma_{1_i}$  is the maximum stress in each of the elements, and  $N$  is equal to the number of elements in the model.  $\sigma_{1_i}$  was determined for each element in the model by using a stress and strain analysis based off of the pressure distribution. This function is based on maximum normal stress theory, which is acceptable for beam bending, and when we consider torsion it will be based on one of the more conservative brittle failure theories. Note that the stresses in this problem depend on the loads, geometry, and material properties. In this problem we are holding the geometry and material properties constant and are looking to find the loads acting on the feather shaft.

### 3.3 Parameterization of Loads

The pressure distribution along the beam can be expressed using finite element shape functions as seen in Equation 16.

$$P = h_1 P_1 + h_2 P_2 + h_3 P_3 + h_4 P_4 \quad (16)$$

Where the numbered P values are the four pressure nodes as seen in Figure 9, and the numbered h values are based of a local coordinate system, r, that varies between -1 and 1, with -1 being at the left end of the beam, 1 being at the right end of the beam, and 0 being in the center.

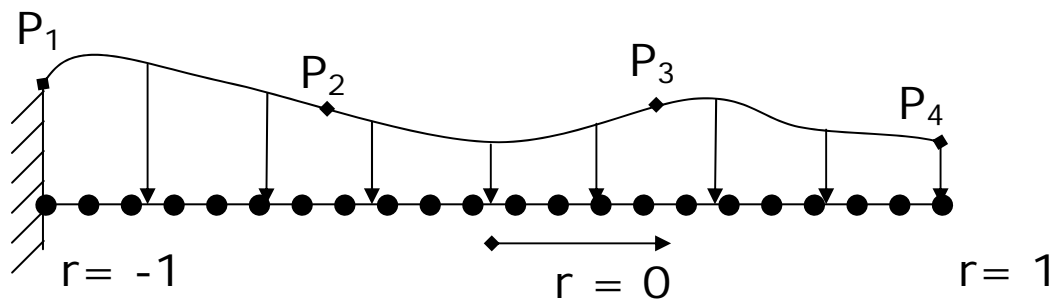
$$h_1 = \frac{1}{16}(1-r)(-1+9r^2) \quad (17)$$

$$h_2 = \frac{9}{16}(1-3r)(1-r^2) \quad (18)$$

$$h_3 = \frac{9}{16}(1+3r)(1-r^2) \quad (19)$$

$$h_4 = \frac{1}{16}(1+r)(-1+9r^2) \quad (20)$$

Also an r value of -1 corresponds to P<sub>1</sub>, an r value of -.333 corresponds to P<sub>2</sub>, an r value .333 corresponds to P<sub>3</sub>, and an r value of 1 corresponds to P<sub>4</sub>. What these correspondent values mean is that when r is equal to these values the P vector is only equal to only that P node, so when the r value is 1 the P vector will be equal to P<sub>4</sub>.



**Figure 9 – An arbitrary pressure distribution along a cantilever beam.**

Note how the locations of pressure nodes do not necessarily correspond to the locations of beam element nodes. This parameterization was introduced to allow for a much simpler optimization. Only four pressure nodes needed to be optimized because by using the local r coordinate system and shape functions the loads anywhere along the beam were easily determined from the pressure nodes. If we did not include this parameterization the optimization would be much more complicated since it would have as many inputs as there were elements, which could be upwards of 100 depending on the model used.

The nodal forces were calculated in our MATLAB scripts based off of the pressure distribution and were done so by using Equations 21 through 24.

$$F_r = \int_{r_a}^{r_b} (h_1 P_1 + h_2 P_2 + h_3 P_3 + h_4 P_4) \left( \frac{x_a - x_b}{r_b - r_a} \right) dr \quad (21)$$

$$\bar{x} F_r = \int_{r_a}^{r_b} (h_a x_a + h_b x_b) (h_1 P_1 + h_2 P_2 + h_3 P_3 + h_4 P_4) \left( \frac{x_a - x_b}{r_b - r_a} \right) dr \quad (22)$$

$$F_r = F_a + F_b \quad (23)$$

$$\bar{x} F_r = x_a F_a + x_b F_b \quad (24)$$

With these equations we have 4 equations as well as four unknowns and are mainly interested in solving for  $F_a$  and  $F_b$ , which are the nodal forces for each element that is analyzed with this procedure. Through this process we are able to determine all of the nodal forces on any beam element as desired.

### 3.4 Minimization Methods

#### 3.4.1 Minimization by the Simplex Method

The simplex method is a minimization method that is based off of a series of steps to determine the lowest point in a given function. This is a search method that does not use derivatives, and because of this reason is easier than implement compared to methods that utilize derivatives. In our minimization this method will be carried out through the MATLAB function *fminsearch* (*FUN*, *P*), where *FUN* is the function that is being minimized and *P* is an initial guess for the pressure distribution. The output of this function is the minimum value for the function *FUN* as well as the *P* values that yield this minimum value. If needed, other information can also be obtained such as the number of iterations and whether or not the optimization was terminated early.

#### 3.4.2 Minimization by the Univariate Method

A univariate search method of minimization will also be used to compare against the results that we receive from our simplex method optimization. This method is carried out by altering only one of the nodal pressures at a time while holding the others constant. After a minimum function value is found at each pressure node separately the

program is then run through each of the variables again until the minimum value of the function does not significantly change as the program continues to run.

### 3.5 Comparison of Minimization Methods

We will also compare the two optimization methods for a simple arbitrary function with a known minimum value, Equation 25, to see how the two methods differ from one another.

$$f(x_1, x_2) = (x_1 - 5)^2 + (x_2 - 10)^2 \quad (25)$$

By looking at this function it quickly can become apparent that the minimum should occur when  $x_1 = 5$  and  $x_2 = 10$ . If we use the *fminsearch* function we find the results seen in Table 1.

**Table 1 – A table of the results found from the optimization of Equation 25 with the simplex method.**

Starting $x_1, x_2$	Final Value	Final $x_1, x_2$	# of Function Evaluations	Time Taken (s)
0,0	$2.1833 \cdot 10^{-9}$	5,10	74	.02227
2,15	$5.7236 \cdot 10^{-10}$	5,10	49	.01863
13,3	$5.8755 \cdot 10^{-10}$	5,10	53	.01936
20,25	$6.9114 \cdot 10^{-10}$	5,10	50	.01894

These results are what were expected since we got final function values close to 0 and final values for  $x_1$  and  $x_2$  of 5 and 10 respectively as expected. The number of iterations and time taken to run each of these cases was similar to one another as expected. Now we can look at similar results for that of the univariate code in Table 2.

**Table 2 – A table of the results found from the optimization of Equation 25 with the univariate method.**

Starting $x_1, x_2$	Final Value	Final $x_1, x_2$	# of Function Evaluations	Time Taken (s)
0,0	0	5,10	50	.02037
2,15	0	5,10	50	.01960
13,3	0	5,10	50	.02013
20,25	0	5,10	50	.02039

Note that for this method the number of function evaluations is set because the user is able to set the values that values that are evaluated. This method also only worked so

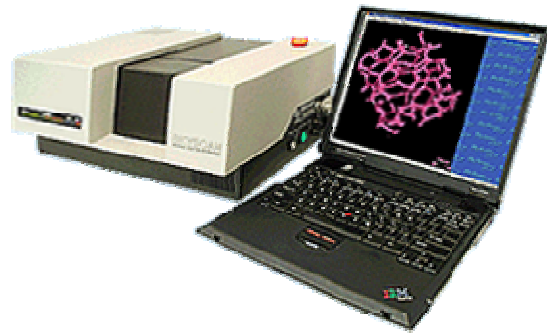
well because of how simple this function is, and if the function was more complicated the amount of time this code would need to find the minimum would be significantly more than the simplex method. These results again were what was expected. The function converged to 0 and the values for  $x_1$  and  $x_2$  again came out to 5 and 10 respectively. Since these are different codes we cannot easily compare the number of iterations, but we are able to compare the time that each code took to run, and by looking at the two tables we can see that these times are very similar to one another.

## 4 Characterization of Feather Shaft Geometry

### 4.1 CT Scanning

#### 4.1.1 Equipment

To carry out the CT scanning we used a Skyscan 1074 that Prof. Rapoff has in his laboratory, Figure 10.



**Figure 10 – An image of the Skyscan 1074 [3].**

This machine runs off of an x-ray power source, and works in a very similar way to that of a CAT scan machine used in most hospitals. The only difference is that in a hospital the machine rotates around the person, while in this application the specimen being scanned is rotated on a platform. During the scanning process the machine rotates  $.9^\circ$  between each image that it takes of the specimen. Also due to the size of the machine it is only able to handle very small specimens. The machine itself is hooked up to a computer and is operated by an associated Skyscan program that allows the user to operate and alter options during the scanning process.

#### 4.1.2 Calibrations

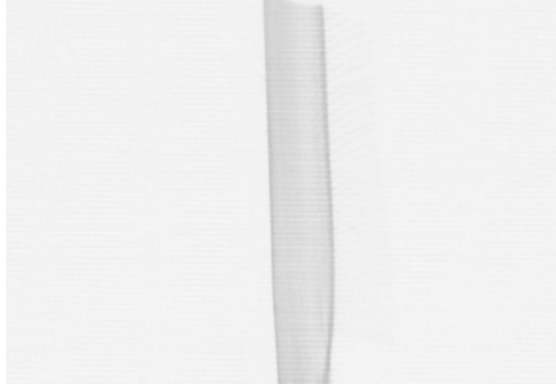
Before we were able to scan the feather shafts the machine first needed to be calibrated each time it was used for scanning. To do so an alignment, vertical, and flat field calibrations were done. The alignment calibration was done using a long vertical object, the vertical calibration was done using a round object, and the flat field calibration

was done with no object in the scanner. These calibrations were done to ensure that the object being scanned was done so in a correct manner.

### **4.1.3 Scanning Process**

The actual scanning of the feather had to be done through multiple scans because of the fact that the scanner can only fit objects with a height of 12mm, while the Canada Goose feather that we were scanning was 175mm in height. To overcome this obstacle the feather was cut into a number of sections that would each be separately scanned due to the size limitations of the scanner. The feather shaft was cut using a sharp set of scissors. I decided to use scissors after testing a number of cutting methods on an extra feather shaft. The data from each of these scans could later be compiled to completely model the entire feather shaft.

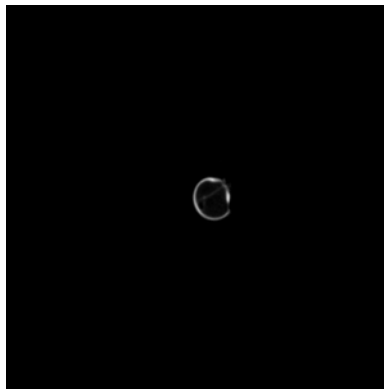
At this point the feather shaft was cut into sections and was ready to begin the scanning process. We started by scanning the section of the shaft closest to the wing and would move outwards along the shaft in the later scans. The first section was inserted into the scanner and propped up into the field of view of the scanner using moldable putty. Before running the scan we confirmed that the feather was correctly placed within the scanner in the field of view, and once this was done we ran the data acquisition feature of the Skyscan program, which was associated with the scanner. The outputs from this program are TIF files, and the images are taken directly from the scanner. For each specimen scanned there were 400 TIF images outputted, with each image being rotated  $.9^\circ$  more than the one previous to it. An arbitrary TIF image can be seen below in Figure 11.



**Figure 11 – An arbitrary TIF file of the Canada Goose outputted from the Skyscan program associated with the Skyscan 1074.**

The feather shaft can be seen as the darker object in the middle of the image, and if you look closely you can also see veins of the feather shaft to the right side of the darker object. Normally these veins would be present on both sides of the feather shaft, but we removed them so that the feather shaft could be more easily scanned.

With the TIF files created the next step in the process to obtain images of the cross sections of the feather shaft was to use a second program known as NRecon. This program took the series of TIF images as inputs from the previous program and outputted 400 bitmap images of the cross section over the length of the 12mm section that was scanned. An arbitrary bitmap image of a cross section of the feather shaft can be seen below in Figure 12.

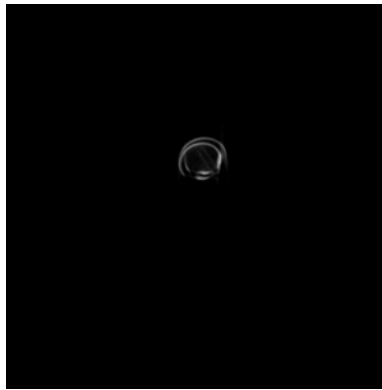


**Figure 12 – An arbitrary image of a cross section of the Canada Goose feather shaft.**

These images are done in gray scale, so that the lighter an area is, the denser it is; and the darker an area is, the less dense it is. This is an important piece of information that will be utilized when finding the cross sectional properties through the MATLAB codes written by Neel Bhatavadekar [4]. Also note that this MATLAB program automatically

determines a good range for the gray scale, so that this does not have to be set by the user. The shape of the feather shaft can be clearly seen in this image by the light circular ring. Also note that if you are to look carefully there is some additional material that can be seen inside the light ring, and in other cross sectional images there may be some additional material on the outside of the ring, which represents the veins of the feather. When these images are run through the MATLAB files to determine cross sectional it is important that this material be ignored and not factored into the calculations. Once these images are saved we were now able to run them through the MATLAB files to determine their cross sectional properties.

It is important to note that not all of the scans resulted in good cross sectional images as seen in Figure 12. Many of the scans resulted in poor results as seen in Figure 13.

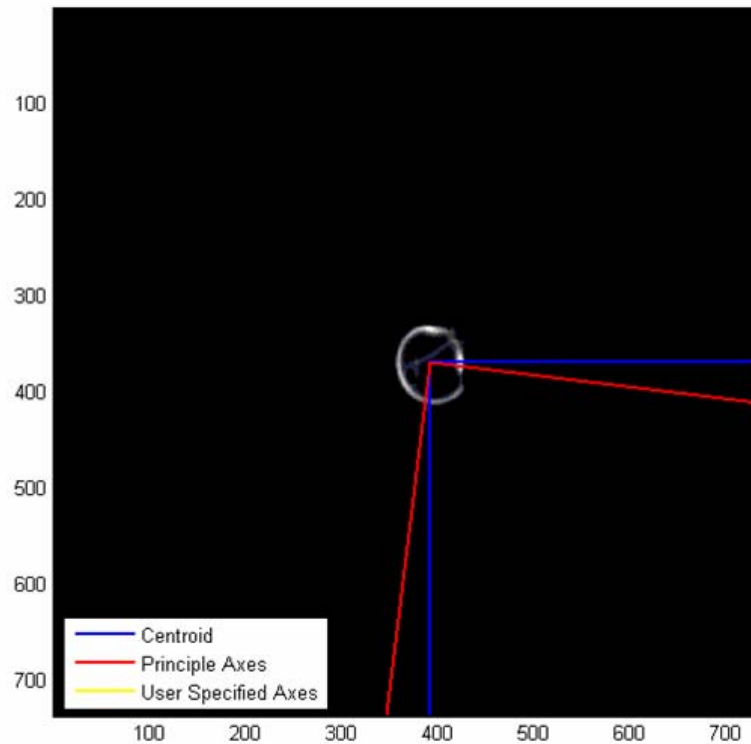


**Figure 13 – An arbitrary image of a bad cross sectional image of the Canada Goose feather shaft.** Note that in this image it looks as if there are two separate outer rings instead of one. These types of scans showed up frequently and considerably slowed down the process of gathering data on the Canada Goose feather shaft. The Canada Goose feather shaft was within the capabilities of the scanner, and it was unexplained why poor scans happened.

## **4.2 Cross Sectional Property Calculations through MATLAB**

The MATLAB files written by Neel Bhatavadekar would be very useful in our project to determine the cross sectional properties of the feather shaft. Note that the equations used to calculate  $I_x$ ,  $I_y$ , and the principal axes were all included in the work done by Neel Bhatavadekar. The only limitation to this capability was the fact that for a one of the scanned 12mm section of the 175mm feather shaft there were 400 bitmap cross

sectional images associated with it. It would be too time consuming to get the data for each of these 400 images and analyze all of the data. Instead we decided to only take the data every millimeter along the feather shaft, so that for each section only 12 images would need to be run through the MATLAB program. We were able to determine which images to run because of the fact that they were each numbered in order and spaced .028mm away from one another. After using these conditions to determine which images to run through the program we were able to use the revision of the program entitled `Moment_Calculations_Shree255pix.m`. The program outputs the centroidal and principal moments of inertia, as well as the area of the cross section. Additionally the angle, theta, between the centroidal and principal axes is outputted. In addition to this data, a modified version of the bitmap image is outputted as seen in Figure 14.



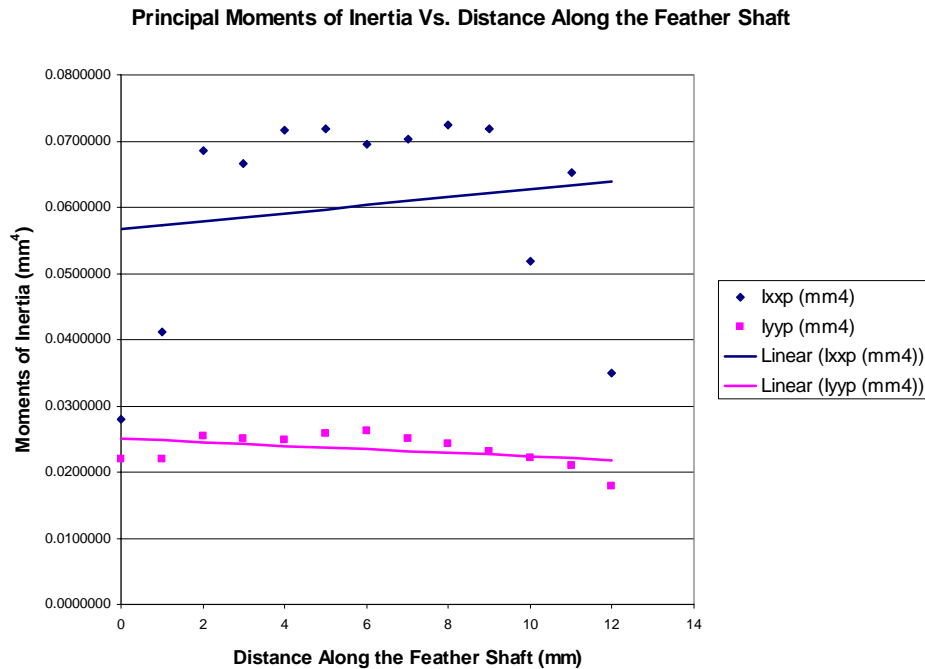
**Figure 14 – An image of the output from `Moment_Calculations_Shree255pix.m` for the same arbitrary cross section seen in Figure 12.**

The centroidal and principal axes are drawn onto the image and the user specified axes are at the edge of Figure 14.

Ultimately in this project we will not be as worried with the cross sectional properties through the principal axes of each cross section. Although we will be able to gather important data from looking at the principal axes of each cross section, it will not be true to what actual occurs when a bird is in flight. To best mimic the conditions when a bird is in flight we will align the axes that define the system with the vanes on the feather shaft. This will allow us to find results that best show the loading on a bird feather shaft when it is flight.

### 4.3 Preliminary Results

After running all of the bitmap images of the feather cross section through the MATLAB program we were able to compile the results to characterize the geometry of the feather shaft as desired for the analysis. Due to time limitations and issues with the scanning process we were only able to gather data for the first 12mm of the Canada Goose feather shaft as seen in Figure 15 below.



**Figure 15 - A graph of the preliminary results from the cross sectional data of the first 12mm of the Canada Goose feather shaft.**

The data in this graph came out as expected for the moments of inertia with respect to the y-axis in the sense that they decreased as we moved further out along the feather shaft. The same is not true for the moments of inertia with respect to the x-axis. This odd set of data can be explained by the fact that this data is such a small set of data so any nicks, dings, or scratches in the feather shaft could completely throw off the data. This data set is obviously not exactly what we were looking for but given all of the obstacles we had to overcome this data is still useful and is a start in the characterization of the geometry of the Canada Goose feather shaft.

#### **4.4 Conclusions**

The method that we used for the geometric characterization of the feather shaft was not the best for what we wanted to accomplish. To start we should have found a scanner that could handle larger specimens because of the fact that it made it very difficult to scan the feather shaft because we had to cut it up into a number of smaller pieces. Also note that there was a large amount of error associated with this cutting process. In addition the scanner that was used in this project was not always consistent, and after carrying out all of the necessary preliminary procedures it would still often give back poor results. Overall we could have chosen a better scanner for this project, but getting a new scanner would have been way outside of our financial constraints. If further geometric characterization is looking to be done in the future we would suggest that another method is utilized because of the very difficult time that we had with the Skyscan 1074.

## **5 Conclusions**

### **5.1 Summary**

The majority of our initial goals were realized by the end of this project. In MATLAB, we developed a 3D beam code within a simplex and univariate optimization code that could take any geometry as its input. As planned this code would determine the loading on the geometry based off of the constant stress hypothesis. Additionally, this code would provide an image of the geometry under this loading. We were able to confirm that this code works for a simple known geometry and now will be able to apply it to unknown geometries, such as bird feather shafts. We were also able to obtain cross sectional data for 12mm of a Canada Goose feather nearest to the wing. This data came out as expected with the highest moments of inertia occurring closest to the wing, and decreasing as we moved outward along the length of the feather shaft. Based off of these trends we were able to assume that the moments of inertia would continue to decrease as we continued to move outward along the length of the feather shaft, which was expected. This project has been very successful so far, and we are currently in the process of applying our optimization code to unknown feather shaft geometries. Additionally, once we gather more cross sectional data on the Canada Goose feather shaft we will be able to apply the optimization code to this geometry and determine the loading on this feather shaft as desired.

### **5.2 Future Work**

We were able to complete a substantial amount of work during the last two terms of this project, but there is still a lot more that can be done with this project especially on the geometric characterization side of the project.

The scanning method used in this project proved to not be very efficient, because of how time consuming it was, and the fact that the results were often times unreliable. The scanning time could be substantially reduced if the outputs from the scanner software could be converted into a SolidWorks model. This is because SolidWorks could then easily output the cross-sectional data desired. Additionally if more work is looking to be

done with the geometric characterization it is suggested that another scanner or a new method for determining the cross sectional be determined, because the current scanner is only able to handle very small specimens. It would be preferred if the object looking to be scanned did not have to be altered so that it could fit into the scanning device and if the scanner gave back more reliable results.

Once these scanning issues were worked out it would also be interesting to compare data for a variety of birds. It is known that for the most part birds have similar wing and feather anatomy [1], so it would be interesting to see whether or not the results were similar for different shapes and sizes of birds. This comparison should not take too much more time given the work we have accomplished on this project. This topic has not had a lot of research done on it, which leaves open a lot of possibilities. Hopefully the work done on this project and in the work done in the future will be able to help us not only gain a better understand of bird flight, but also use this information to help us improve the way that we fly today.

## 6 References

1. Rapoff AJ. Toward an Understanding of the Structures of the Hummingbird Wing. Union College, Schenectady, NY 2007.
2. Pennycuik CJ, Lock A. Elastic Energy Storage in Primary Feather Shafts. *Journal of Experimental Biology* 1976;64(3):677-89.
3. <http://www.microphotonics.com/skyscan/1076/skyscan1074.gif>
4. Bhatavadekar N. Image-Based Weighted Measure of Skeletal Stiffness: Case Studies of Great Ape Mandibles. University of Florida, Gainesville, FL 2004.

## 7 Appendix – MATLAB Scripts

Attached in this appendix are the MATLAB codes used in this project.

### 7.1 3D Beam Code Function

The following code is a MATLAB function for the 3D beam code that took the pressure nodes as an input, and output the squares of the difference between the ultimate stress and maximum stress at each of the elements.

```
%Alex Weintraub
%MER497
%OptimizationR6

%Create a function to be optimized
function output = optimizationR6(input)

%Convert the input to a value P, which represents the four pressure
nodes
%placed along the length of the feather shaft
P = input;
%P=abs(P);
%P=[-1099,114,-155,534];

%Run the input file
%Input the material properties
E = 30e6;
stress_ult = 700;
PR = .3;
G = E/(2*(1+PR));

%Input the geometry
r = 1;
J = pi/2;
Iy = pi/4;
Iz = pi/4;
A = pi;
Ltot=10.;

%Input an orienting point to determine the local coordinate systems
o = [50; 50; 0];

%Run the mesh file
%Define the number of elements
e = 20;

%Determine the nodal coordinates
dL = Ltot/e;
for a=1:e+1
    Xn(1,a)=(a-1)*dL;
    Xn(2,a)=0.;
```

```

    Xn(3,a)=0.;
end

%Determine the node connectivities
for a = 1:e
    CONN(a,:) = [a a+1];
end

%Determine the length of each element
for a = 1:e
    L_element(a) = sqrt((Xn(1,CONN(a,2))-Xn(1,CONN(a,1)))^2 + ...
        (Xn(2,CONN(a,2))-Xn(2,CONN(a,1)))^2 + (Xn(3,CONN(a,2))-
Xn(3,CONN(a,1)))^2);
end

%Define the forces
%Initialize the nodal forces
S = zeros(6,e+1);

%Determine the local nodal coordinates
for a = 1:e
    xa = Xn(1,CONN(a,1));
    xb = Xn(1,CONN(a,2));
    ra = -1 + 2*(Xn(1,CONN(a,1)) - Xn(1,1))/Ltot;
    rb = -1 + 2*(Xn(1,CONN(a,2)) - Xn(1,1))/Ltot;

    %Calculate Fr
    rl = linspace(ra,rb,100);
    y1 = P(1)/16.*(1-rl).*(-1+9.*rl.^2) + 9*P(2)/16.*(1-3.*rl).*(1-
rl.^2)...
        + 9*P(3)/16.*(1+3.*rl).*(1-rl.^2) + P(4)/16.*(1+rl).*(-
1+9.*rl.^2);
    c = (xb - xa)/(rb-ra);
    Fr = c*trapz(rl,y1);

    %Calculate xFr
    y2 = (1/2*(1-rl).*Xn(1,1) + 1/2*(1+rl).*Xn(1,e+1)).*(P(1)/16.*(1-
rl).*(-1+9.*rl.^2) + 9*P(2)/16.*(1-3.*rl).*(1-rl.^2) +
9*P(3)/16.*(1+3.*rl).*(1-rl.^2) + P(4)/16.*(1+rl).*(-1+9.*rl.^2));
    xFr = c*trapz(rl,y2);

    %Calculate Fa and Fb
    F = [1 1; xa xb]\[Fr; xFr];
    Fa = F(1);
    Fb = F(2);

    %Assemble the loads matrix
    S(2,a) = Fa + S(2,a);
    S(2,a+1) = Fb + S(2,a+1);
end

%Define the displacement boundary conditions
BC = zeros (6,e+1);
BC(1,1) = 1;
BC(2,1) = 1;

```

```

BC(3,1) = 1;
BC(4,1) = 1;
BC(5,1) = 1;
BC(6,1) = 1;

%Assemble the global stiffness matrix
Kg = zeros(6*(e+1),6*(e+1));
for a = 1:e
    L = L_element(a);

    %Determine the nodal coordinates
    Xa = Xn(1,CONN(a,1));
    Xb = Xn(1,CONN(a,2));
    Ya = Xn(2,CONN(a,1));
    Yb = Xn(2,CONN(a,2));
    Za = Xn(3,CONN(a,1));
    Zb = Xn(3,CONN(a,2));

    %Run the transformation matrix file
    %Define the local i term
    i_local = [Xb-Xa; Yb-Ya; Zb-Za]/sqrt((Xb-Xa)^2 + (Yb-Ya)^2 + (Zb-
Za)^2);

    %Define the 0 term
    O = [o(1)-Xa; o(2)-Ya; o(3)-Za]/sqrt((o(1)-Xa)^2 + (o(2)-Ya)^2 +
(o(3)-Za)^2);

    %Determine the local k term
    k_local = [i_local(2)*O(3)-i_local(3)*O(2); -(i_local(1)*O(3)-
i_local(3)*O(1)); i_local(1)*O(2)-
i_local(2)*O(1)]/sqrt((i_local(2)*O(3)-i_local(3)*O(2))^2 +
(i_local(1)*O(3)-i_local(3)*O(1))^2 + (i_local(1)*O(2)-
i_local(2)*O(1))^2);

    %Determine the local j term
    j_local = [k_local(2)*i_local(3)-k_local(3)*i_local(2); -
(k_local(1)*i_local(3)-k_local(3)*i_local(1)); k_local(1)*i_local(2)-
k_local(2)*i_local(1)]/sqrt((k_local(2)*i_local(3)-
k_local(3)*i_local(2))^2 + (k_local(1)*i_local(3)-
k_local(3)*i_local(1))^2 + (k_local(1)*i_local(2)-
k_local(2)*i_local(1))^2);

    %Assemble the rotation matrix
    T = [i_local(1), i_local(2), i_local(3);j_local(1), j_local(2),
j_local(3);k_local(1), k_local(2), k_local(3)];

    %Assemble the rotaton matrix
    R = zeros(12,12);
    R(1:3,1:3) = T;
    R(4:6,4:6) = T;
    R(7:9,7:9) = T;
    R(10:12,10:12) = T;

    %Run the stiffness matrix file
    %Initialize the stiffness matrix to zero

```

```

K = zeros(12);

%Determine all non-zero terms with a minimized number of
calculations

%Determine the EA/L terms
K(1,1) = E*A/L;
K(7,7) = K(1,1);
K(1,7) = -K(1,1);
K(7,1) = -K(1,1);

%Determine the 12EIz/L^3 terms
K(2,2) = 12*E*Iz/L^3;
K(8,8) = K(2,2);
K(2,8) = -K(2,2);
K(8,2) = -K(2,2);

%Determine the 12EIy/L^3 terms
K(3,3) = 12*E*Iy/L^3;
K(9,9) = K(3,3);
K(3,9) = -K(3,3);
K(9,3) = -K(3,3);

%Determine the GJ/L terms
K(4,4) = G*J/L;
K(10,10) = K(4,4);
K(4,10) = -K(4,4);
K(10,4) = -K(4,4);

%Determine the 4EIy/l terms
K(5,5) = 4*E*Iy/L;
K(11,11) = K(5,5);

%Determine the 4EIz/l terms
K(6,6) = 4*E*Iz/L;
K(12,12) = K(6,6);

%Determine the 6EIy/L^2
K(3,5) = -6*E*Iy/L^2;
K(5,3) = K(3,5);
K(3,11) = K(3,5);
K(11,3) = K(3,5);
K(5,9) = -K(3,5);
K(9,5) = -K(3,5);
K(9,11) = -K(3,5);
K(11,9) = -K(3,5);

%Determine the 6EIz/L^2 terms
K(2,6) = 6*E*Iz/L^2;
K(6,2) = K(2,6);
K(2,12) = K(2,6);
K(12,2) = K(2,6);
K(6,8) = -K(2,6);
K(8,6) = -K(2,6);
K(8,12) = -K(2,6);

```

```

K(12,8) = -K(2,6);

%Determine the 2EIy/l terms
K(5,11) = 2*E*Iy/L;
K(11,5) = K(5,11);

%Determine the 2EIz/l terms
K(6,12) = 2*E*Iz/L;
K(12,6) = K(6,12);

%Determine the local stiffness matrix
Kl = R'*K*R;

%Determine the global stiffness matrix
Kg(6*CONN(a,1)-5:6*CONN(a,1),6*CONN(a,1)-5:6*CONN(a,1)) =
Kl(1:6,1:6) + Kg(6*CONN(a,1)-5:6*CONN(a,1),6*CONN(a,1)-5:6*CONN(a,1));
Kg(6*CONN(a,1)-5:6*CONN(a,1),6*CONN(a,2)-5:6*CONN(a,2)) =
Kl(1:6,7:12) + Kg(6*CONN(a,1)-5:6*CONN(a,1),6*CONN(a,2)-5:6*CONN(a,2));
Kg(6*CONN(a,2)-5:6*CONN(a,2),6*CONN(a,1)-5:6*CONN(a,1)) =
Kl(7:12,1:6) + Kg(6*CONN(a,2)-5:6*CONN(a,2),6*CONN(a,1)-5:6*CONN(a,1));
Kg(6*CONN(a,2)-5:6*CONN(a,2),6*CONN(a,2)-5:6*CONN(a,2)) =
Kl(7:12,7:12) + Kg(6*CONN(a,2)-5:6*CONN(a,2),6*CONN(a,2)-
5:6*CONN(a,2));
end

%Impose displacement boundary conditions
for a = 1:e+1
    for b = 1:6
        if BC(b,a) == 1
            Kg(6*a+b-6,6*a+b-6) = Kg(6*a+b-6,6*a+b-6)*10e8;
        else
            Kg(6*a+b-6,6*a+b-6) = Kg(6*a+b-6,6*a+b-6);
        end
    end
end

%Calculate the global nodal displacements
U_column = Kg\S(:);
for a = 1:e+1
    U(:,a) = U_column(6*a-5:6*a);
end

%Initialize the local node forces matrix
Sn = zeros(12,e);

%Determine the global nodal coordinates
stress_prin = zeros(12,e);
for a = 1:e
    L = L_element(a);

    Xa = Xn(1,CONN(a,1));
    Xb = Xn(1,CONN(a,2));
    Ya = Xn(2,CONN(a,1));
    Yb = Xn(2,CONN(a,2));

```

```

Za = Xn(3,CONN(a,1));
Zb = Xn(3,CONN(a,2));

%Run the transformation matrix file
%Define the local i term
i_local = [Xb-Xa; Yb-Ya; Zb-Za]/sqrt((Xb-Xa)^2 + (Yb-Ya)^2 + (Zb-
Za)^2);

%Define the 0 term
O = [o(1)-Xa; o(2)-Ya; o(3)-Za]/sqrt((o(1)-Xa)^2 + (o(2)-Ya)^2 +
(o(3)-Za)^2);

%Determine the local k term
k_local = [i_local(2)*O(3)-i_local(3)*O(2); -(i_local(1)*O(3)-
i_local(3)*O(1)); i_local(1)*O(2)-
i_local(2)*O(1)]/sqrt((i_local(2)*O(3)-i_local(3)*O(2))^2 +
(i_local(1)*O(3)-i_local(3)*O(1))^2 + (i_local(1)*O(2)-
i_local(2)*O(1))^2);

%Determine the local j term
j_local = [k_local(2)*i_local(3)-k_local(3)*i_local(2); -
(k_local(1)*i_local(3)-k_local(3)*i_local(1)); k_local(1)*i_local(2)-
k_local(2)*i_local(1)]/sqrt((k_local(2)*i_local(3)-
k_local(3)*i_local(2))^2 + (k_local(1)*i_local(3)-
k_local(3)*i_local(1))^2 + (k_local(1)*i_local(2)-
k_local(2)*i_local(1))^2);

%Assemble the rotation matrix
T = [i_local(1), i_local(2), i_local(3);j_local(1), j_local(2),
j_local(3);k_local(1), k_local(2), k_local(3)];

%Assemble the rotaton matrix
R = zeros(12,12);
R(1:3,1:3) = T;
R(4:6,4:6) = T;
R(7:9,7:9) = T;
R(10:12,10:12) = T;

%Extract the global nodal displacements
U_extract(1:6,1) = U_column(6*CONN(a,1)-5:6*CONN(a,1));
U_extract(7:12,1) = U_column(6*CONN(a,2)-5:6*CONN(a,2));

%Transform the global nodal displacements into the local nodal
%displacements
U1 = R*U_extract;

%Run the stiffness matrix file
%Initialize the stiffness matrix to zero
K = zeros(12);

%Determine all non-zero terms with a minimized number of
calculations

%Determine the EA/L terms
K(1,1) = E*A/L;

```

```

K(7,7) = K(1,1);
K(1,7) = -K(1,1);
K(7,1) = -K(1,1);

%Determine the 12EIz/L^3 terms
K(2,2) = 12*E*Iz/L^3;
K(8,8) = K(2,2);
K(2,8) = -K(2,2);
K(8,2) = -K(2,2);

%Determine the 12EIy/L^3 terms
K(3,3) = 12*E*Iy/L^3;
K(9,9) = K(3,3);
K(3,9) = -K(3,3);
K(9,3) = -K(3,3);

%Determine the GJ/L terms
K(4,4) = G*J/L;
K(10,10) = K(4,4);
K(4,10) = -K(4,4);
K(10,4) = -K(4,4);

%Determine the 4EIy/l terms
K(5,5) = 4*E*Iy/L;
K(11,11) = K(5,5);

%Determine the 4EIz/l terms
K(6,6) = 4*E*Iz/L;
K(12,12) = K(6,6);

%Determine the 6EIy/L^2 terms
K(3,5) = -6*E*Iy/L^2;
K(5,3) = K(3,5);
K(3,11) = K(3,5);
K(11,3) = K(3,5);
K(5,9) = -K(3,5);
K(9,5) = -K(3,5);
K(9,11) = -K(3,5);
K(11,9) = -K(3,5);

%Determine the 6EIz/L^2 terms
K(2,6) = 6*E*Iz/L^2;
K(6,2) = K(2,6);
K(2,12) = K(2,6);
K(12,2) = K(2,6);
K(6,8) = -K(2,6);
K(8,6) = -K(2,6);
K(8,12) = -K(2,6);
K(12,8) = -K(2,6);

%Determine the 2EIy/l terms
K(5,11) = 2*E*Iy/L;
K(11,5) = K(5,11);

%Determine the 2EIz/l terms

```

```

K(6,12) = 2*E*Iz/L;
K(12,6) = K(6,12);

%Calculate the local nodal forces for the element
Sn1 = K*U1;
Sn(1:12,a) = Sn1;

%Calculate the local maximum and minimum stress states at the nodes
stress(1,a) = sqrt(Sn1(5)^2+Sn1(6)^2)*r/Iy - Sn1(1)/A;
stress(2,a) = -sqrt(Sn1(5)^2+Sn1(6)^2)*r/Iy - Sn1(1)/A;
stress(3,a) = Sn1(4)*r/J;
stress(4,a) = sqrt(Sn1(11)^2+Sn1(12)^2)*r/Iy + Sn1(7)/A;
stress(5,a) = -sqrt(Sn1(11)^2+Sn1(12)^2)*r/Iy + Sn1(7)/A;
stress(6,a) = Sn1(10)*r/J;

%Calculate the principle stresses
stress_prin(1:3,a) = eig([stress(1,a) stress(3,a) 0;stress(3,a) 0
0;0 0 0]);
stress_prin(4:6,a) = eig([stress(2,a) stress(3,a) 0;stress(3,a) 0
0;0 0 0]);
stress_prin(7:9,a) = eig([stress(4,a) stress(6,a) 0;stress(6,a) 0
0;0 0 0]);
stress_prin(10:12,a) = eig([stress(5,a) stress(6,a) 0;stress(6,a) 0
0;0 0 0]);
end

%Determine the maximum stress for each node and overall
stress_max = max(abs(stress_prin));
stress_max_1 = max(stress_max);

for a = 1:e
    stress_diff(a) = (stress_ult - stress_max(a)).^2;
end

output = sum(stress_diff);

```

## 7.2 Univariate Optimization Code

The following code is the univariate optimization code used along with the 3D beam code function. Note that the bounds on the four pressure nodes can easily be changed by the user depending on the model that is being looked at.

```

%Alex Weintraub
%MER497
%Univariate Optimization

%Clear
clear
clc

%Input starting values and set the iteration counter
xa_min = 1;
xb_min = 1;

```

```

xc_min = 1;
xd_min = 1;
x = [xa_min,xb_min,xc_min,xd_min];
count = 0;

%Run the program
while count<5

    A = 0:10000:1000000;
    min = inf;
    for i = 1:length(A)
        cost = optimizationR6([A(i),xb_min,xc_min,xd_min]);
        if cost<min
            min = cost;
            xa_min = A(i);
        end
    end

    B = 0:10000:1000000;
    min = inf;
    for i = 1:length(B)
        cost = optimizationR6([xa_min,B(i),xc_min,xd_min]);
        if cost<min
            min = cost;
            xb_min = B(i);
        end
    end

    C = 0:10000:1000000;
    min = inf;
    for i = 1:length(C)
        cost = optimizationR6([xa_min,xb_min,C(i),xd_min]);
        if cost<min
            min = cost;
            xc_min = C(i);
        end
    end

    D = 0:10000:1000000;
    min = inf;
    for i = 1:length(D)
        cost = optimizationR6([xa_min,xb_min,xc_min,D(i)]);
        if cost<min
            min = cost;
            xd_min = D(i);
        end
    end

    count = count + 1;
    x = [xa_min,xb_min,xc_min,xd_min]
end

```